Data Mining Phase 4

Faridreza Momtazzandi 9812762601 Alireza Nourbakhsh 9812762496

We finally reached our last and fourth phase which is to use all the knowledge we got from previous phases and finally answer the main questions we're facing!

At this point, we know our datasets and we're familiar with their shortcomings so if our employer asks us some critical questions about these datasets and try to get as much detail possible on how he wants to use this dataset in his favor we can now try to use all we know and answer them.

As we saw in phase Three, our main focus is to look for incoming and outcoming assets and after getting information on three main topics which are the Incoming rate of assets, Count and Value of assets based on Holding and Draft or finalized accounting document we can move on to what we need to do in this phase.

Like the last phase, we need to discuss three main topics and we get into detail on each one of them in their parts.

1.The ratio of bought assets to returned assets based on the organization

Unlike the last phase which we focused on incoming assets based on holding and accounting units, in this part, we're trying to work on organizations and as we look into our datasets we learn some attributes are used to demonstrate organization such as AD_ORG_ID and REF_ORG in INOUT dataset and AD_ORG_REF_ID in PRODUCTINSTANCE dataset.

In [1]:
```
import pandas as pd
```

In [2]:
```
inout_df = pd.read_csv('E:\ce\data mining\DataSets\INOUT.csv' , low_memory = False)
inoutline_df = pd.read_csv('E:\ce\data mining\DataSets\INOUTLINE.csv' , low_memory
productinstance_df = pd.read_csv('E:\ce\data mining\DataSets\PRODUCTINSTANCE.csv' ,
```

We want to look into bought asstes which is all the assets in our dataset and returned assets are assets that got aborted. first lets see how AD_ORG_ID, REF_ORG and AD_ORG_REF_ID are doing so we can use them to find what we're looking for.

```
In [3]: print("\033[94m let's see all the organizations base on AD_ORG_ID attribute")
        print('\033[90m')
        print(inout_df['AD_ORG_ID'].value_counts())
        print("\033[94m______________________________________________________________")
        print(" let's see all the organizations base on REF_ORG attribute")
        print('\033[90m')
        print(inout_df['REF_ORG'].value_counts())
        print("\033[94m______________________________________________________________")
        print(" let's see all the organizations base on AD_ORG_REF_ID attribute")
        print('\033[90m')
        print(productinstance_df['AD_ORG_REF_ID'].value_counts())
```

```
 let's see all the organizations base on AD_ORG_ID attribute

0            5717
104000002    1883
Name: AD_ORG_ID, dtype: int64
______________________________________________________________
 let's see all the organizations base on REF_ORG attribute

200000137.0    331
200000114.0    273
200000654.0    252
210000089.0    242
200000138.0    203
              ...
210000118.0      1
469638078.0      1
469638503.0      1
469638393.0      1
469637721.0      1
Name: REF_ORG, Length: 171, dtype: int64
______________________________________________________________
 let's see all the organizations base on AD_ORG_REF_ID attribute

106536008.0    42572
200000137.0     9096
200000138.0     5185
469732738.0     5012
469678455.0     3636
               ...
210033816.0        1
105005321.0        1
469685459.0        1
469739817.0        1
469637579.0        1
Name: AD_ORG_REF_ID, Length: 359, dtype: int64
```

With what we got from the code above we find an anomaly in our database although in our dictionary it's stated that AD_ORG_ID is the equivalent to REF_ORG their values are different so we can't see which of them can be useful for us so it's better to use AD_ORG_REF_ID attribute because not only it's in our PRODUCTINSTANCE dataset and it means we have this attribute for all incoming assets, also we can use RETURNAMVALTOANBAR attribute which as its name suggests is the state of returned assets. Let's see how this attribute works:

In [4]:
```
print("\033[94m let's see values of RETURNAMVALTOANBAR:")
print('\033[90m')
print(productinstance_df['RETURNAMVALTOANBAR'].value_counts())
```

```
 let's see values of RETURNAMVALTOANBAR:

4.0    6290
1.0    2703
3.0     257
2.0      18
Name: RETURNAMVALTOANBAR, dtype: int64
```

As we saw in the dictionary RETURNAMVALTOANBAR attribute can have four values which are: 1 for Used usable, 2 for The asset is required to repair, 3 for Separable abortion, and 4 for Inseparable abortion so we're looking out for 3 and 4 results. In the code below we try to find the ratio of bought assets to returned assets based on organization and use AD_ORG_REF_ID and RETURNAMVALTOANBAR attributes.

In [5]:
```
result = []

for ind in productinstance_df.index:
    if productinstance_df['RETURNAMVALTOANBAR'][ind] == 3:
        result.append(productinstance_df['AD_ORG_REF_ID'][ind])

    elif productinstance_df['RETURNAMVALTOANBAR'][ind] == 4:
        result.append(productinstance_df['AD_ORG_REF_ID'][ind])

for ind in pd.Series(result).value_counts().index:
    print(pd.Series(result).value_counts().loc[[ind]])
```

```
200000137.0    5685
dtype: int64
106536008.0    196
dtype: int64
469732738.0    183
dtype: int64
469638568.0    48
dtype: int64
200000127.0    28
dtype: int64
200000141.0    24
dtype: int64
200000142.0    23
dtype: int64
469638360.0    20
dtype: int64
210032657.0    17
dtype: int64
200000129.0    17
dtype: int64
469638515.0    15
dtype: int64
469747289.0    13
dtype: int64
469637613.0    12
dtype: int64
469732686.0    11
dtype: int64
210000017.0    10
dtype: int64
200000654.0    8
dtype: int64
469638115.0    8
dtype: int64
210000001.0    8
dtype: int64
469733251.0    8
dtype: int64
469637680.0    8
dtype: int64
469678178.0    7
dtype: int64
469732631.0    7
dtype: int64
200000615.0    7
dtype: int64
469732813.0    7
dtype: int64
200000220.0    6
dtype: int64
200000144.0    6
dtype: int64
200000653.0    6
dtype: int64
469738837.0    6
dtype: int64
```

```
469732633.0    5
dtype: int64
210000080.0    5
dtype: int64
469737873.0    5
dtype: int64
469737677.0    5
dtype: int64
210034398.0    5
dtype: int64
469637868.0    5
dtype: int64
200000627.0    4
dtype: int64
210034287.0    4
dtype: int64
469739828.0    4
dtype: int64
469736320.0    4
dtype: int64
210034190.0    4
dtype: int64
200000167.0    4
dtype: int64
200000407.0    3
dtype: int64
469738661.0    3
dtype: int64
210034667.0    3
dtype: int64
210034257.0    3
dtype: int64
469732682.0    3
dtype: int64
210034397.0    3
dtype: int64
210034740.0    3
dtype: int64
200000138.0    3
dtype: int64
210000071.0    3
dtype: int64
200000484.0    3
dtype: int64
200000471.0    3
dtype: int64
469732739.0    3
dtype: int64
210034616.0    2
dtype: int64
200000192.0    2
dtype: int64
469732626.0    2
dtype: int64
200000548.0    2
dtype: int64
```

```
210619769.0    2
dtype: int64
469732661.0    2
dtype: int64
210000089.0    2
dtype: int64
469637895.0    2
dtype: int64
200000115.0    2
dtype: int64
200000723.0    2
dtype: int64
210000087.0    2
dtype: int64
469637963.0    2
dtype: int64
200000663.0    2
dtype: int64
200000585.0    2
dtype: int64
469637973.0    2
dtype: int64
200000632.0    1
dtype: int64
469735605.0    1
dtype: int64
200000487.0    1
dtype: int64
469732652.0    1
dtype: int64
210034337.0    1
dtype: int64
469732720.0    1
dtype: int64
200000599.0    1
dtype: int64
469733076.0    1
dtype: int64
200000119.0    1
dtype: int64
469732655.0    1
dtype: int64
200000143.0    1
dtype: int64
469747188.0    1
dtype: int64
469732727.0    1
dtype: int64
469732810.0    1
dtype: int64
469638346.0    1
dtype: int64
210000011.0    1
dtype: int64
210000098.0    1
dtype: int64
```

```
469732708.0    1
dtype: int64
200000116.0    1
dtype: int64
469638544.0    1
dtype: int64
469735538.0    1
dtype: int64
469732629.0    1
dtype: int64
469740215.0    1
dtype: int64
469732659.0    1
dtype: int64
200000114.0    1
dtype: int64
200000634.0    1
dtype: int64
200000130.0    1
dtype: int64
469738783.0    1
dtype: int64
469732657.0    1
dtype: int64
469678455.0    1
dtype: int64
469638210.0    1
dtype: int64
200000715.0    1
dtype: int64
469732627.0    1
dtype: int64
```

After we wrote the code above we checked each orgonization for it's RETURNAMVALTOANBAR attribute and added to a list if it's 3 or 4 so we can now see how much returned assets each organization has, for example organization 200000137 has the most returned assets which is 5685, 106536008 and 469732738 comes after it. also we can see the organizations with the least assets and it's only 1.

Finally our first question is finished and we know the ratio of bought assets to returned assets based on organization so we can move on to next part.

2.Analysis of asset inflow to asset outflow by time and organization

For this part, we're gonna take a look at the inflow of assets to outflow assets and analyze them by time and organization. The only attributes we can use for this part are CREATED and UPDATED attributes which are in every dataset but we choose PRODUCTINSTANCE so we can make sure assets are all taken look at! let's see what these attributes returns:

In [6]:
```
print("\033[94m let's see all the values of CREATED attribute")
print('\033[90m')
print(productinstance_df['CREATED'].value_counts())
print("\033[94m______________________________________________________________")
print(" let's see all the values of UPDATED attribute")
print('\033[90m')
print(productinstance_df['UPDATED'].value_counts())
```

```
 let's see all the values of CREATED attribute

1/14/2015 12:33     212087
10/31/2019 7:11      79967
4/30/2015 10:02      61370
5/19/2017 12:02      34066
10/26/2014 14:39     12901
                     ...
5/16/2018 13:16          1
5/16/2018 13:10          1
5/16/2018 13:01          1
5/16/2018 12:49          1
60212                    1
Name: CREATED, Length: 92756, dtype: int64
______________________________________________________________
 let's see all the values of UPDATED attribute

10/31/2019 7:11    77656
2/12/2022 14:54    45377
5/19/2017 12:02    24423
1/14/2015 12:33    17944
6/1/2019 12:04      8887
                   ...
2/20/2021 9:19         1
4/23/2020 13:16        1
7/7/2021 13:03         1
9/11/2018 11:18        1
11/2/2019 7:27         1
Name: UPDATED, Length: 207714, dtype: int64
```

So What we need for this part other than CREATED and UPDATED attributes is an attribute to demonstrate organizations in which we use the AD_ORG_REF_ID attribute as we used before so let's get what we need for this part and start analyzing asset inflow to asset outflow by time and organization.

In [7]:
```
created = []
updated = []
organization = []
ind = len(productinstance_df.index) - 1
while ind >= 0:
    if not productinstance_df['CREATED'][ind] in created:
        created.append(productinstance_df['CREATED'][ind])
        updated.append(productinstance_df['UPDATED'][ind])
        organization.append(productinstance_df['AD_ORG_REF_ID'][ind])
    ind -= 1
```

What we did in our last cell was to get every individual created values and the last updated value it has assigned to it and also we got the organization that asset has. Let's check if our three lists are correct:

```
In [8]: print(len(created))
        print(len(updated))
        print(len(organization))
```

```
92757
92757
92757
```

As we see all the lists have exactly 92757 values and that's the number of our individual created values. Unfortunately printing 92757 values three times is time consuming so let's only print 5 examples of these lists:

```
In [9]: print(created[26], updated[26], organization[26])
        print(created[34], updated[34], organization[34])
        print(created[42], updated[42], organization[42])
        print(created[49], updated[49], organization[49])
        print(created[135], updated[135], organization[135])
```

```
9/12/2022 13:01 9/12/2022 13:01 200000137.0
9/12/2022 11:33 9/12/2022 11:33 469638568.0
9/12/2022 10:24 9/12/2022 10:24 200000137.0
9/12/2022 9:17 9/12/2022 9:17 200000137.0
9/8/2022 13:03 9/8/2022 13:03 200000137.0
```

After all the code we've written we can now have a complete understanding on our assets, their last updated value and their assigned organization. so we can now analyse and get any information we wish based in time and organization.

3.The status of similar products and the price prediction of each product according to its cluster

There are multiple ways to find similar assets such as looking for assets with updated time and price and clustering the items which share these attributes and end up with clusters of assets with the same year and almost close price.

Another way to face this part is to get an asset and try to predict its price for the upcoming year but with these datasets and how messy they are, we can't get a meaningful prediction.

As we saw in our last phase assets only came in the years 2013 to 2022 and for the first step of clustering we can get all the assets in 8 different clusters based on their year of 'CREATED' attribute, code below demonstrate this:

In [10]:
```
year_2013 = 0
year_2014 = 0
year_2015 = 0
year_2016 = 0
year_2017 = 0
year_2018 = 0
year_2019 = 0
year_2020 = 0
year_2021 = 0
year_2022 = 0

for ind in productinstance_df.index:
    if '2013' in str(productinstance_df['CREATED'][ind]):
        year_2013 += 1
    elif '2014' in str(productinstance_df['CREATED'][ind]):
        year_2014 += 1
    elif '2015' in str(productinstance_df['CREATED'][ind]):
        year_2015 += 1
    elif '2016' in str(productinstance_df['CREATED'][ind]):
        year_2016 += 1
    elif '2017' in str(productinstance_df['CREATED'][ind]):
        year_2017 += 1
    elif '2018' in str(productinstance_df['CREATED'][ind]):
        year_2018 += 1
    elif '2019' in str(productinstance_df['CREATED'][ind]):
        year_2019 += 1
    elif '2020' in str(productinstance_df['CREATED'][ind]):
        year_2020 += 1
    elif '2021' in str(productinstance_df['CREATED'][ind]):
        year_2021 += 1
    elif '2022' in str(productinstance_df['CREATED'][ind]):
        year_2022 += 1
print('\033[94m Lets see how much asset came in on each year and how we documented
print('\033[90m')
print('in 2013:' , year_2013)
print('in 2014:' , year_2014)
print('in 2015:' , year_2015)
print('in 2016:' , year_2016)
print('in 2017:' , year_2017)
print('in 2018:' , year_2018)
print('in 2019:' , year_2019)
print('in 2020:' , year_2020)
print('in 2021:' , year_2021)
print('in 2022:' , year_2022)
```

```
 Lets see how much asset came in on each year and how we documented it in PRODUCTIN
STANCE dataset:

in 2013: 3483
in 2014: 50848
in 2015: 292464
in 2016: 24760
in 2017: 71845
in 2018: 45691
in 2019: 132252
in 2020: 22441
in 2021: 24616
in 2022: 25657
```

Now let's do the same thing but this time add the asset's primal value to a list so we can cluster them later:

In [11]:
```
p_2013 = []
p_2014 = []
p_2015 = []
p_2016 = []
p_2017 = []
p_2018 = []
p_2019 = []
p_2020 = []
p_2021 = []
p_2022 = []


for ind in productinstance_df.index:
    if '2013' in str(productinstance_df['CREATED'][ind]):
        p_2013.append(productinstance_df['PRIMALVALUE'][ind])
    elif '2014' in str(productinstance_df['CREATED'][ind]):
        p_2014.append(productinstance_df['PRIMALVALUE'][ind])
    elif '2015' in str(productinstance_df['CREATED'][ind]):
        p_2015.append(productinstance_df['PRIMALVALUE'][ind])
    elif '2016' in str(productinstance_df['CREATED'][ind]):
        p_2016.append(productinstance_df['PRIMALVALUE'][ind])
    elif '2017' in str(productinstance_df['CREATED'][ind]):
        p_2017.append(productinstance_df['PRIMALVALUE'][ind])
    elif '2018' in str(productinstance_df['CREATED'][ind]):
        p_2018.append(productinstance_df['PRIMALVALUE'][ind])
    elif '2019' in str(productinstance_df['CREATED'][ind]):
        p_2019.append(productinstance_df['PRIMALVALUE'][ind])
    elif '2020' in str(productinstance_df['CREATED'][ind]):
        p_2020.append(productinstance_df['PRIMALVALUE'][ind])
    elif '2021' in str(productinstance_df['CREATED'][ind]):
        p_2021.append(productinstance_df['PRIMALVALUE'][ind])
    elif '2022' in str(productinstance_df['CREATED'][ind]):
        p_2022.append(productinstance_df['PRIMALVALUE'][ind])
```

With the code above we clustered all the PRIMALVALUES based on their year in CREATED attribute! What we suggest doing at this point is to find clusters with almost equal members and a similar range of price we think the best number to choose for the number of members is 25k for each cluster.

With each cluster only have about 25k members we can put each year's assets as comes below:

- 2013: 1 cluster
- 2014: 2 cluster
- 2015: 12 cluster
- 2016: 1 cluster
- 2017: 3 cluster
- 2018: 2 cluster
- 2019: 6 cluster
- 2020: 1 cluster
- 2021: 1 cluster
- 2022: 1 cluster

so we end up with exactly 30 clusters! and we'll sort each cluster based on its value but unfortunately, the PRODUCTINSTANCE dataset doesn't have enough records so we can't get any useful information but if our dataset was better by what we just did we could predict any product's price, we just need to know what year we want to check this product in and look for it in the cluster it belongs too.

As we finished the last phase of this project we learned that after getting to know our dataset, doing necessary preprocessing on it, and answering some real-world questions about it we can finally learn enough about our dataset to analyze, predict or any other data-related need our employer might have.