FRA Project(Milestone-1)

Problem Statement

Businesses or companies can fall prey to default if they are not able to keep up their debt obligations. Defaults will lead to a lower credit rating for the company which in turn reduces its chances of getting credit in the future and may have to pay higher interests on existing debts as well as any new obligations. From an investor's point of view, he would want to invest in a company if it is capable of handling its financial obligations, can grow quickly, and is able to manage the growth scale.

A balance sheet is a financial statement of a company that provides a snapshot of what a company owns, owes, and the amount invested by the shareholders. Thus, it is an important tool that helps evaluate the performance of a business.

Data that is available includes information from the financial statement of the companies for the previous year (2015). Also, information about the Networth of the company in the following year (2016) is provided which can be used to drive the labeled field.

Questions:

1.1 Outlier Treatment

1.2 Missing Value Treatment

1.3 Transform Target variable into 0 and 1

1.4 Univariate & Bivariate analysis with proper interpretation. (You may choose to include only those variables which were significant in the model building)

1.5 Train Test Split

1.6 Build Logistic Regression Model (using statsmodel library) on most important variables on Train Dataset and choose the optimum cutoff. Also showcase your model building approach

1.7 Validate the Model on Test Dataset and state the performance matrices. Also state interpretation from the model

Quality of Business report.(Please refer to the Evaluation Guidelines for Business report checklist. Marks in this criteria are at the moderator's discretion).

In [1]:
```
#Importing libraries

import numpy as np, pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn.metrics as metrics
```

```
color = sns.color_palette()
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#Loading dataset
df = pd.read_excel(r'D:\SHUBHANK !\GL\12. TOPIC 11 - FRA\Milestone 1 project\Company_Da
```

In [3]:
```
#Reading dataset
df.head()
```

Out[3]:

	Co_Code	Co_Name	Networth Next Year	Equity Paid Up	Networth	Capital Employed	Total Debt	Gross Block	Net Working Capital	Current Assets
0	16974	Hind.Cables	-8021.60	419.36	-7027.48	-1007.24	5936.03	474.30	-1076.34	40.50
1	21214	Tata Tele. Mah.	-3986.19	1954.93	-2968.08	4458.20	7410.18	9070.86	-1098.88	486.86
2	14852	ABG Shipyard	-3192.58	53.84	506.86	7714.68	6944.54	1281.54	4496.25	9097.64
3	2439	GTL	-3054.51	157.30	-623.49	2353.88	2326.05	1033.69	-2612.42	1034.12
4	23505	Bharati Defence	-2967.36	50.30	-1070.83	4675.33	5740.90	1084.20	1836.23	4685.81

5 rows × 67 columns

In [4]:
```
#Renaming the columns for better readability

df.columns = ['Co_Code','Co_Name','Networth_Next_Year','Equity_Paid_Up','Networth','Cap
              'Total_Debt','Gross_Block','Net_Working_Capital','Curr_Assets','Curr_Liab
              'Gross_Sales','Net_Sales','Other_Income','Value_Of_Output','Cost_of_Prod'
              'PBIT','PBT','PAT','Adjusted_PAT','CP','Rev_earn_in_forex','Rev_exp_in_fo
              'Book_Value_Unit_Curr','Book_Value_Adj_Unit_Curr','Market_Capitalisation'
              'Cash_Flow_From_Opr','Cash_Flow_From_Inv','Cash_Flow_From_Fin','ROG_Net_W
              'ROG_Capital_Employed_perc','ROG_Gross_Block_perc','ROG_Gross_Sales_perc'
              'OG_Cost_of_Prod_perc','ROG_Total_Assets_perc','OG_PBIDT_perc','ROG_PBDT_
              'ROG_PAT_perc','ROG_CP_perc','ROG_Rev_earn_in_forex_perc','ROG_Rev_exp_in
              'ROG_Market_Capitalisation_perc','Curr_Ratio_Latest','Fixed_Assets_Ratio_
              'Debtors_Ratio_Latest','Total_Asset_Turnover_Ratio_Latest','Interest_Cove
              'PBITM_perc_Latest','PBDTM_perc_Latest','CPM_perc_Latest','APATM_perc_Lat
              'Creditors_Vel_Days','Inventory_Vel_Days','Value_of_Output_to_Total_Asset
```

In [5]:
```
#Checking Shape
df.shape
```

Out[5]:
```
(3586, 67)
```

In [6]:
```
#Checking Size
df.size
```

Out[6]:
```
240262
```

In [7]:
```
#Checking the duplicates

dups = df.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))
```

```
Number of duplicate rows = 0
```

In [8]:
```
#Checking the basic info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3586 entries, 0 to 3585
Data columns (total 67 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Co_Code                         3586 non-null   int64
 1   Co_Name                         3586 non-null   object
 2   Networth_Next_Year              3586 non-null   float64
 3   Equity_Paid_Up                  3586 non-null   float64
 4   Networth                        3586 non-null   float64
 5   Capital_Employed                3586 non-null   float64
 6   Total_Debt                      3586 non-null   float64
 7   Gross_Block                     3586 non-null   float64
 8   Net_Working_Capital             3586 non-null   float64
 9   Curr_Assets                     3586 non-null   float64
 10  Curr_Liab_and_Prov              3586 non-null   float64
 11  Total_Assets_to_Liab            3586 non-null   float64
 12  Gross_Sales                     3586 non-null   float64
 13  Net_Sales                       3586 non-null   float64
 14  Other_Income                    3586 non-null   float64
 15  Value_Of_Output                 3586 non-null   float64
 16  Cost_of_Prod                    3586 non-null   float64
 17  Selling_Cost                    3586 non-null   float64
 18  PBIDT                           3586 non-null   float64
 19  PBDT                            3586 non-null   float64
 20  PBIT                            3586 non-null   float64
 21  PBT                             3586 non-null   float64
 22  PAT                             3586 non-null   float64
 23  Adjusted_PAT                    3586 non-null   float64
 24  CP                              3586 non-null   float64
 25  Rev_earn_in_forex               3586 non-null   float64
 26  Rev_exp_in_forex                3586 non-null   float64
 27  Capital_exp_in_forex            3586 non-null   float64
 28  Book_Value_Unit_Curr            3586 non-null   float64
 29  Book_Value_Adj_Unit_Curr        3582 non-null   float64
 30  Market_Capitalisation           3586 non-null   float64
 31  CEPS_annualised_Unit_Curr       3586 non-null   float64
 32  Cash_Flow_From_Opr              3586 non-null   float64
 33  Cash_Flow_From_Inv              3586 non-null   float64
 34  Cash_Flow_From_Fin              3586 non-null   float64
 35  ROG_Net_Worth_perc              3586 non-null   float64
 36  ROG_Capital_Employed_perc       3586 non-null   float64
 37  ROG_Gross_Block_perc            3586 non-null   float64
 38  ROG_Gross_Sales_perc            3586 non-null   float64
 39  ROG_Net_Sales_perc              3586 non-null   float64
```

```
 40  OG_Cost_of_Prod_perc               3586 non-null   float64
 41  ROG_Total_Assets_perc              3586 non-null   float64
 42  OG_PBIDT_perc                      3586 non-null   float64
 43  ROG_PBDT_perc                      3586 non-null   float64
 44  ROG_PBIT_perc                      3586 non-null   float64
 45  ROG_PBT_perc                       3586 non-null   float64
 46  ROG_PAT_perc                       3586 non-null   float64
 47  ROG_CP_perc                        3586 non-null   float64
 48  ROG_Rev_earn_in_forex_perc         3586 non-null   float64
 49  ROG_Rev_exp_in_forex_perc          3586 non-null   float64
 50  ROG_Market_Capitalisation_perc     3586 non-null   float64
 51  Curr_Ratio_Latest                  3585 non-null   float64
 52  Fixed_Assets_Ratio_Latest          3585 non-null   float64
 53  Inventory_Ratio_Latest             3585 non-null   float64
 54  Debtors_Ratio_Latest               3585 non-null   float64
 55  Total_Asset_Turnover_Ratio_Latest  3585 non-null   float64
 56  Interest_Cover_Ratio_Latest        3585 non-null   float64
 57  PBIDTM_perc_Latest                 3585 non-null   float64
 58  PBITM_perc_Latest                  3585 non-null   float64
 59  PBDTM_perc_Latest                  3585 non-null   float64
 60  CPM_perc_Latest                    3585 non-null   float64
 61  APATM_perc_Latest                  3585 non-null   float64
 62  Debtors_Vel_Days                   3586 non-null   int64
 63  Creditors_Vel_Days                 3586 non-null   int64
 64  Inventory_Vel_Days                 3483 non-null   float64
 65  Value_of_Output_to_Total_Assets    3586 non-null   float64
 66  Value_of_Output_to_Gross_Block     3586 non-null   float64
dtypes: float64(63), int64(3), object(1)
memory usage: 1.8+ MB
```

In [9]:

```
#Dropping the company code column (Co_Code) because that is not very useful for our ana

df.drop(['Co_Code','Co_Name'], axis = 1, inplace = True)

#Checking the df.describe

pd.set_option('display.float_format', lambda x: '%.2f' % x) #Keeping the decimal points

df.describe()
```

Out[9]:

	Networth_Next_Year	Equity_Paid_Up	Networth	Capital_Employed	Total_Debt	Gross_Block	Net_
count	3586.00	3586.00	3586.00	3586.00	3586.00	3586.00	
mean	725.05	62.97	649.75	2799.61	1994.82	594.18	
std	4769.68	778.76	4091.99	26975.14	23652.84	4871.55	
min	-8021.60	0.00	-7027.48	-1824.75	-0.72	-41.19	
25%	3.98	3.75	3.89	7.60	0.03	0.57	
50%	19.02	8.29	18.58	39.09	7.49	15.87	
75%	123.80	19.52	117.30	226.60	72.35	131.90	
max	111729.10	42263.46	81657.35	714001.25	652823.81	128477.59	

8 rows × 65 columns

Dependent variable - We need to create a default variable that should take the value of 1 when net worth next year is negative & 0 when net worth next year is positive.

In [10]:
```
#Creating a 'Default' Variable with the below conditions

#Creating a list of our conditions:

conditions = [(df['Networth_Next_Year'] < 0), (df['Networth_Next_Year'] > 0)]

#Creating a list of the values that we have to assign for each condition
values = ['1', '0']

#Now, Creating a new column named 'Default'
#Using np.select to assign values to this new variable using the lists (conditions and

df['Default'] = np.select(conditions, values)
```

In [81]:
```
#Let's look at the head after created the default variable.

df[['Networth_Next_Year','Default']].head()
```

Out[81]:

	Networth_Next_Year	Default
0	-8021.60	1
1	-3986.19	1
2	-3192.58	1
3	-3054.51	1
4	-2967.36	1

In [82]:
```
#Let's look at the tail after created the default variable.

df[['Networth_Next_Year','Default']].tail()
```

Out[82]:

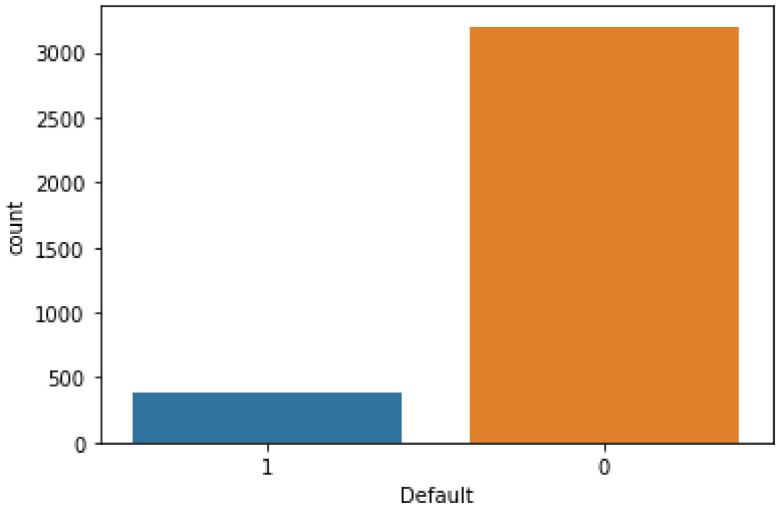
	Networth_Next_Year	Default
3581	72677.77	0
3582	79162.19	0
3583	88134.31	0
3584	91293.70	0
3585	111729.10	0

In [11]:
```
df['Default'].value_counts()
```

Out[11]:
```
0    3199
1     387
```

```
Name: Default, dtype: int64
```

In [12]:
```
ax = sns.countplot(x="Default", data=df)
```



In [13]:
```
df_x = df.drop('Default', axis =1)
df_y = df['Default']
```

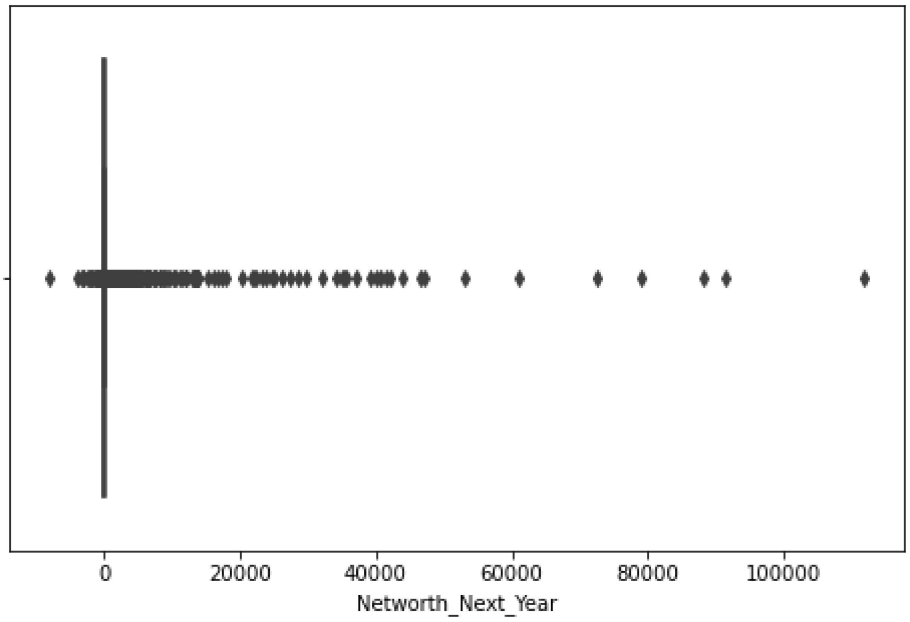
In [14]:
```
df_x.head()
```

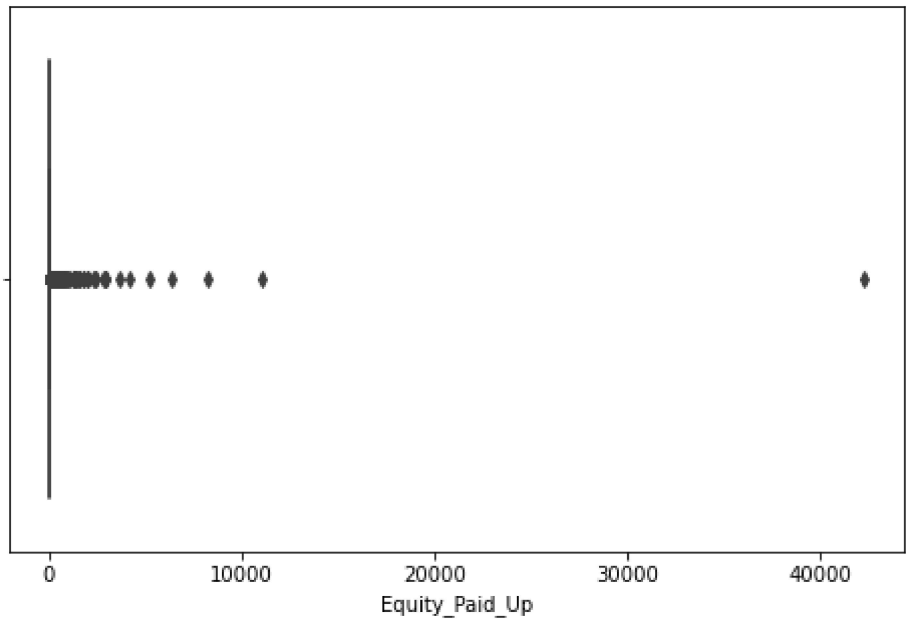
Out[14]:

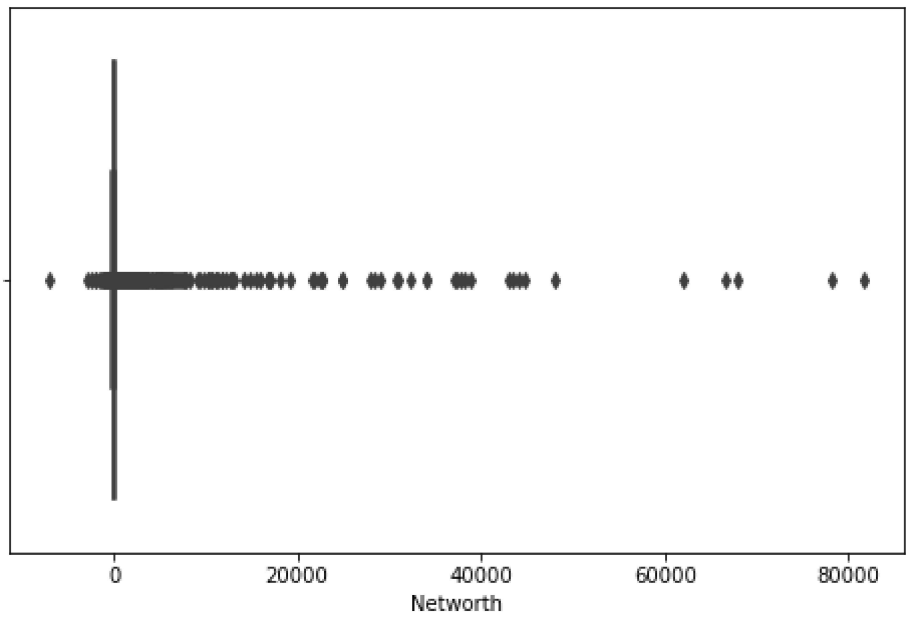
	Networth_Next_Year	Equity_Paid_Up	Networth	Capital_Employed	Total_Debt	Gross_Block	Net_Wor
0	-8021.60	419.36	-7027.48	-1007.24	5936.03	474.30	
1	-3986.19	1954.93	-2968.08	4458.20	7410.18	9070.86	
2	-3192.58	53.84	506.86	7714.68	6944.54	1281.54	
3	-3054.51	157.30	-623.49	2353.88	2326.05	1033.69	
4	-2967.36	50.30	-1070.83	4675.33	5740.90	1084.20	

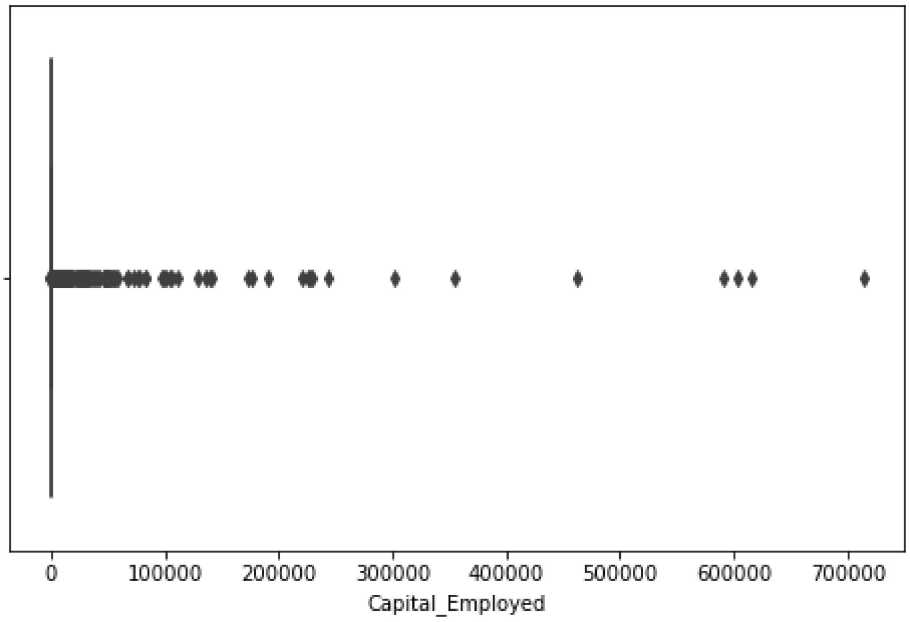
5 rows × 65 columns

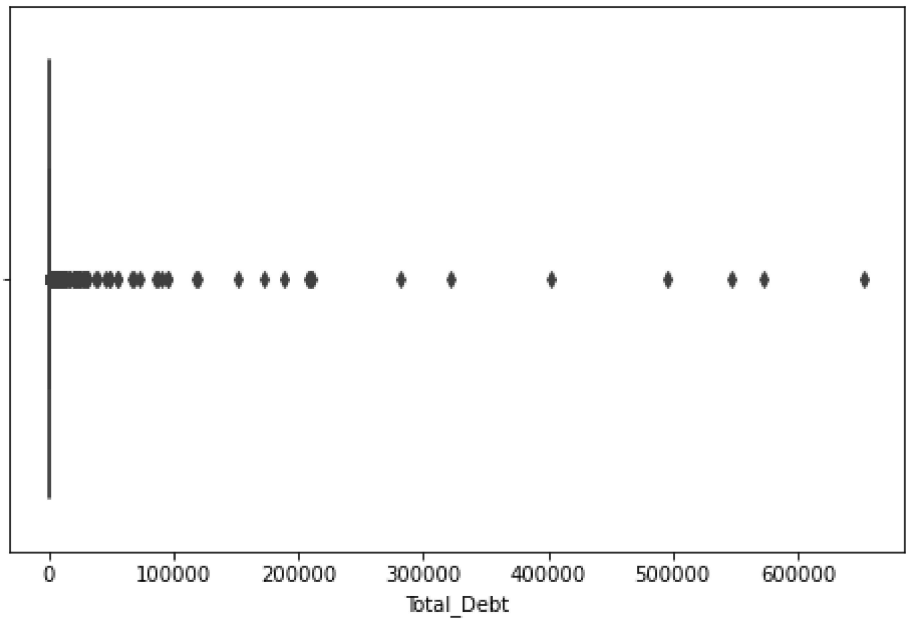
In [15]:
```
#Checking outliers

for x in df_x.columns:
    plt.figure(figsize=(8,5))
    sns.boxplot(df[x])
    plt.show()
```

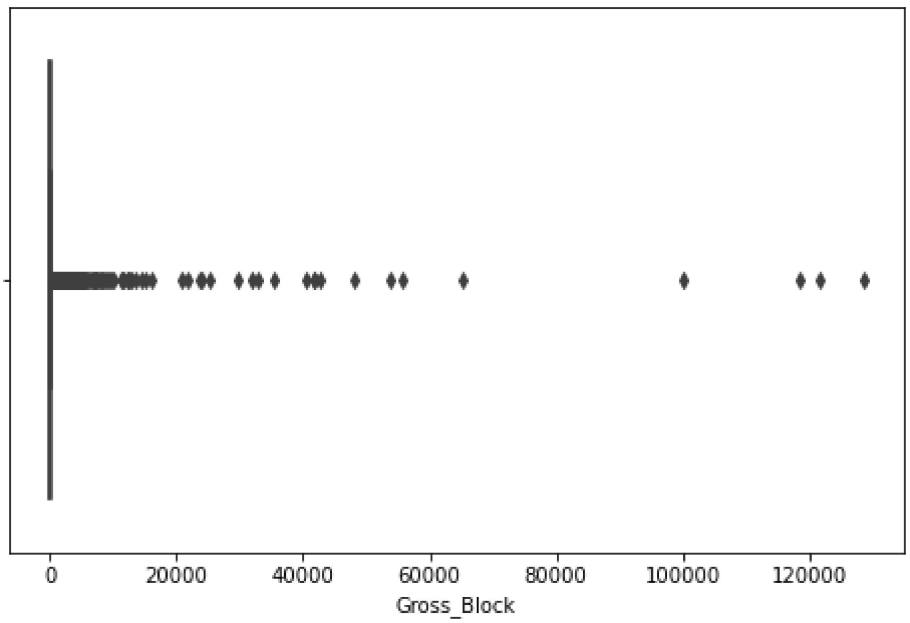

0
20000
40000
60000
80000
100000
Networth_Next_Year

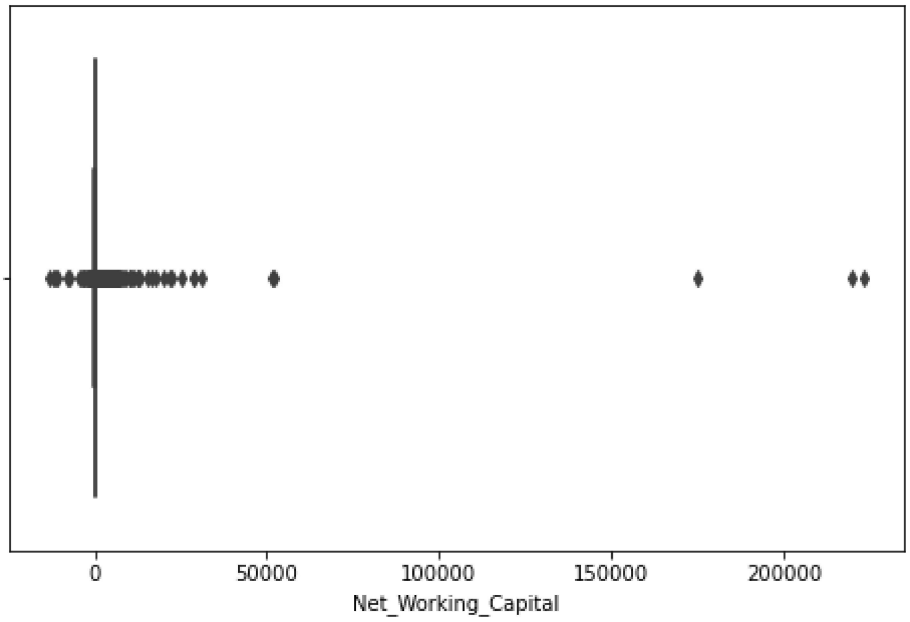

0
10000
20000
30000
40000
Equity_Paid_Up

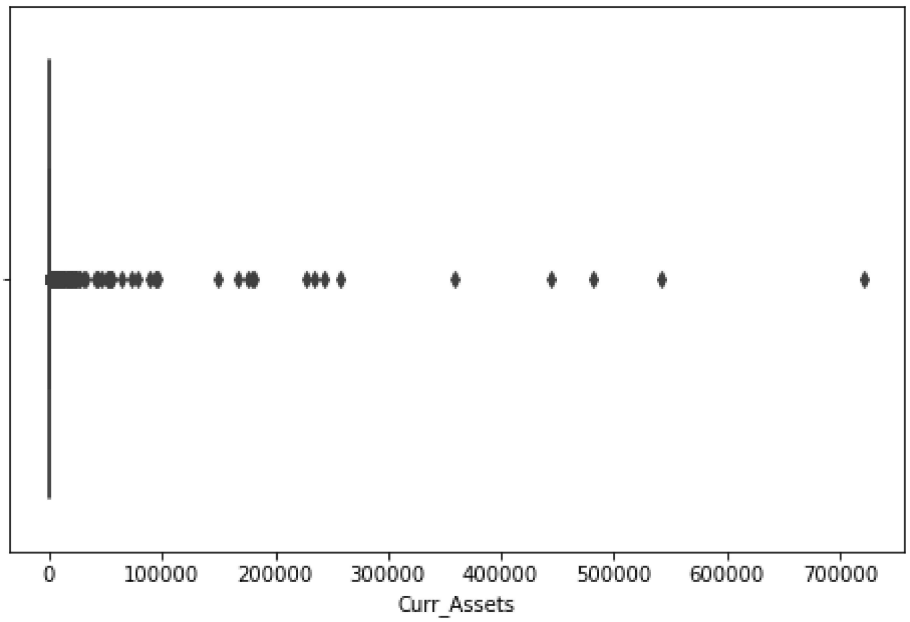

0
20000
40000
60000
80000
Networth

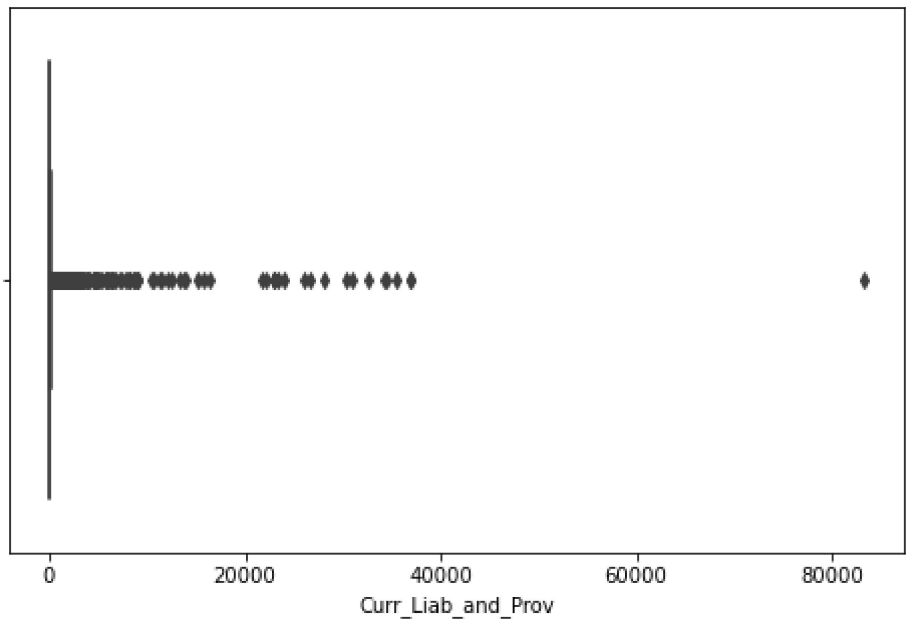

0
100000
200000
300000
400000
500000
600000
700000
Capital_Employed

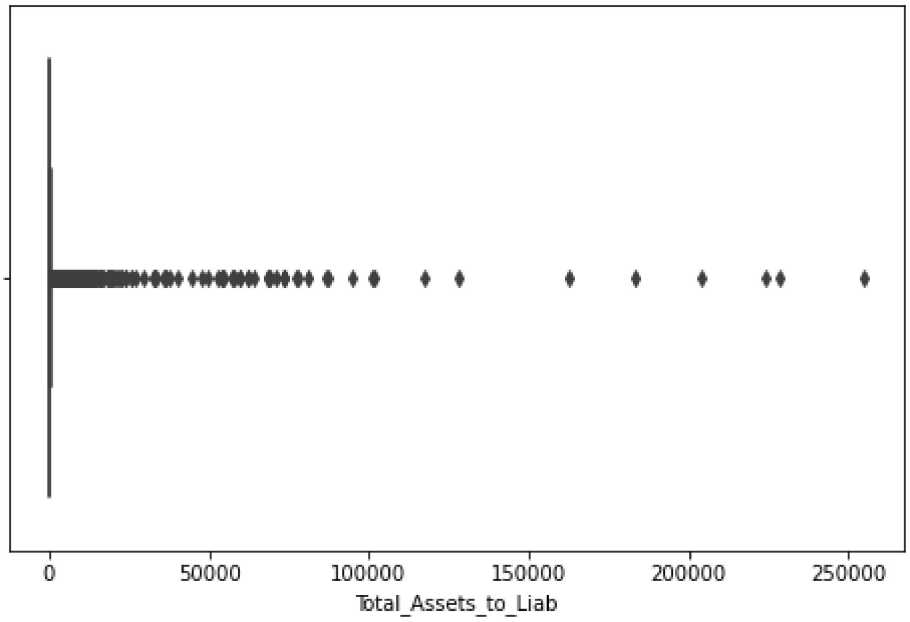

0
100000
200000
300000
400000
500000
600000
Total_Debt

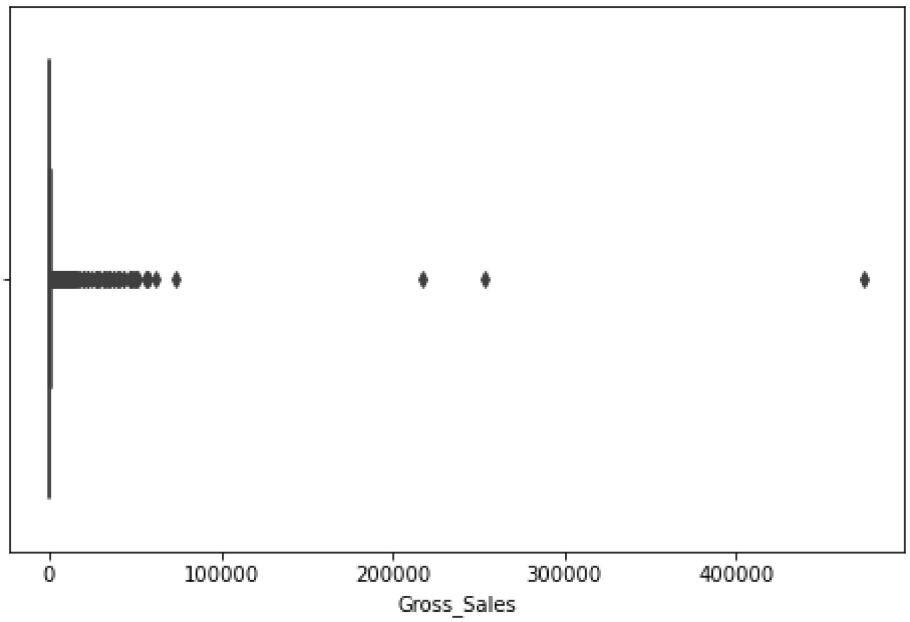

0
20000
40000
60000
80000
100000
120000
Gross_Block

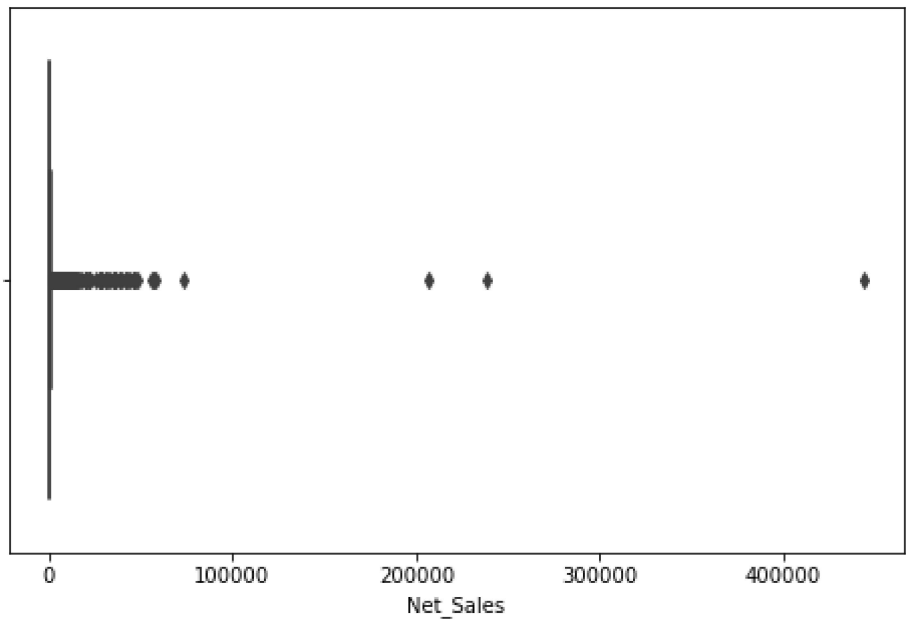

0
50000
100000
150000
200000
Net_Working_Capital

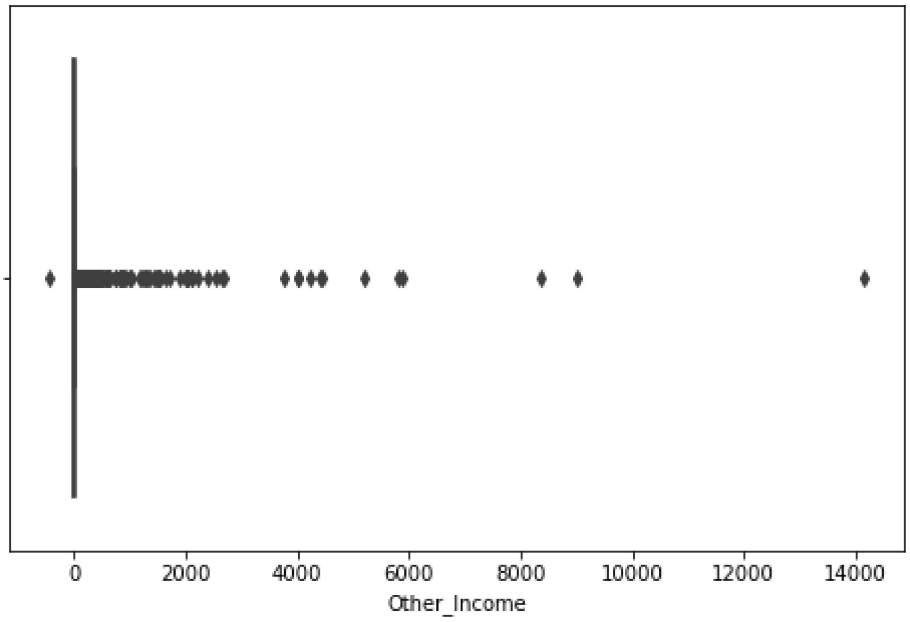

0
100000
200000
300000
400000
500000
600000
700000
Curr_Assets

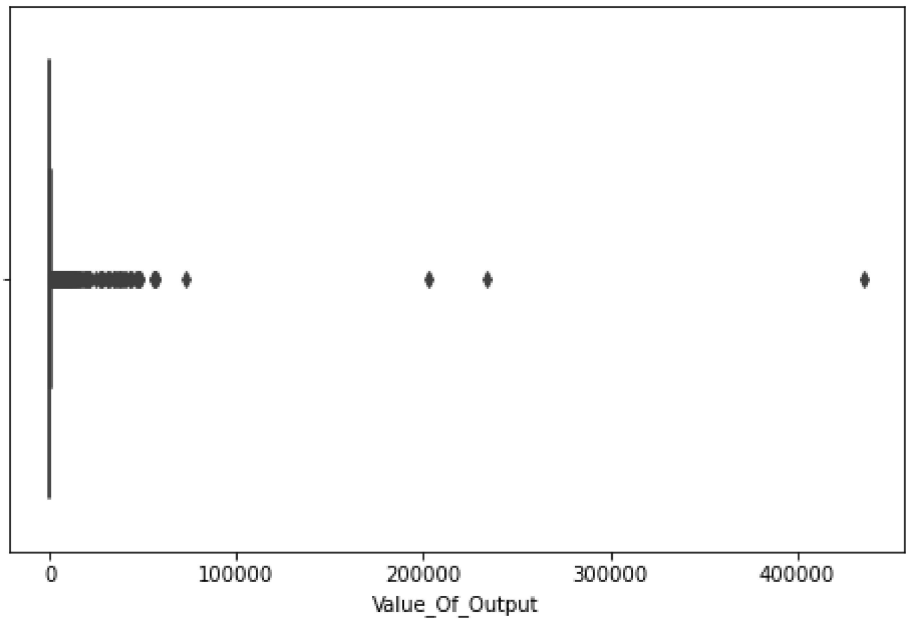

0
20000
40000
60000
80000
Curr_Liab_and_Prov

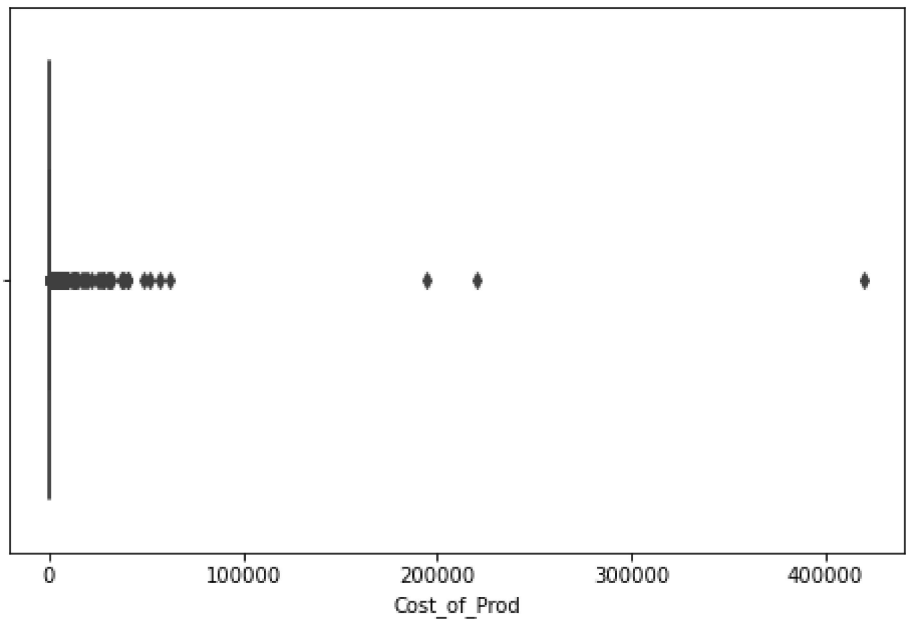

0
50000
100000
150000
200000
250000
Total_Assets_to_Liab

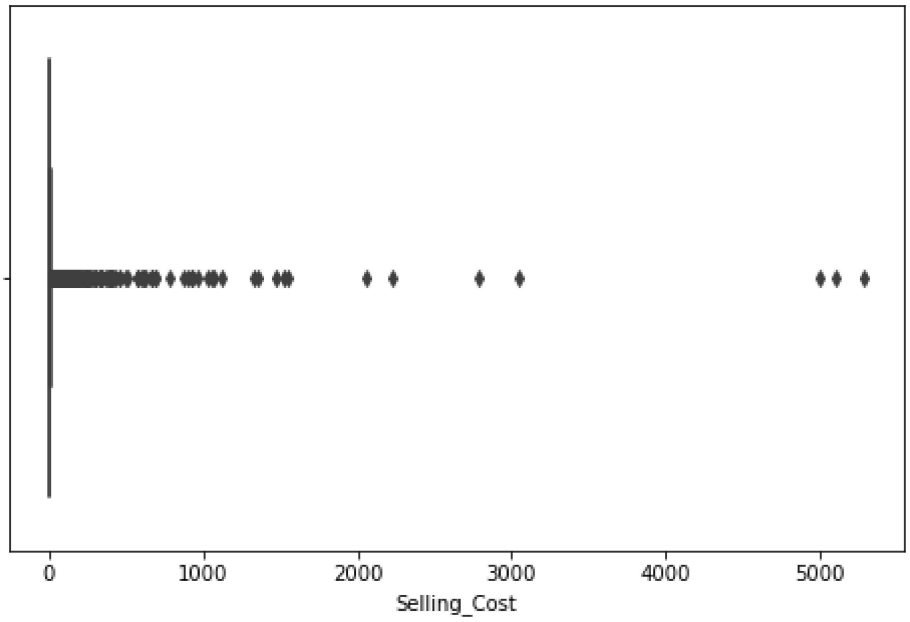

0
100000
200000
300000
400000
Gross_Sales

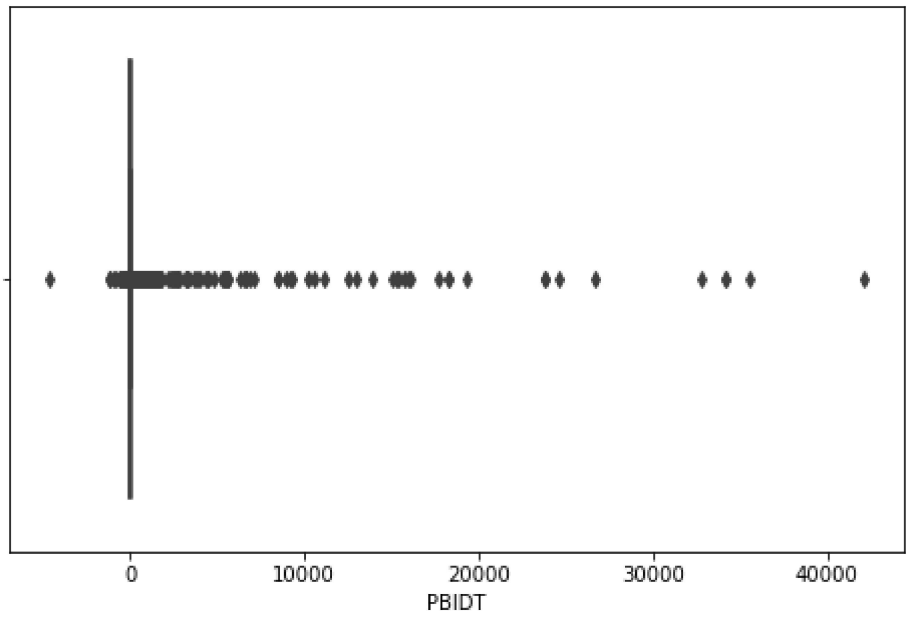

0
100000
200000
300000
400000
Net_Sales

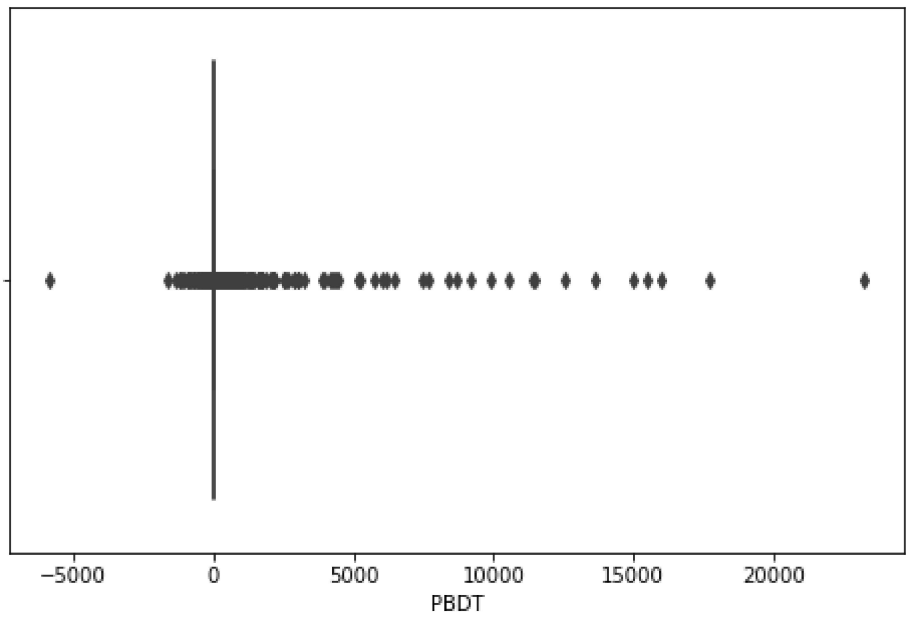

0
2000
4000
6000
8000
10000
12000
14000
Other_Income

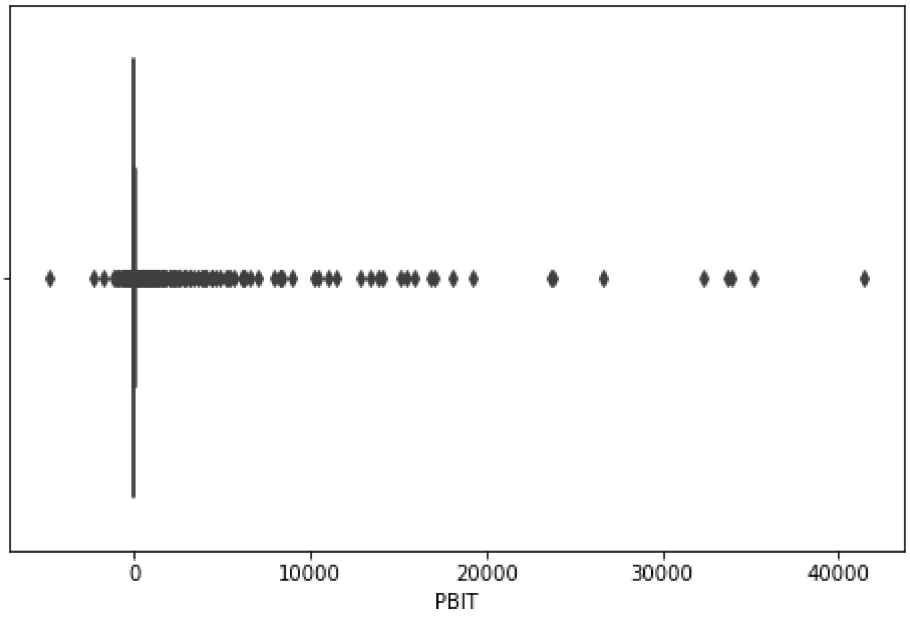

0
100000
200000
300000
400000
Value_Of_Output

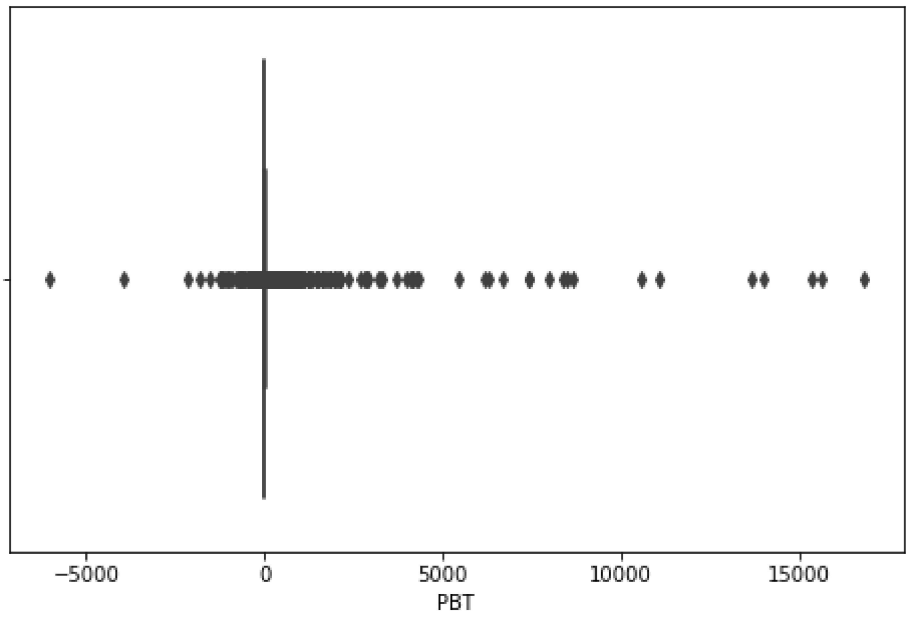

0
100000
200000
300000
400000
Cost_of_Prod

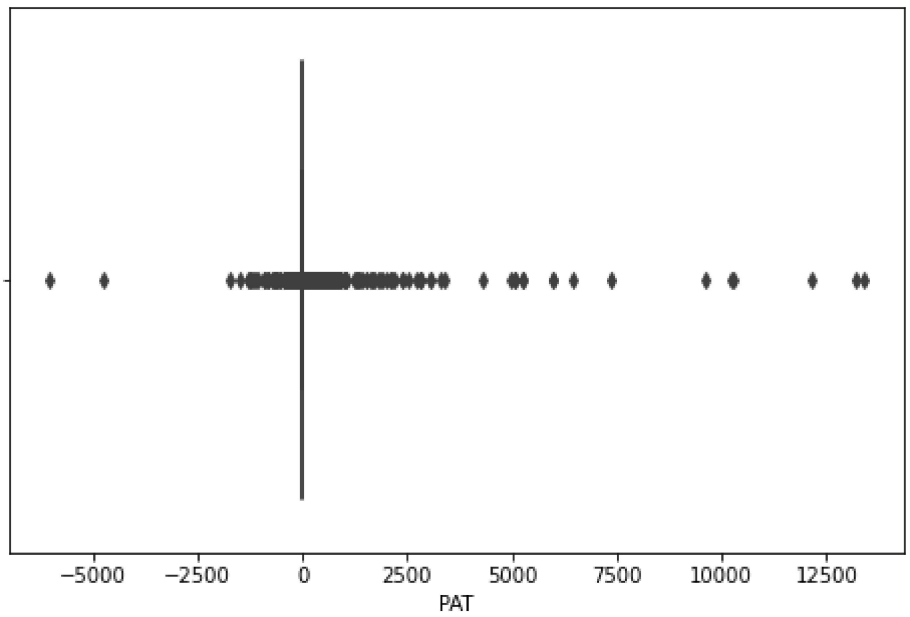

0
1000
2000
3000
4000
5000
Selling_Cost

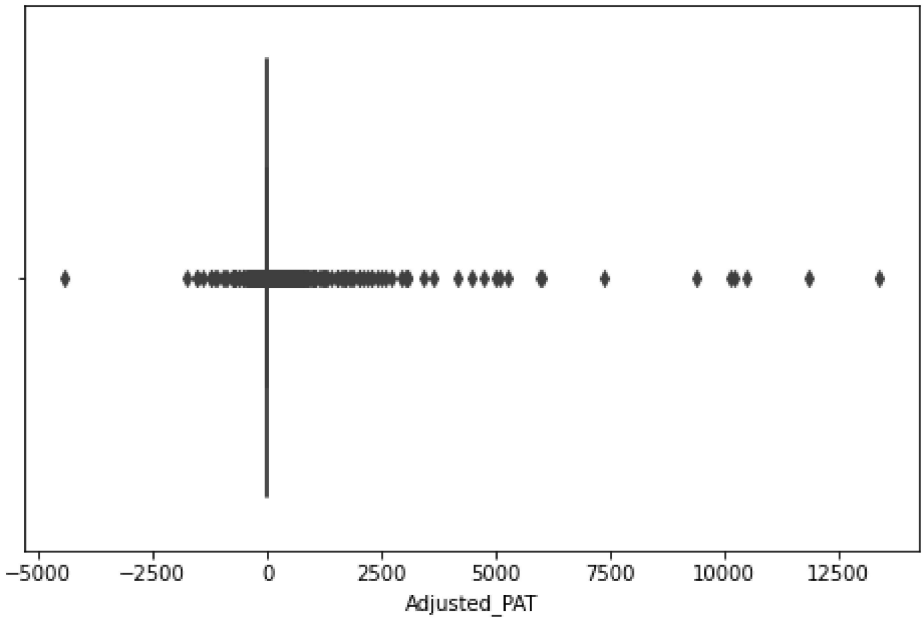

0
10000
20000
30000
40000
PBIDT

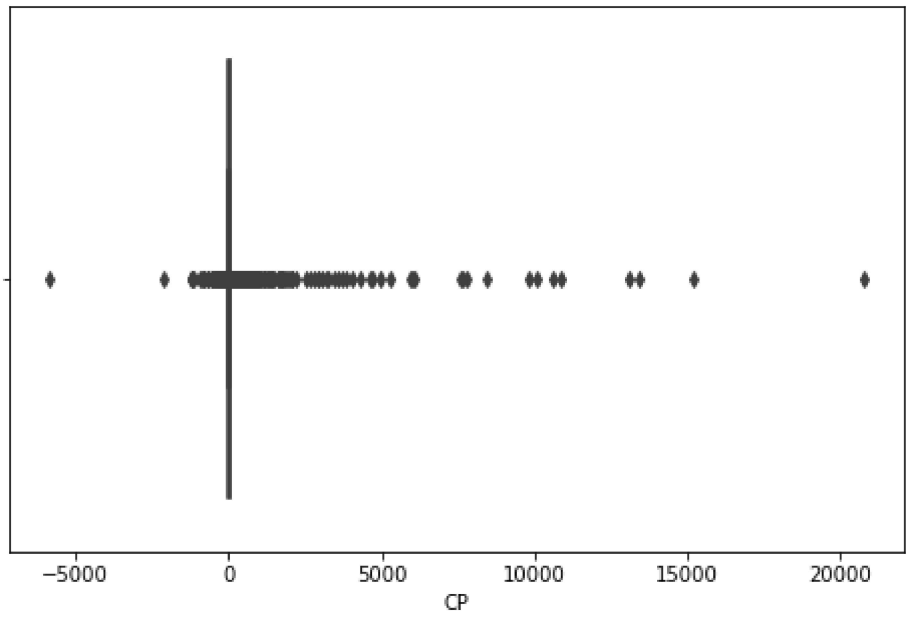

−5000
0
5000
10000
15000
20000
PBDT

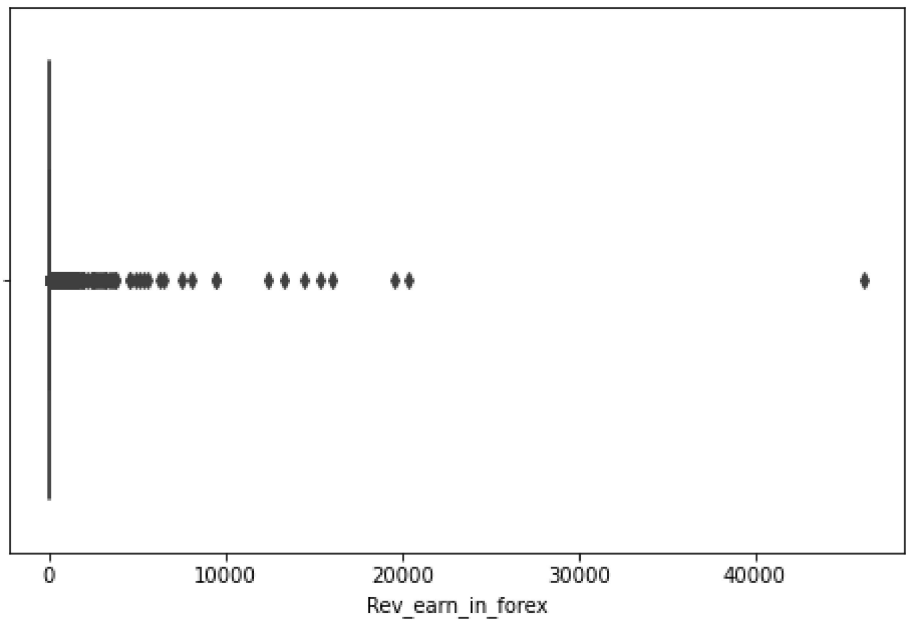

0
10000
20000
30000
40000
PBIT

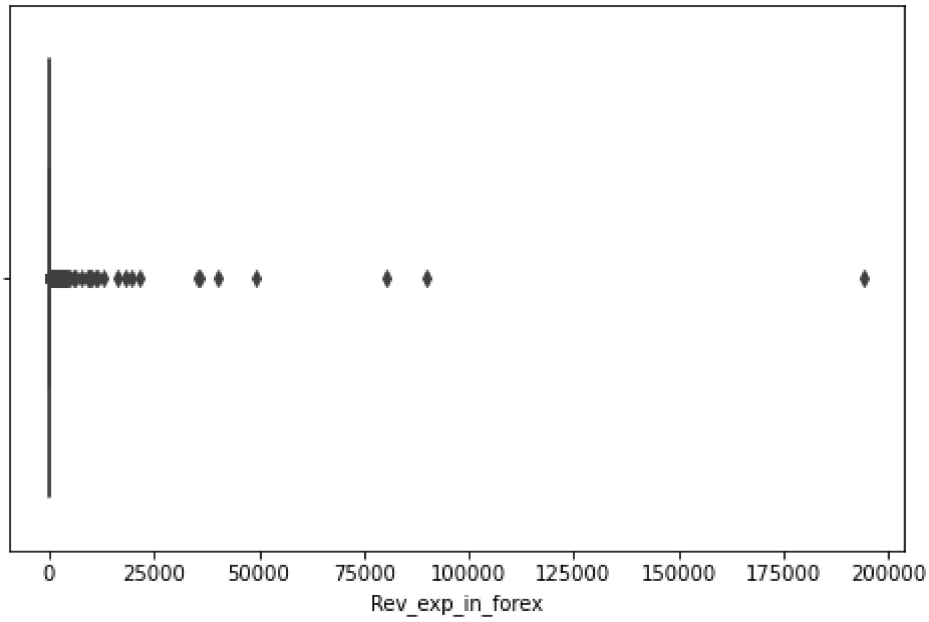

−5000
0
5000
10000
15000
PBT

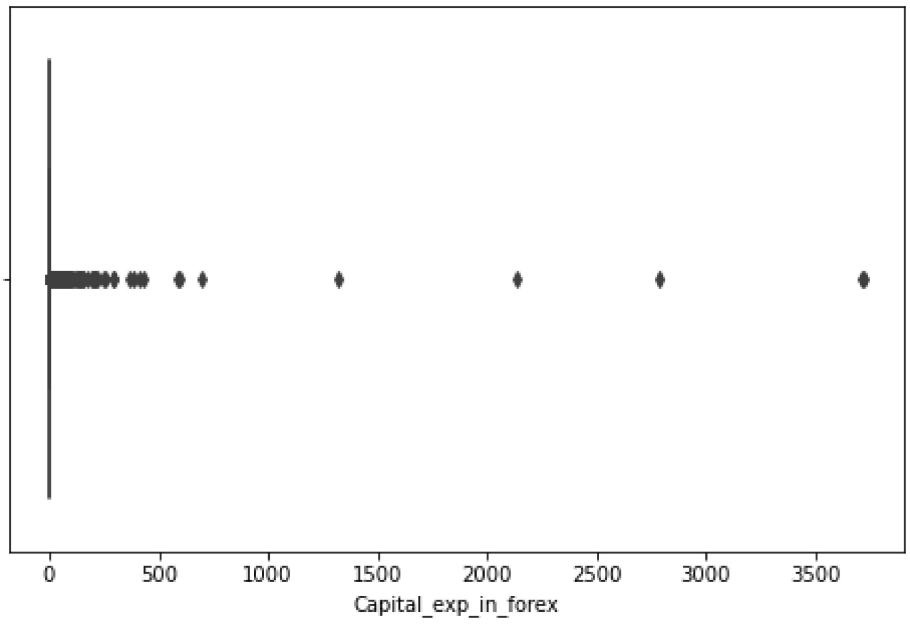

−5000
−2500
0
2500
5000
7500
10000
12500
PAT

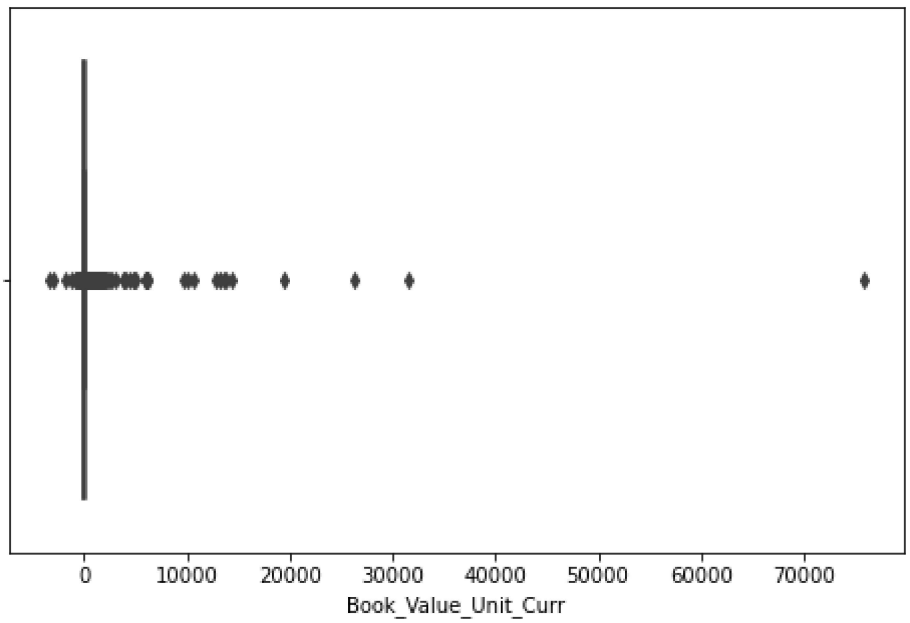

−5000
−2500
0
2500
5000
7500
10000
12500
Adjusted_PAT

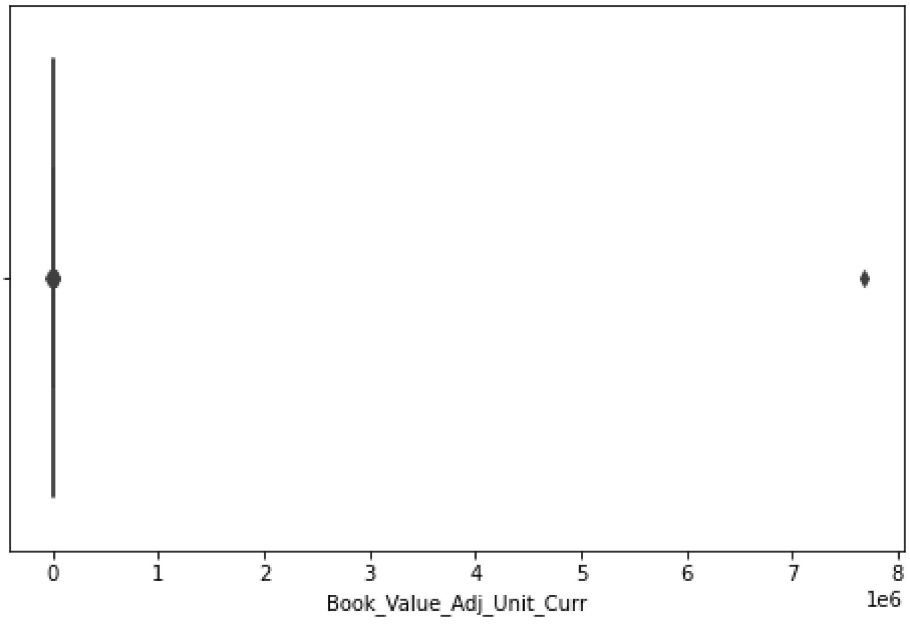

−5000
0
5000
10000
15000
20000
CP

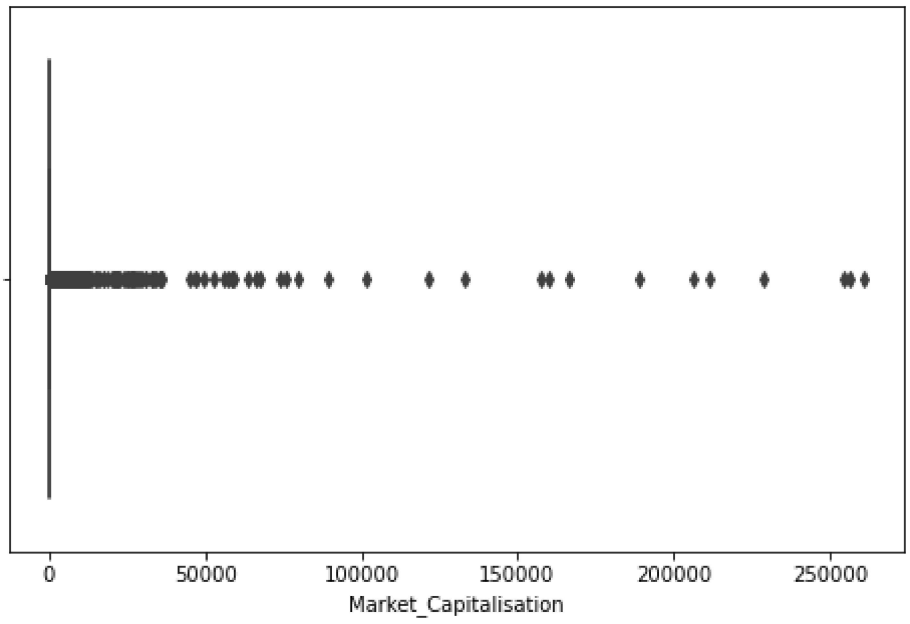

0
10000
20000
30000
40000
Rev_earn_in_forex

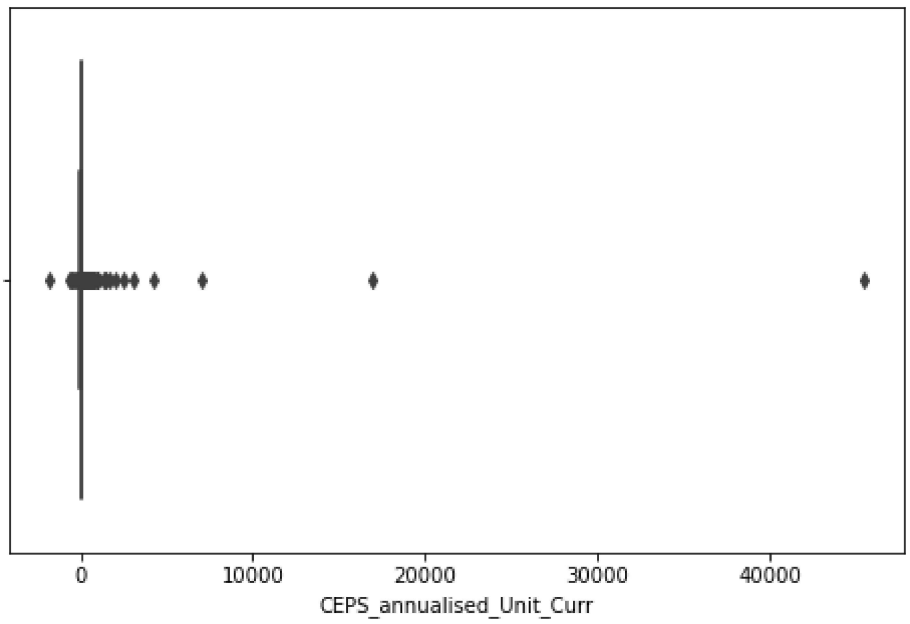

0
25000
50000
75000
100000
125000
150000
175000
200000
Rev_exp_in_forex

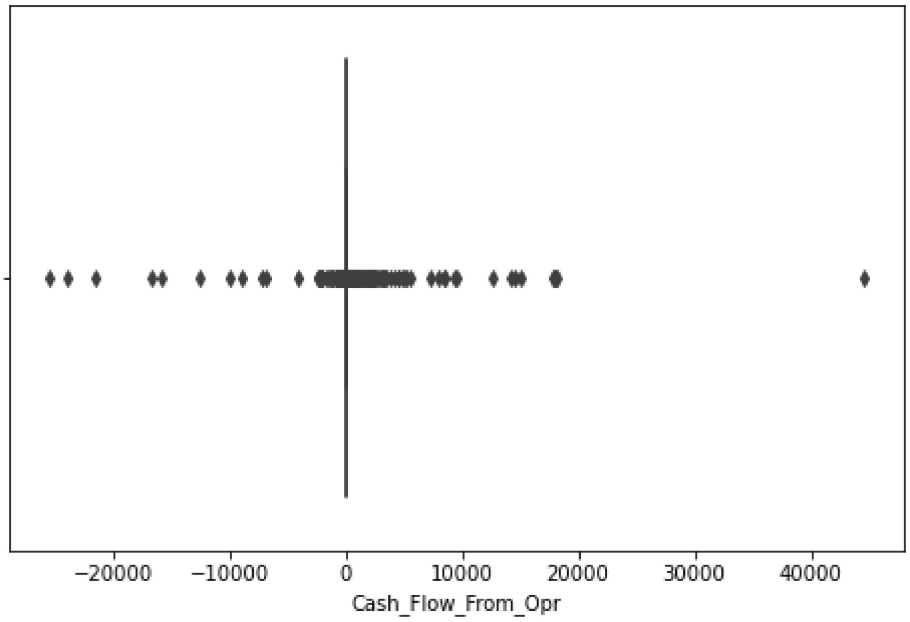

0
500
1000
1500
2000
2500
3000
3500
Capital_exp_in_forex

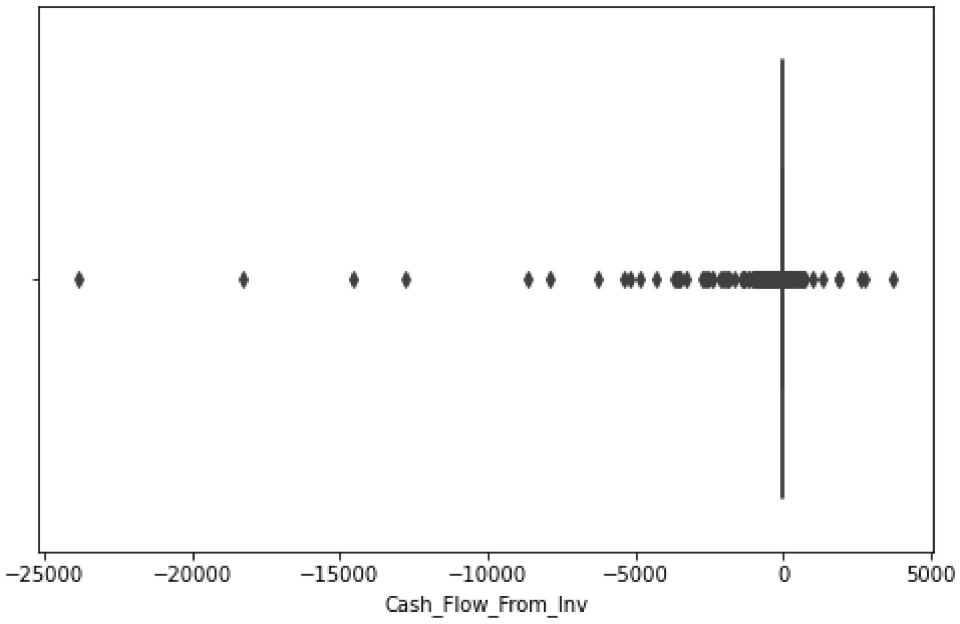

0
10000
20000
30000
40000
50000
60000
70000
Book_Value_Unit_Curr

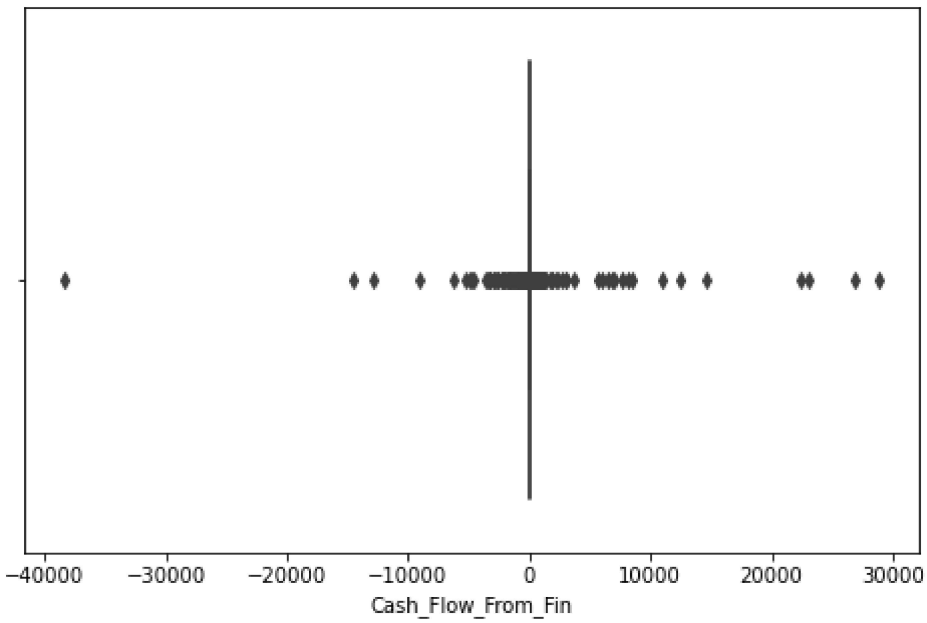

0
1
2
3
4
5
6
7
8
1e6
Book_Value_Adj_Unit_Curr

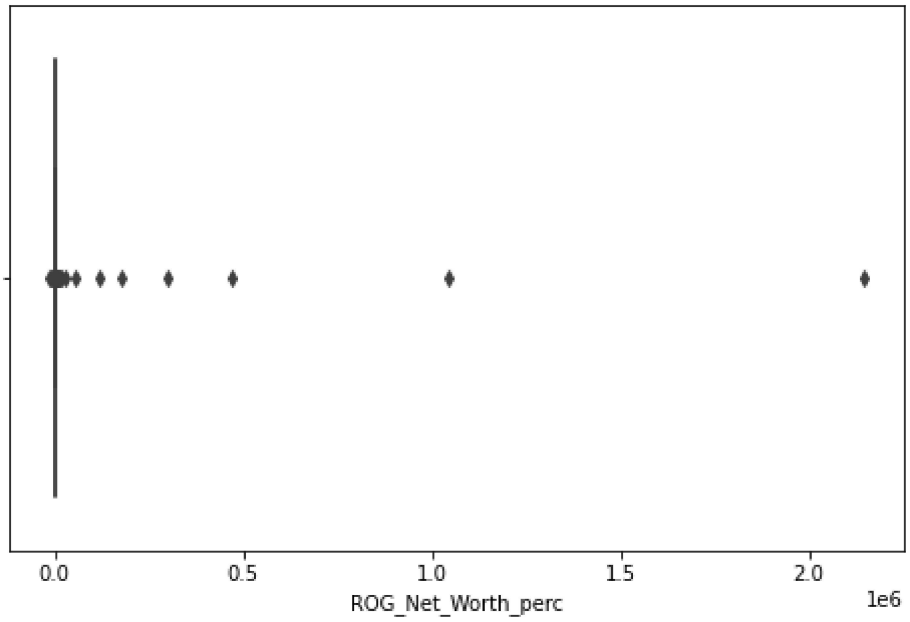

0
50000
100000
150000
200000
250000
Market_Capitalisation

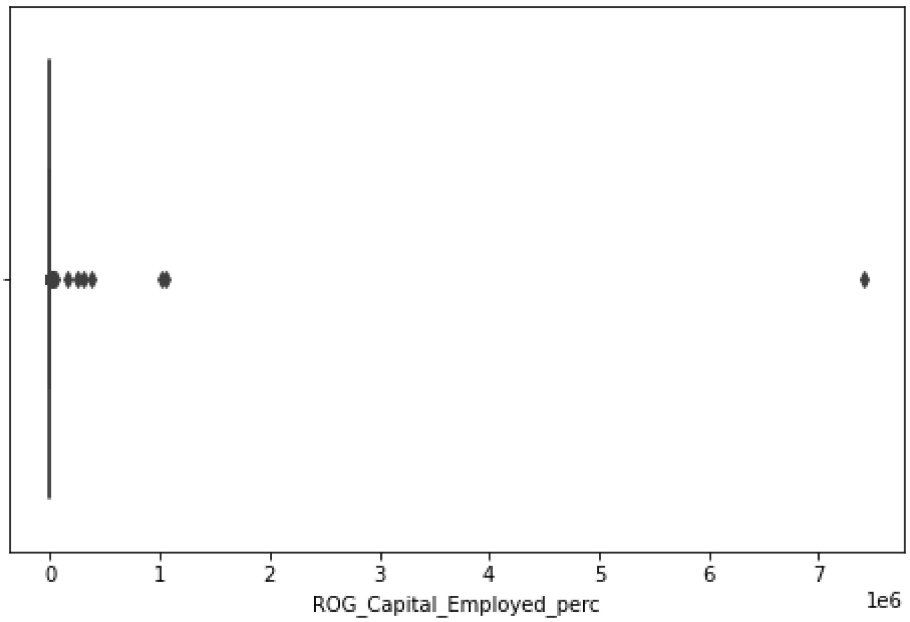

0
10000
20000
30000
40000
CEPS_annualised_Unit_Curr

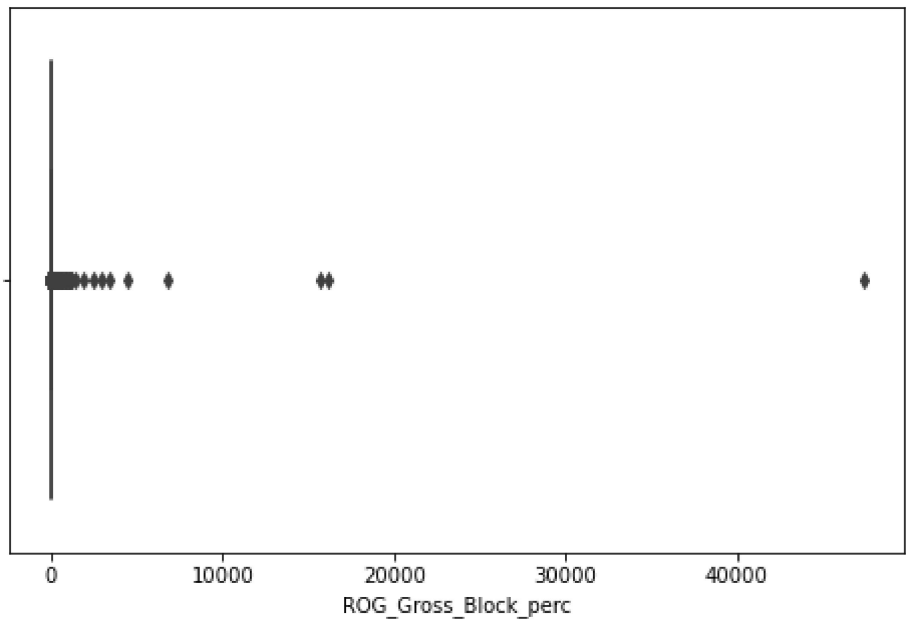

−20000
−10000
0
10000
20000
30000
40000
Cash_Flow_From_Opr

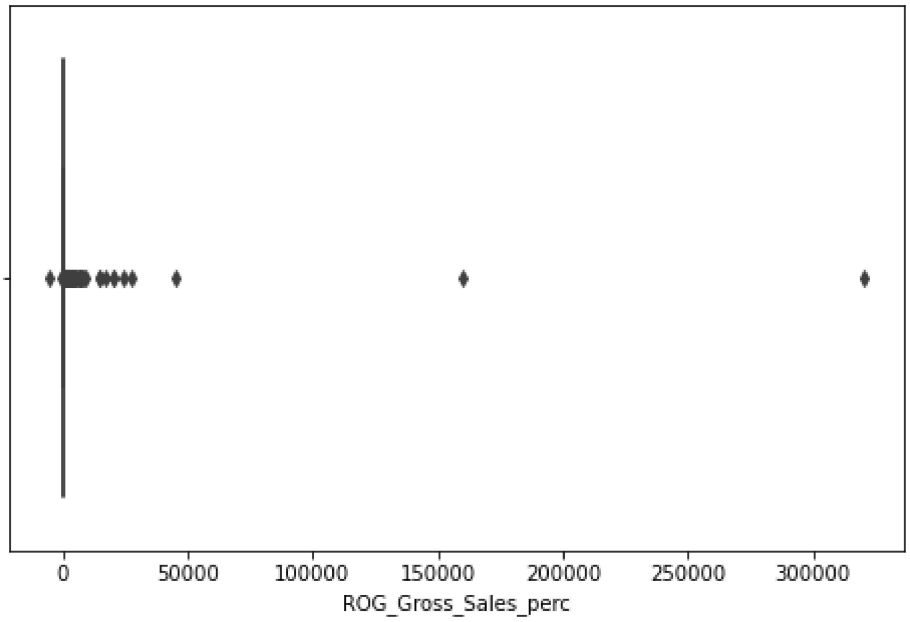

−25000
−20000
−15000
−10000
−5000
0
5000
Cash_Flow_From_Inv

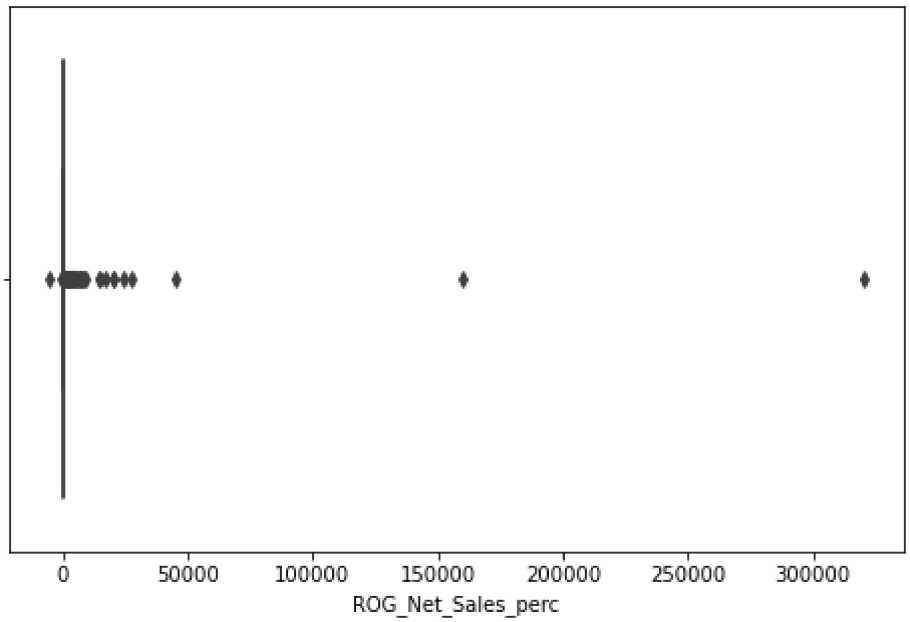

−40000
−30000
−20000
−10000
0
10000
20000
30000
Cash_Flow_From_Fin

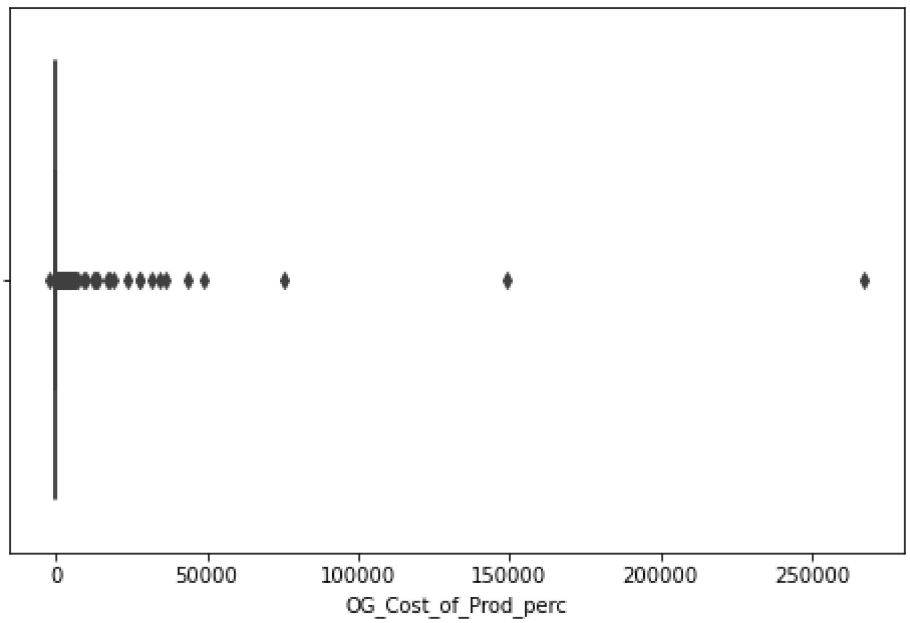

0.0
0.5
1.0
1.5
2.0
ROG_Net_Worth_perc
1e6

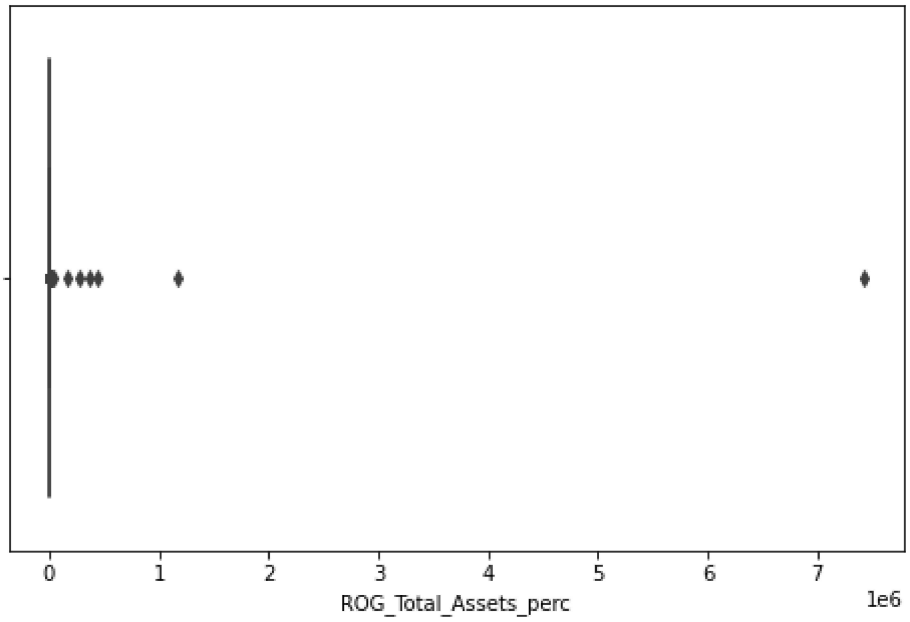

0
1
2
3
4
5
6
7
ROG_Capital_Employed_perc
1e6

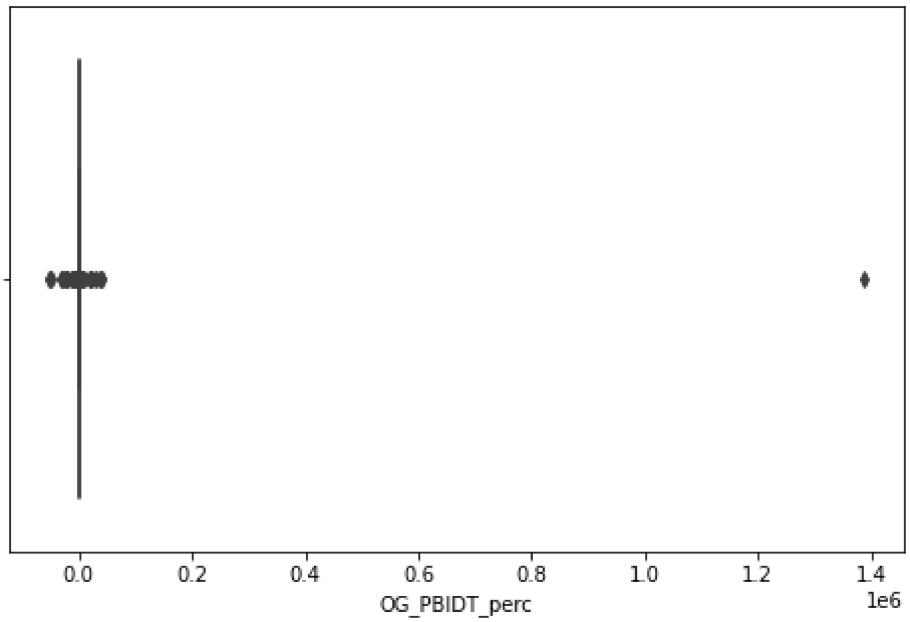

0
10000
20000
30000
40000
ROG_Gross_Block_perc

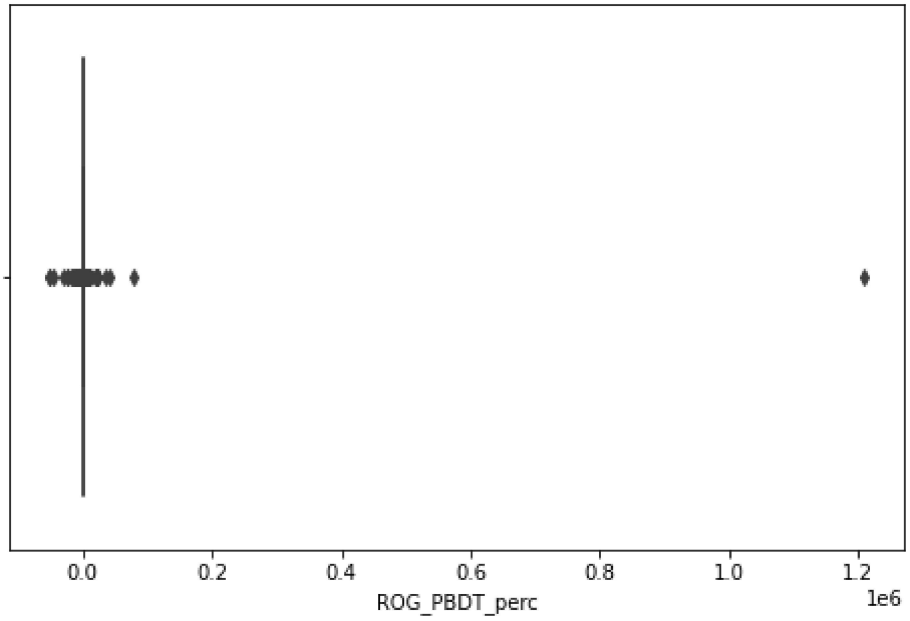

0
50000
100000
150000
200000
250000
300000
ROG_Gross_Sales_perc

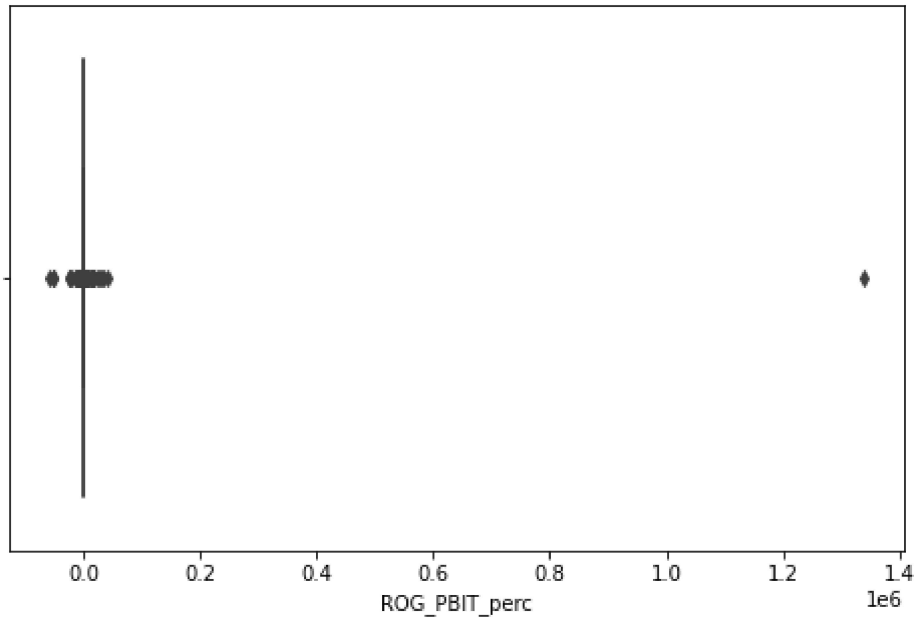

0
50000
100000
150000
200000
250000
300000
ROG_Net_Sales_perc

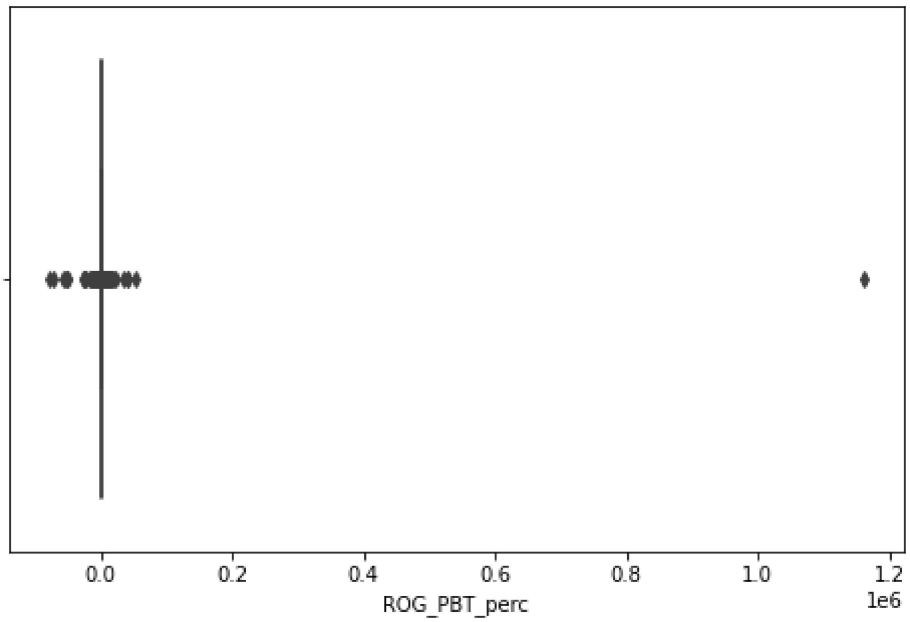

0
50000
100000
150000
200000
250000
OG_Cost_of_Prod_perc

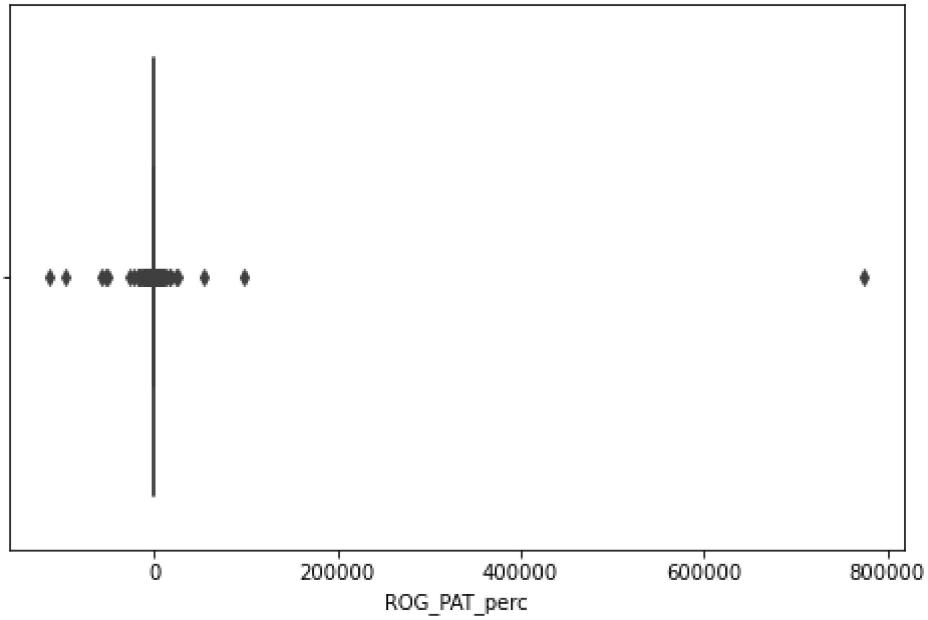

0
1
2
3
4
5
6
7
ROG_Total_Assets_perc
1e6

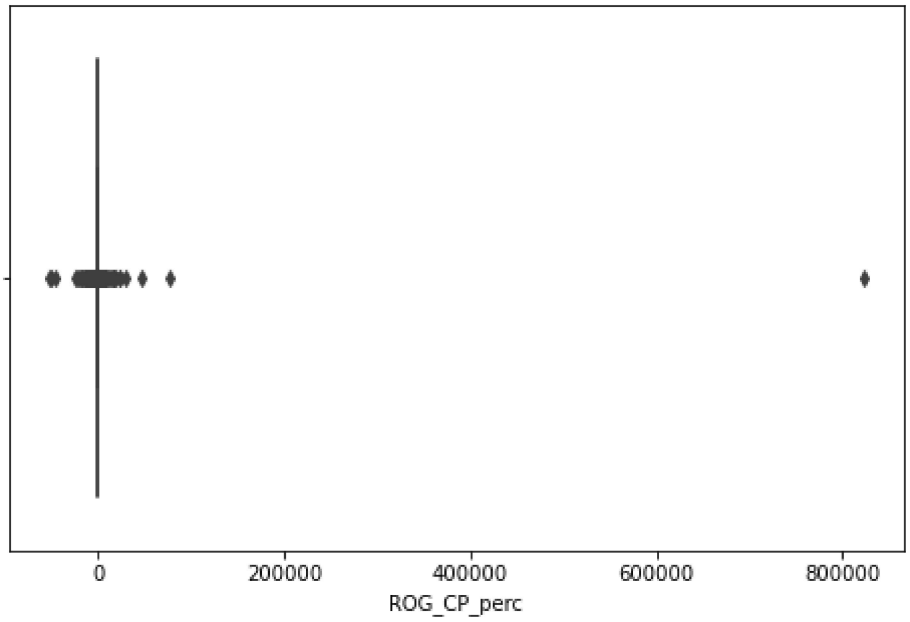

0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
OG_PBIDT_perc
1e6

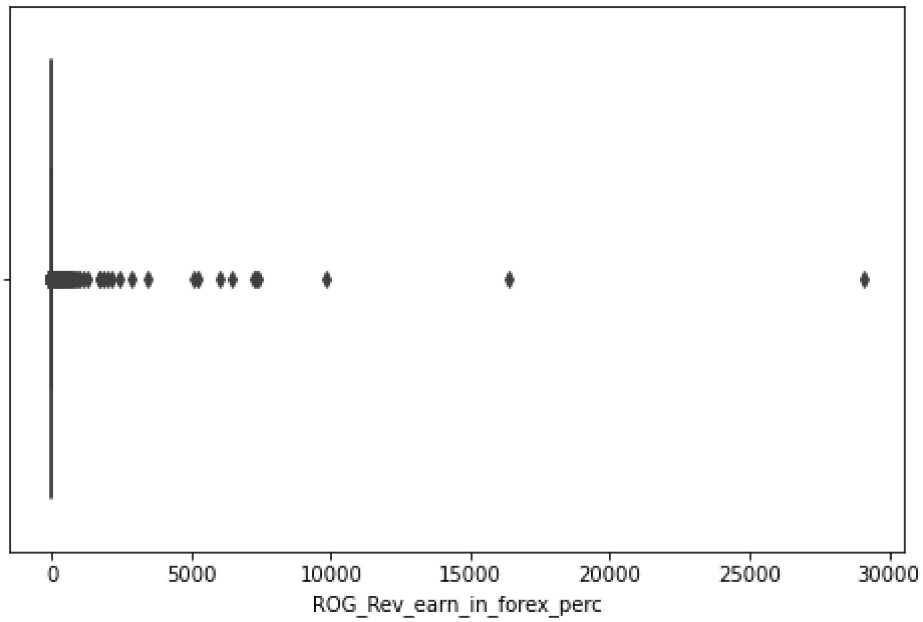

0.0
0.2
0.4
0.6
0.8
1.0
1.2
ROG_PBDT_perc
1e6

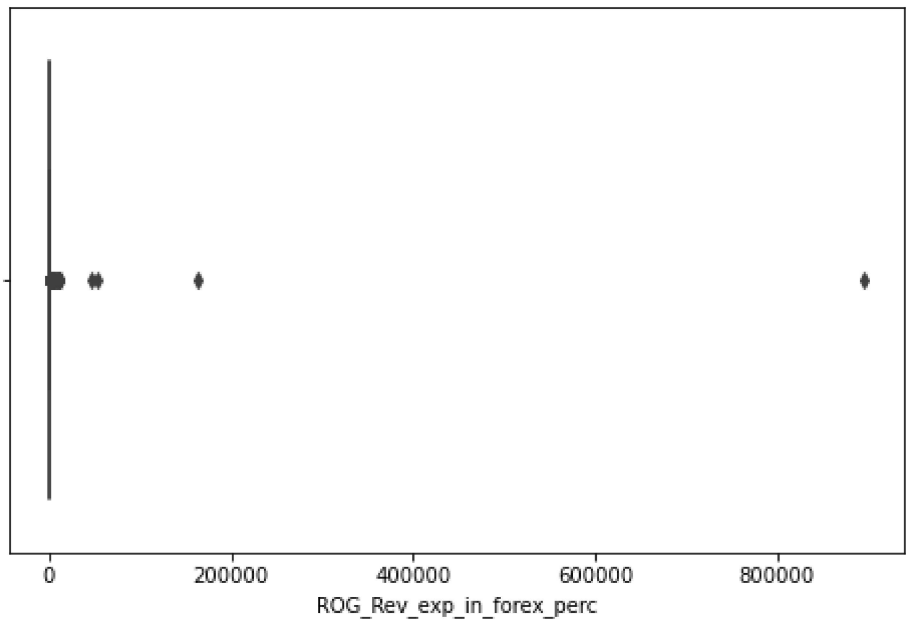

0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e6
ROG_PBIT_perc

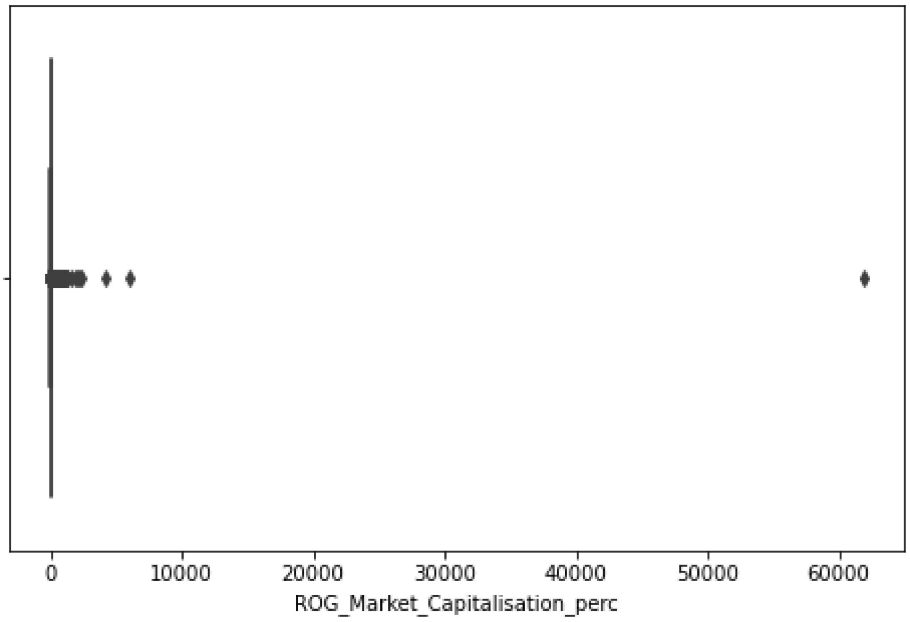

0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e6
ROG_PBT_perc

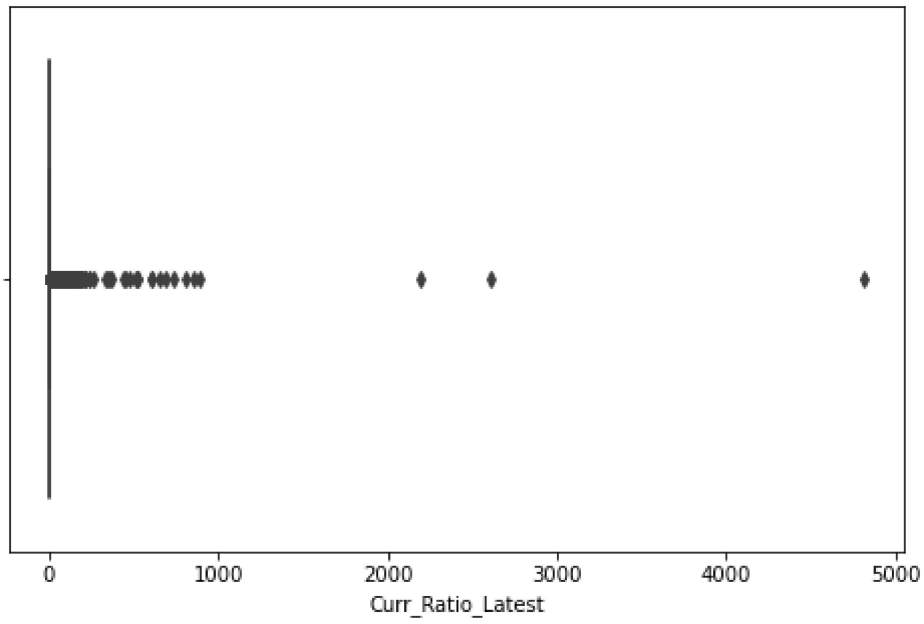

0
200000
400000
600000
800000
ROG_PAT_perc

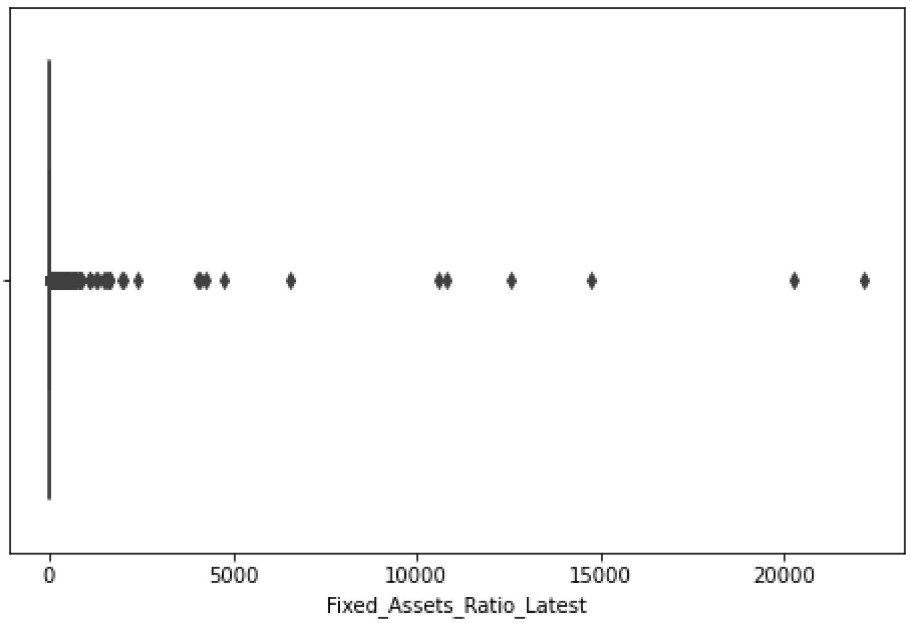

0
200000
400000
600000
800000
ROG_CP_perc

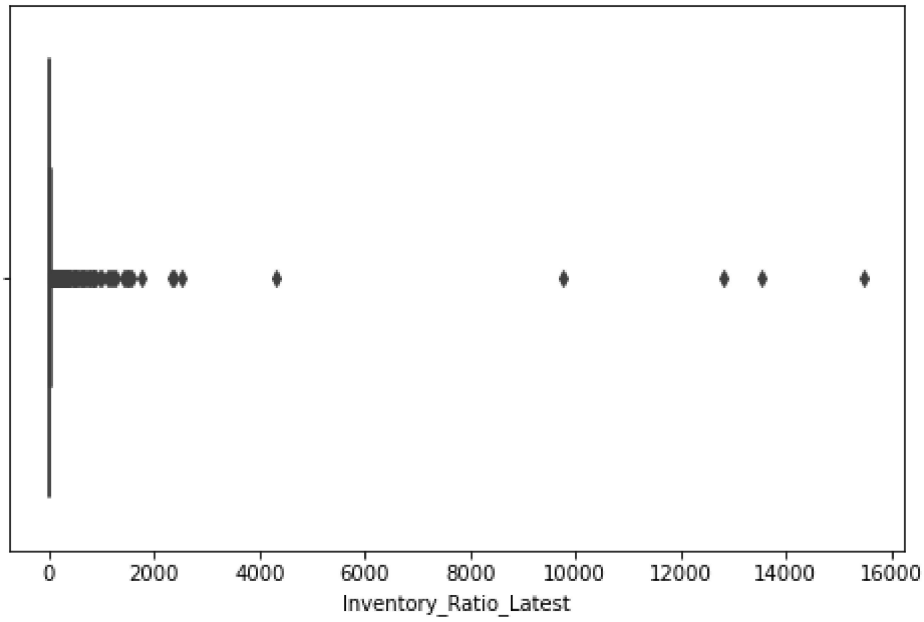

0
5000
10000
15000
20000
25000
30000
ROG_Rev_earn_in_forex_perc

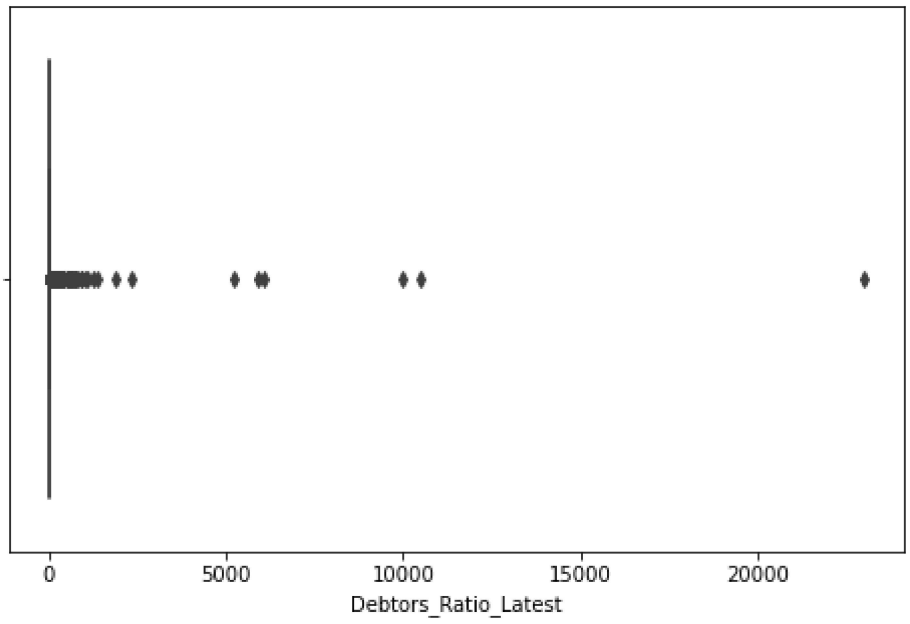

0
200000
400000
600000
800000
ROG_Rev_exp_in_forex_perc

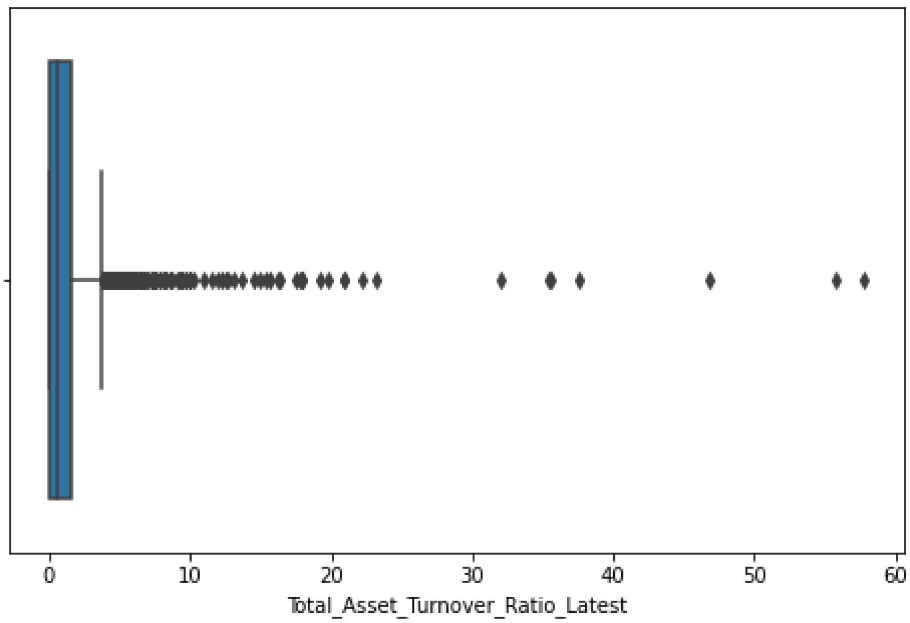

0
10000
20000
30000
40000
50000
60000
ROG_Market_Capitalisation_perc

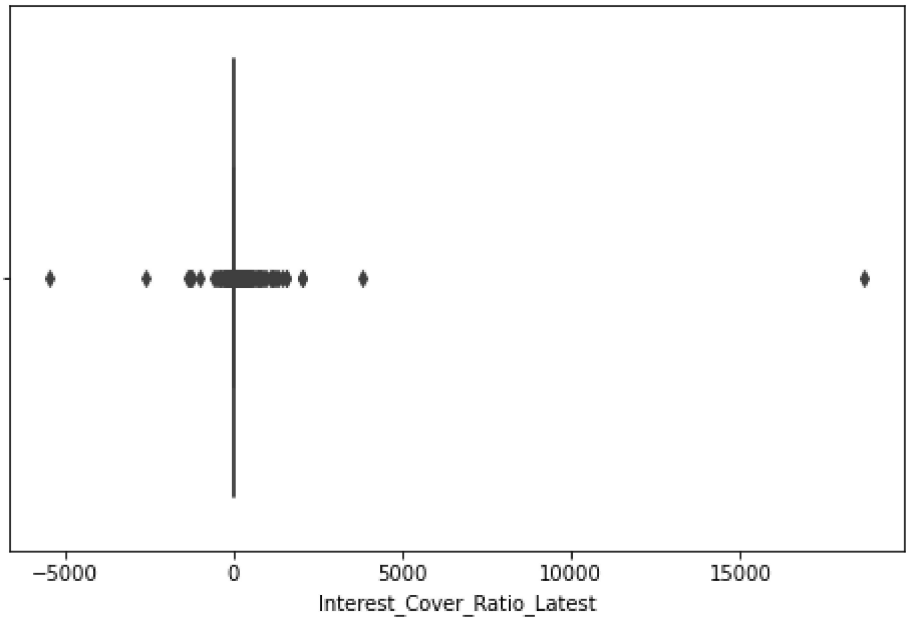

0
1000
2000
3000
4000
5000
Curr_Ratio_Latest

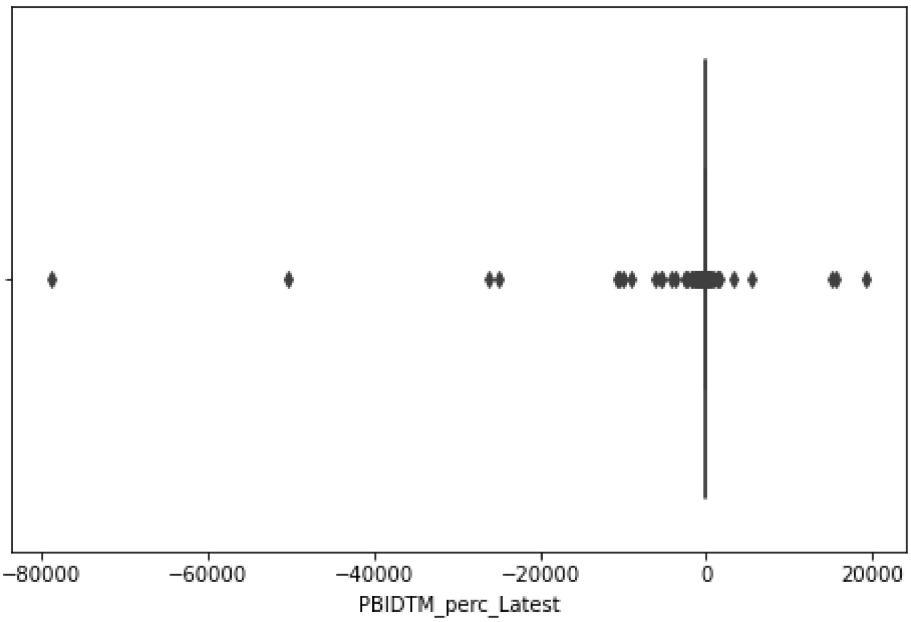

0
5000
10000
15000
20000
Fixed_Assets_Ratio_Latest

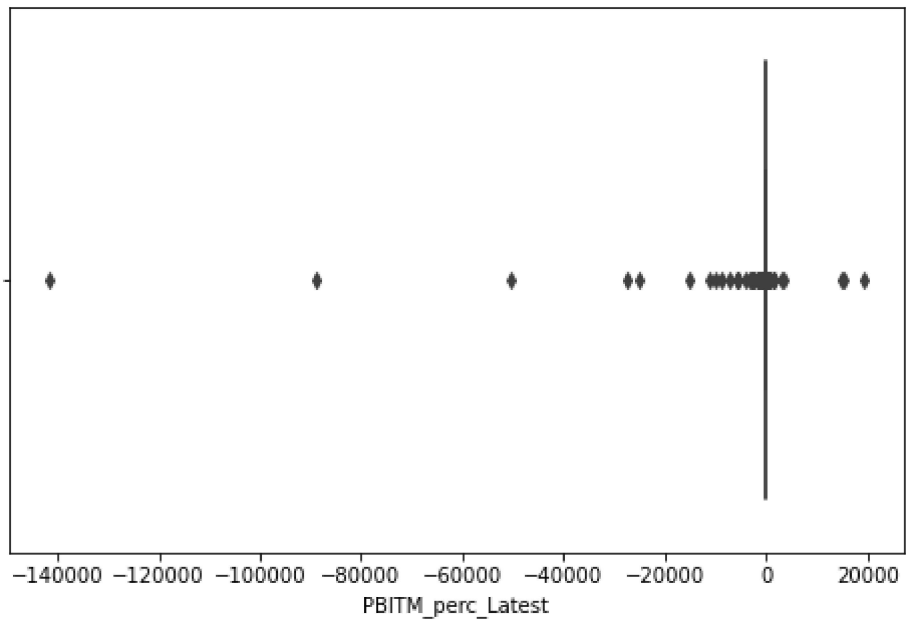

0
2000
4000
6000
8000
10000
12000
14000
16000
Inventory_Ratio_Latest

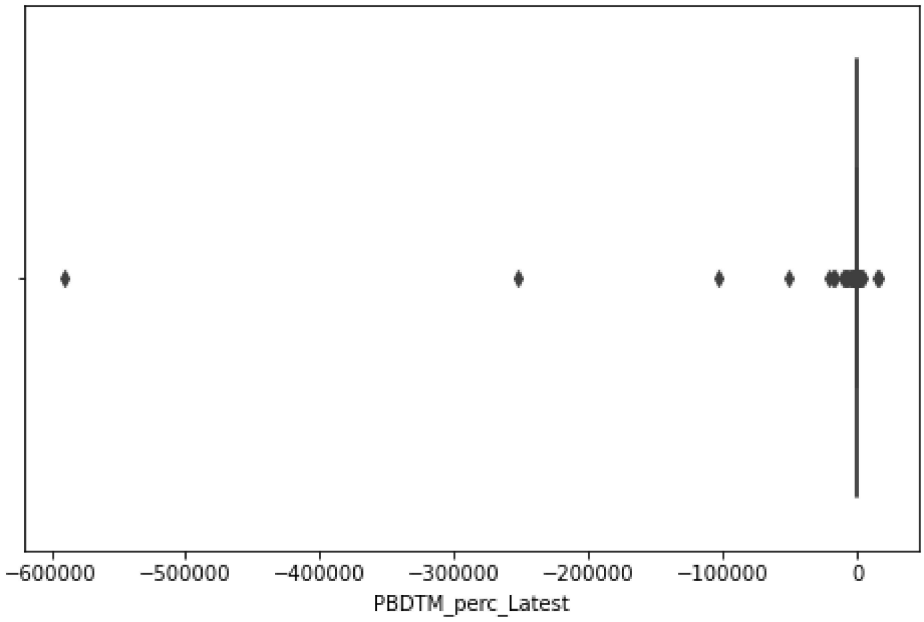

0
5000
10000
15000
20000
Debtors_Ratio_Latest

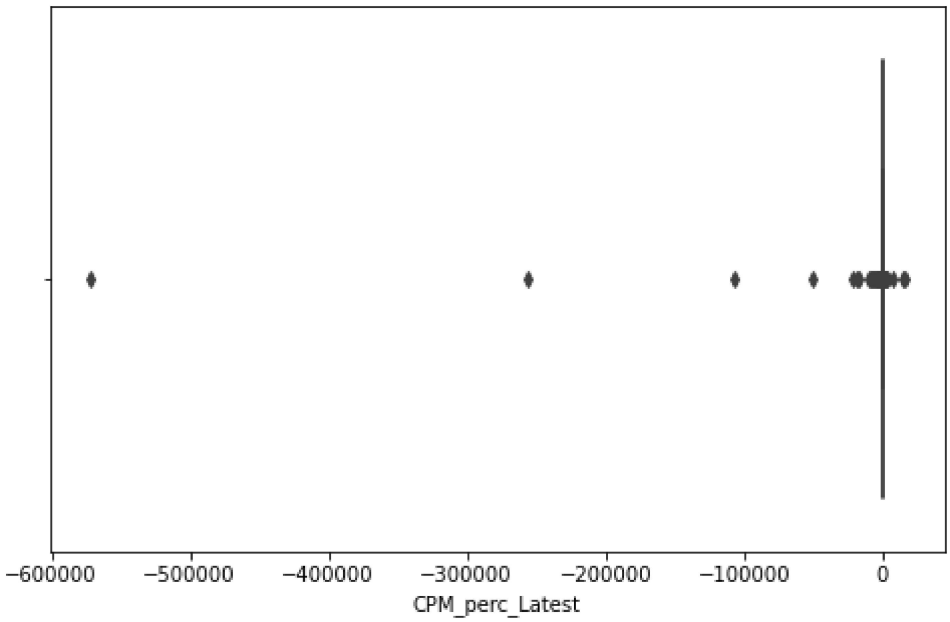

0
10
20
30
40
50
60
Total_Asset_Turnover_Ratio_Latest

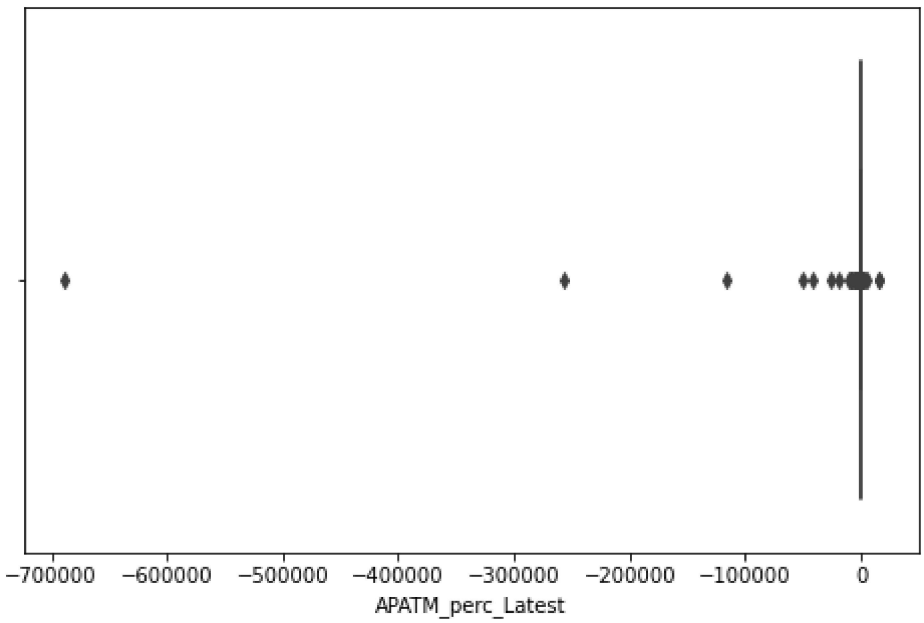

−5000
0
5000
10000
15000
Interest_Cover_Ratio_Latest

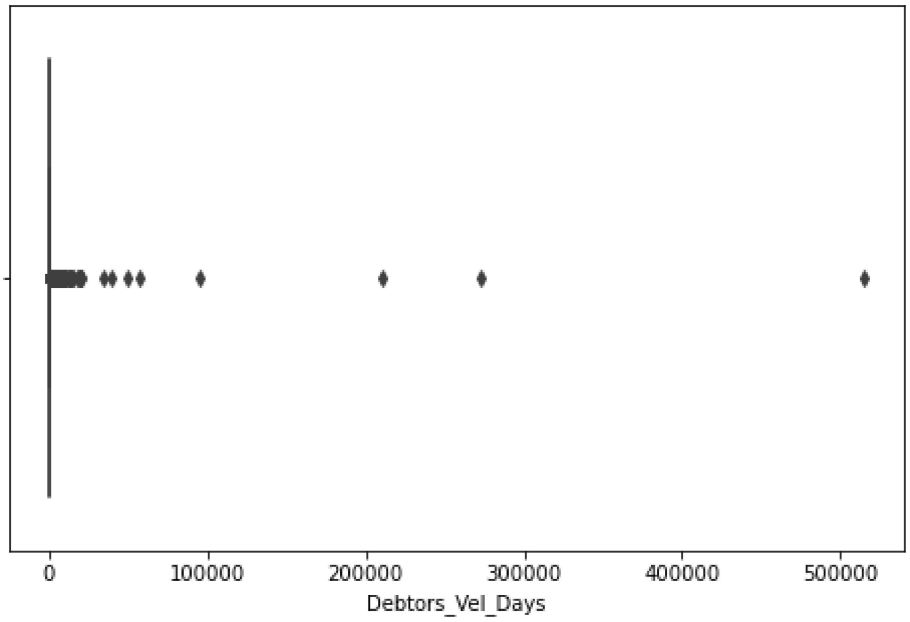

−80000
−60000
−40000
−20000
0
20000
PBIDTM_perc_Latest

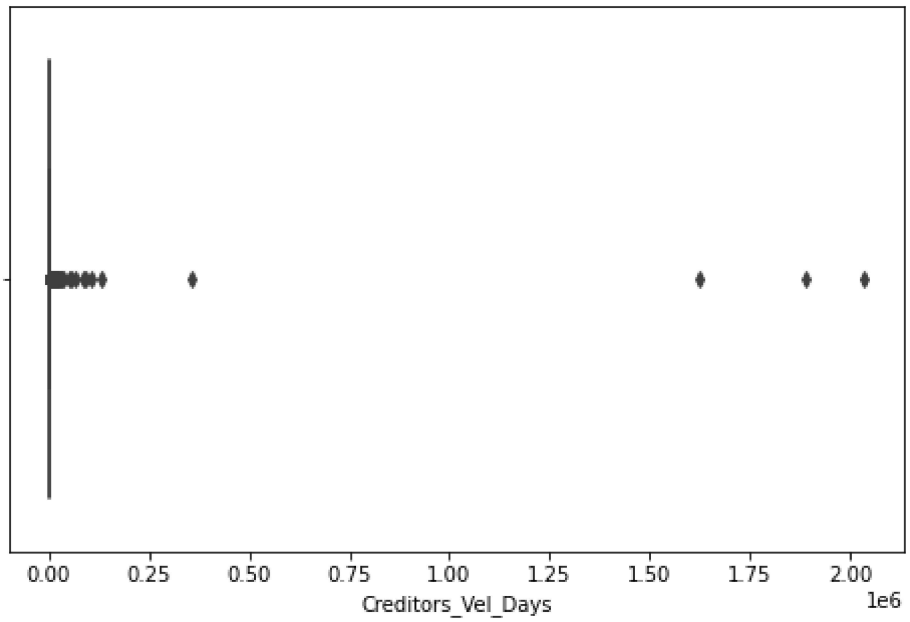

−140000
−120000
−100000
−80000
−60000
−40000
−20000
0
20000
PBITM_perc_Latest

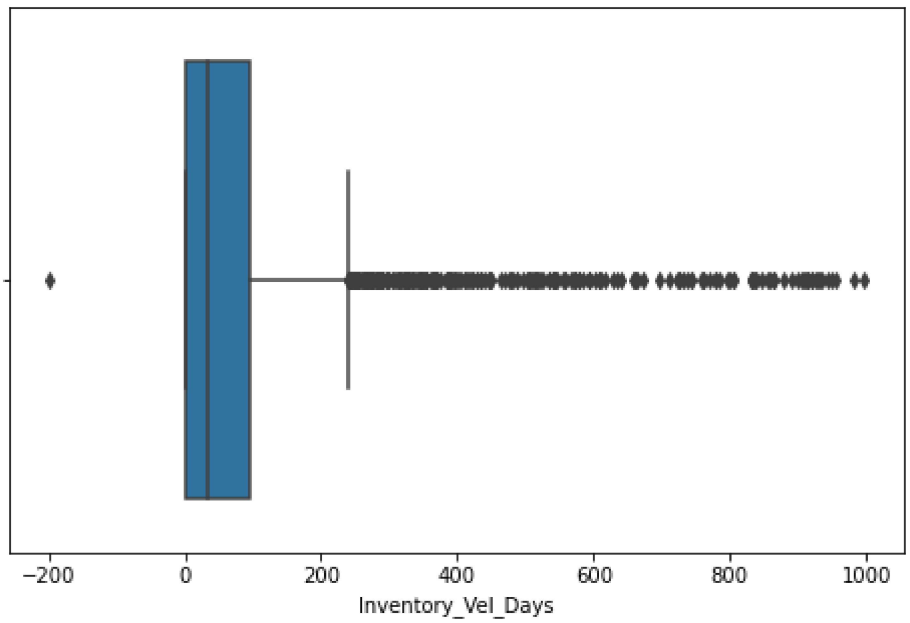

−600000
−500000
−400000
−300000
−200000
−100000
0
PBDTM_perc_Latest

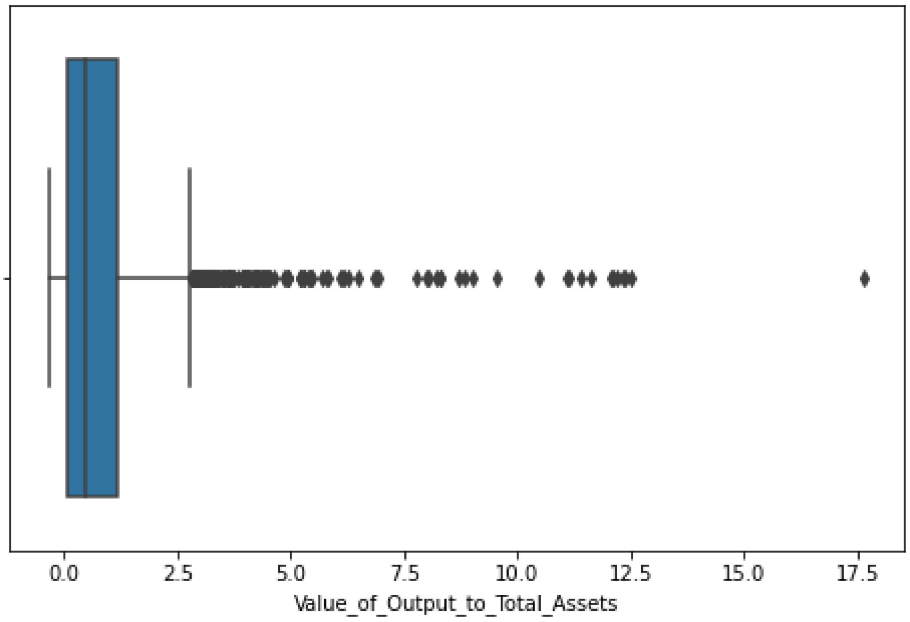

−600000
−500000
−400000
−300000
−200000
−100000
0
CPM_perc_Latest

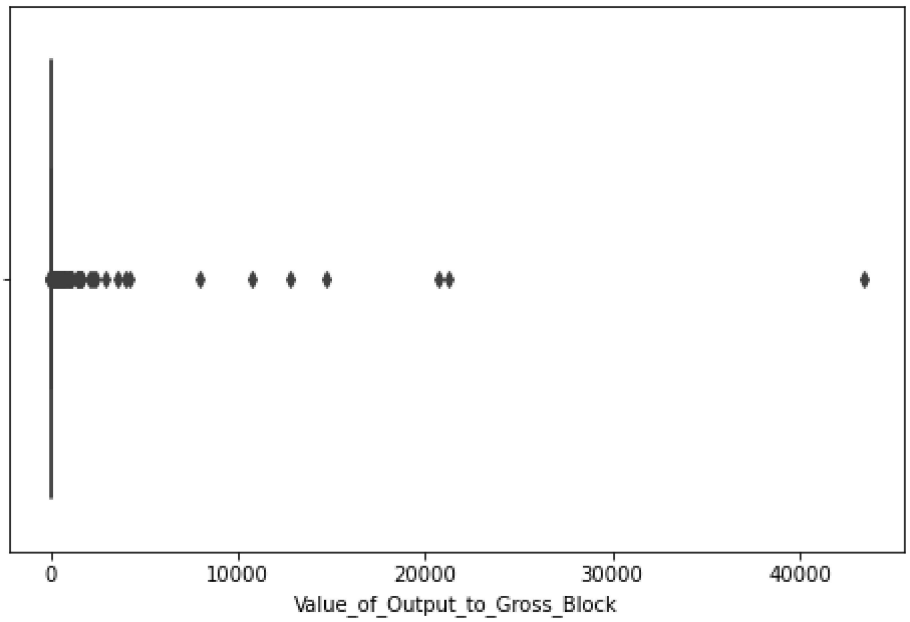

−700000
−600000
−500000
−400000
−300000
−200000
−100000
0
APATM_perc_Latest


0
100000
200000
300000
400000
500000
Debtors_Vel_Days


0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
1e6
Creditors_Vel_Days

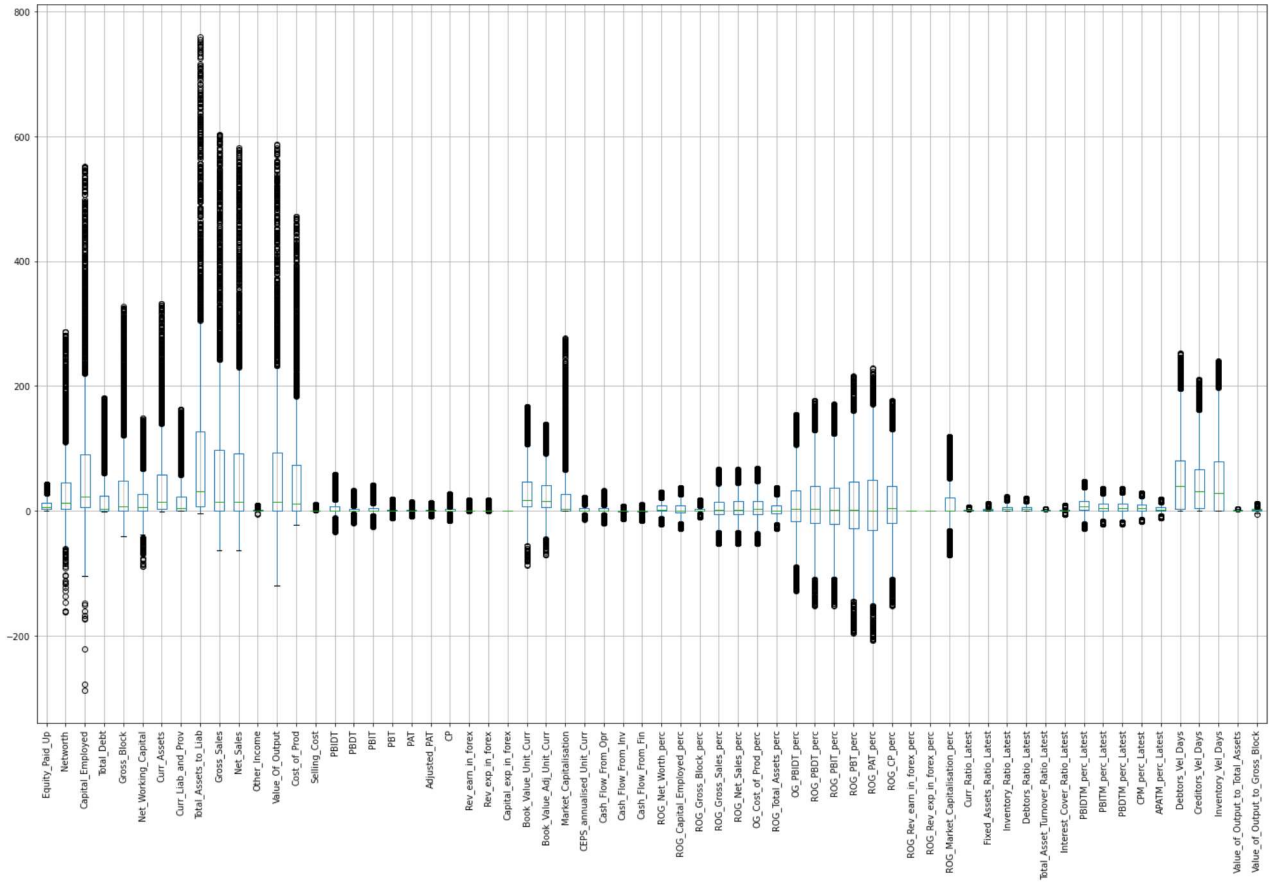

−200
0
200
400
600
800
1000
Inventory_Vel_Days





In [75]:
```
df_x.boxplot(figsize=(25, 15), rot=90)
```

Out[75]: <AxesSubplot:>



In [16]:
```
Q1 = df_x.quantile(0.25)
Q3 = df_x.quantile(0.75)
IQR = Q3 - Q1
UL = Q3 + 1.5*IQR
LL = Q1 - 1.5*IQR
```

In [17]:
```
#Sum of outliers in each column
pd.options.display.max_rows = None
((df_x > UL) | (df_x < LL)).sum()
```

Out[17]:

Networth_Next_Year	676
Equity_Paid_Up	448
Networth	650
Capital_Employed	596
Total_Debt	583
Gross_Block	540
Net_Working_Capital	625
Curr_Assets	577
Curr_Liab_and_Prov	581
Total_Assets_to_Liab	574
Gross_Sales	554
Net_Sales	556
Other_Income	603
Value_Of_Output	559
Cost_of_Prod	560
Selling_Cost	605
PBIDT	671
PBDT	815
PBIT	720

```
PBT                                941
PAT                                959
Adjusted_PAT                       954
CP                                 816
Rev_earn_in_forex                  738
Rev_exp_in_forex                   693
Capital_exp_in_forex               694
Book_Value_Unit_Curr               485
Book_Value_Adj_Unit_Curr           486
Market_Capitalisation              639
CEPS_annualised_Unit_Curr          602
Cash_Flow_From_Opr                 801
Cash_Flow_From_Inv                 876
Cash_Flow_From_Fin                1005
ROG_Net_Worth_perc                 747
ROG_Capital_Employed_perc          572
ROG_Gross_Block_perc               830
ROG_Gross_Sales_perc               671
ROG_Net_Sales_perc                 667
OG_Cost_of_Prod_perc               675
ROG_Total_Assets_perc              483
OG_PBIDT_perc                      611
ROG_PBDT_perc                      628
ROG_PBIT_perc                      616
ROG_PBT_perc                       611
ROG_PAT_perc                       598
ROG_CP_perc                        637
ROG_Rev_earn_in_forex_perc        1317
ROG_Rev_exp_in_forex_perc         1615
ROG_Market_Capitalisation_perc     497
Curr_Ratio_Latest                  565
Fixed_Assets_Ratio_Latest          495
Inventory_Ratio_Latest             375
Debtors_Ratio_Latest               371
Total_Asset_Turnover_Ratio_Latest  201
Interest_Cover_Ratio_Latest        725
PBIDTM_perc_Latest                 595
PBITM_perc_Latest                  717
PBDTM_perc_Latest                  695
CPM_perc_Latest                    720
APATM_perc_Latest                  933
Debtors_Vel_Days                   398
Creditors_Vel_Days                 391
Inventory_Vel_Days                 262
Value_of_Output_to_Total_Assets    150
Value_of_Output_to_Gross_Block     481
dtype: int64
```

In [18]:
```
#Total outliers
((df_x > UL) | (df_x < LL)).sum().sum()
```

Out[18]:
```
42031
```

As we can see there are so many outliers available. So for going ahead with the logistic regression, we have to treat the outliers in a much better way than we treated them every time in the past. So here we are replacing the outliers with Nan.

In [19]:

```
df_x[((df_x > UL) | (df_x < LL))] = np.nan
```

After converting outliers with nan, now we will treat those missing values.

In [20]:
```
#Now checking the missing values
df_x.isnull().sum()
```

Out[20]:

Networth_Next_Year	676
Equity_Paid_Up	448
Networth	650
Capital_Employed	596
Total_Debt	583
Gross_Block	540
Net_Working_Capital	625
Curr_Assets	577
Curr_Liab_and_Prov	581
Total_Assets_to_Liab	574
Gross_Sales	554
Net_Sales	556
Other_Income	603
Value_Of_Output	559
Cost_of_Prod	560
Selling_Cost	605
PBIDT	671
PBDT	815
PBIT	720
PBT	941
PAT	959
Adjusted_PAT	954
CP	816
Rev_earn_in_forex	738
Rev_exp_in_forex	693
Capital_exp_in_forex	694
Book_Value_Unit_Curr	485
Book_Value_Adj_Unit_Curr	490
Market_Capitalisation	639
CEPS_annualised_Unit_Curr	602
Cash_Flow_From_Opr	801
Cash_Flow_From_Inv	876
Cash_Flow_From_Fin	1005
ROG_Net_Worth_perc	747
ROG_Capital_Employed_perc	572
ROG_Gross_Block_perc	830
ROG_Gross_Sales_perc	671
ROG_Net_Sales_perc	667
OG_Cost_of_Prod_perc	675
ROG_Total_Assets_perc	483
OG_PBIDT_perc	611
ROG_PBDT_perc	628
ROG_PBIT_perc	616
ROG_PBT_perc	611
ROG_PAT_perc	598
ROG_CP_perc	637
ROG_Rev_earn_in_forex_perc	1317
ROG_Rev_exp_in_forex_perc	1615
ROG_Market_Capitalisation_perc	497
Curr_Ratio_Latest	566
Fixed_Assets_Ratio_Latest	496

```
Inventory_Ratio_Latest                376
Debtors_Ratio_Latest                  372
Total_Asset_Turnover_Ratio_Latest     202
Interest_Cover_Ratio_Latest           726
PBIDTM_perc_Latest                    596
PBITM_perc_Latest                     718
PBDTM_perc_Latest                     696
CPM_perc_Latest                       721
APATM_perc_Latest                     934
Debtors_Vel_Days                      398
Creditors_Vel_Days                    391
Inventory_Vel_Days                    365
Value_of_Output_to_Total_Assets       150
Value_of_Output_to_Gross_Block        481
dtype: int64
```

In [21]:
```
#Total missing values
df_x.isnull().sum().sum()
```

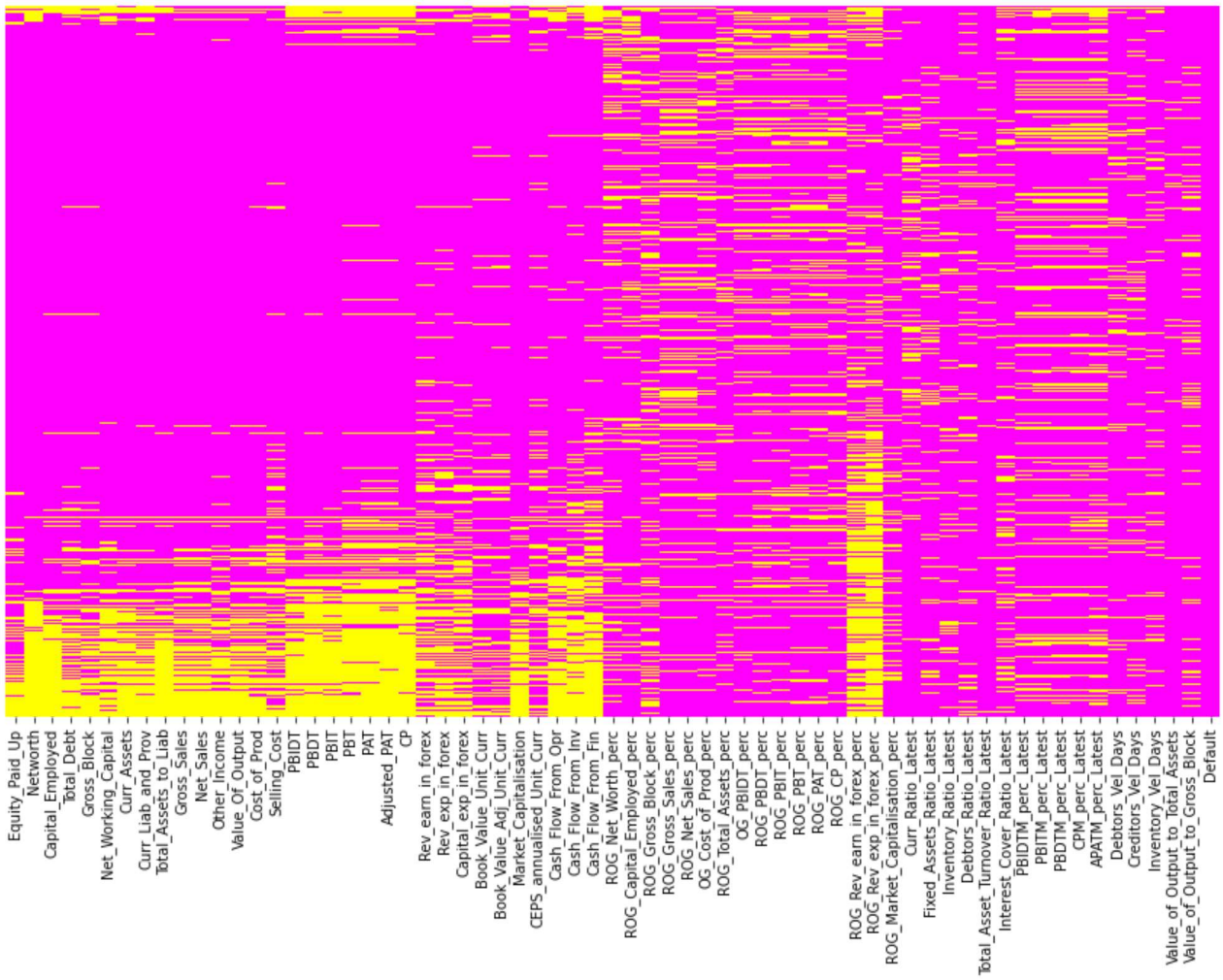
Out[21]:
```
42149
```

In [22]:
```
df_x = df_x.drop(['Networth_Next_Year'], axis = 1) #since this we have already converte
```

In [23]:
```
df_concat = pd.concat([df_x, df_y], axis = 1)
```

In [24]:
```
### Now let's see the heatmap for missing values

plt.figure(figsize = (15,9))
sns.heatmap(df_concat.isnull(), cbar = False, cmap = 'spring', yticklabels = False)
```

Out[24]:
```
<AxesSubplot:>
```



Yellow color here represent the data points which are missing, that we can see that there are many missing data in columns in dataset.

In [25]:
```
##Now here we will also check total missing values in each row (Row Wise missing values
##To get which are the companies has most missing values


df_concat.isnull().sum(axis = 1)
```

Out[25]:
```
0       19
1       34
2       43
3       36
4       35
5        9
6       37
7       34
8       24
9       27
10      20
11      34
12      21
13      35
14      27
15      35
16      34
17      31
```

18	12
19	22
20	23
21	16
22	33
23	38
24	28
25	21
26	14
27	11
28	14
29	33
30	27
31	22
32	21
33	15
34	30
35	28
36	22
37	30
38	11
39	25
40	25
41	27
42	16
43	4
44	23
45	21
46	34
47	27
48	29
49	10
50	28
51	28
52	16
53	35
54	31
55	20
56	17
57	17
58	10
59	12
60	23
61	20
62	29
63	15
64	19
65	16
66	32
67	26
68	15
69	17
70	23
71	16
72	9
73	11
74	34
75	18
76	21
77	26

78	20
79	18
80	20
81	5
82	23
83	10
84	22
85	21
86	14
87	14
88	24
89	23
90	26
91	29
92	17
93	13
94	16
95	3
96	4
97	16
98	6
99	20
100	25
101	15
102	24
103	0
104	15
105	5
106	21
107	17
108	8
109	15
110	5
111	20
112	12
113	6
114	9
115	10
116	18
117	12
118	13
119	9
120	6
121	20
122	14
123	25
124	10
125	23
126	9
127	12
128	19
129	20
130	7
131	7
132	24
133	13
134	20
135	27
136	8
137	5

138	6
139	7
140	12
141	18
142	31
143	11
144	1
145	7
146	14
147	4
148	2
149	1
150	8
151	5
152	6
153	15
154	14
155	14
156	6
157	27
158	3
159	7
160	9
161	6
162	6
163	13
164	17
165	5
166	6
167	15
168	8
169	12
170	6
171	1
172	19
173	15
174	11
175	9
176	6
177	9
178	1
179	10
180	30
181	11
182	3
183	4
184	5
185	2
186	22
187	6
188	13
189	0
190	9
191	17
192	13
193	11
194	16
195	12
196	1
197	4

198	3
199	11
200	19
201	13
202	4
203	7
204	8
205	0
206	3
207	4
208	3
209	20
210	4
211	4
212	2
213	7
214	8
215	7
216	1
217	1
218	2
219	0
220	12
221	13
222	7
223	4
224	9
225	14
226	10
227	2
228	0
229	10
230	2
231	8
232	8
233	6
234	1
235	11
236	19
237	12
238	1
239	11
240	0
241	16
242	1
243	0
244	11
245	9
246	6
247	8
248	17
249	10
250	6
251	8
252	6
253	5
254	5
255	3
256	2
257	9

258	11
259	6
260	4
261	3
262	5
263	1
264	8
265	10
266	6
267	9
268	7
269	1
270	8
271	7
272	1
273	13
274	3
275	6
276	17
277	6
278	4
279	2
280	13
281	7
282	2
283	8
284	16
285	7
286	0
287	0
288	13
289	7
290	5
291	11
292	7
293	2
294	1
295	4
296	4
297	1
298	15
299	6
300	0
301	14
302	8
303	5
304	5
305	8
306	0
307	9
308	3
309	16
310	3
311	0
312	3
313	7
314	5
315	0
316	5
317	10

318	6
319	11
320	8
321	1
322	5
323	9
324	1
325	12
326	4
327	5
328	5
329	14
330	10
331	8
332	2
333	4
334	2
335	3
336	1
337	7
338	6
339	4
340	10
341	17
342	7
343	8
344	5
345	4
346	4
347	9
348	4
349	5
350	4
351	8
352	7
353	5
354	1
355	9
356	9
357	4
358	0
359	2
360	2
361	2
362	4
363	27
364	1
365	0
366	9
367	1
368	7
369	7
370	13
371	12
372	0
373	4
374	11
375	10
376	12
377	12

```
378      2
379      9
380      1
381      5
382      3
383      2
384     11
385      3
386      0
387      0
388      3
389     11
390     10
391      3
392      4
393      7
394      5
395      5
396      7
397      1
398      0
399      2
400      0
401      1
402      4
403      1
404      6
405      1
406      5
407     11
408      1
409     12
410      1
411      1
412      0
413     10
414      9
415      0
416      0
417      9
418      9
419      2
420      5
421      4
422      2
423     12
424      3
425      0
426      2
427      6
428     10
429      4
430      3
431      5
432      9
433      0
434      2
435      3
436      5
437      5
```

438	5
439	4
440	0
441	8
442	10
443	1
444	2
445	3
446	10
447	7
448	7
449	12
450	1
451	3
452	1
453	9
454	11
455	4
456	0
457	0
458	0
459	0
460	2
461	5
462	4
463	7
464	4
465	10
466	3
467	4
468	8
469	9
470	5
471	9
472	2
473	5
474	11
475	1
476	8
477	16
478	9
479	4
480	5
481	4
482	7
483	17
484	12
485	0
486	13
487	6
488	7
489	1
490	2
491	1
492	6
493	2
494	4
495	3
496	10
497	7

498	10
499	5
500	5
501	1
502	2
503	4
504	13
505	5
506	0
507	6
508	6
509	10
510	5
511	14
512	8
513	9
514	9
515	7
516	5
517	5
518	8
519	7
520	5
521	10
522	1
523	12
524	2
525	4
526	18
527	3
528	10
529	0
530	0
531	7
532	6
533	6
534	11
535	10
536	2
537	10
538	7
539	6
540	10
541	4
542	3
543	9
544	11
545	4
546	12
547	15
548	12
549	2
550	1
551	12
552	1
553	5
554	1
555	1
556	5
557	6

558	3
559	3
560	3
561	15
562	9
563	5
564	18
565	6
566	6
567	6
568	0
569	17
570	6
571	10
572	7
573	2
574	14
575	13
576	5
577	8
578	0
579	1
580	10
581	6
582	12
583	5
584	7
585	0
586	1
587	4
588	2
589	6
590	6
591	5
592	0
593	5
594	13
595	1
596	4
597	0
598	2
599	9
600	7
601	7
602	0
603	10
604	2
605	10
606	9
607	2
608	1
609	3
610	3
611	5
612	10
613	4
614	4
615	0
616	13
617	2

618	4
619	1
620	0
621	6
622	3
623	12
624	2
625	23
626	5
627	1
628	14
629	3
630	6
631	9
632	10
633	5
634	9
635	10
636	3
637	10
638	17
639	8
640	16
641	9
642	3
643	10
644	3
645	18
646	2
647	4
648	12
649	13
650	4
651	7
652	14
653	5
654	6
655	1
656	12
657	1
658	5
659	7
660	5
661	6
662	7
663	7
664	9
665	5
666	5
667	7
668	11
669	10
670	1
671	9
672	0
673	10
674	7
675	9
676	2
677	3

678	0
679	5
680	7
681	1
682	6
683	9
684	1
685	12
686	8
687	7
688	6
689	11
690	7
691	6
692	6
693	2
694	5
695	4
696	7
697	3
698	4
699	9
700	1
701	9
702	12
703	3
704	1
705	9
706	6
707	7
708	14
709	1
710	10
711	7
712	1
713	0
714	6
715	9
716	5
717	3
718	6
719	4
720	7
721	2
722	5
723	2
724	9
725	2
726	13
727	7
728	3
729	5
730	12
731	6
732	2
733	0
734	8
735	4
736	3
737	5

```
738     11
739      2
740      8
741      3
742      4
743      5
744      9
745      5
746      2
747      3
748      5
749      0
750      3
751      7
752      4
753      2
754      1
755      4
756      6
757     13
758      7
759     10
760      1
761      6
762      3
763     13
764     10
765      5
766      6
767      4
768      1
769      0
770      4
771      9
772      6
773      2
774      7
775      3
776      7
777      2
778     11
779      7
780      4
781      3
782      4
783      3
784      1
785      2
786      9
787      7
788      8
789      8
790      3
791      3
792      5
793      1
794      6
795     14
796      5
797      4
```

```
798     13
799      9
800      4
801     14
802      8
803      6
804     10
805      8
806     10
807      2
808      3
809      2
810      1
811      3
812      2
813     12
814      2
815      2
816      8
817      1
818      7
819      6
820     11
821      5
822      9
823     11
824     13
825      6
826      3
827      6
828      2
829     11
830     10
831      2
832      3
833      9
834      4
835     10
836      2
837     14
838      2
839      0
840      9
841      4
842      4
843      1
844      4
845      6
846     14
847      1
848      6
849      7
850     15
851      4
852     10
853      7
854      8
855     10
856      1
857      9
```

858	2
859	9
860	5
861	3
862	2
863	0
864	3
865	3
866	5
867	1
868	4
869	5
870	0
871	3
872	4
873	4
874	8
875	0
876	13
877	2
878	14
879	8
880	4
881	7
882	4
883	6
884	10
885	3
886	6
887	5
888	1
889	2
890	6
891	8
892	3
893	17
894	7
895	0
896	14
897	3
898	8
899	15
900	7
901	8
902	1
903	3
904	4
905	7
906	30
907	7
908	5
909	12
910	2
911	6
912	10
913	13
914	5
915	9
916	12
917	6

918	8
919	11
920	10
921	5
922	1
923	6
924	2
925	4
926	5
927	4
928	4
929	13
930	11
931	0
932	8
933	3
934	2
935	8
936	3
937	11
938	7
939	19
940	3
941	10
942	6
943	6
944	0
945	2
946	6
947	0
948	6
949	5
950	5
951	12
952	7
953	11
954	11
955	11
956	10
957	2
958	2
959	14
960	10
961	1
962	4
963	14
964	7
965	10
966	3
967	4
968	7
969	5
970	5
971	15
972	4
973	14
974	3
975	11
976	6
977	5

978	7
979	0
980	4
981	0
982	13
983	13
984	1
985	5
986	9
987	9
988	3
989	7
990	6
991	6
992	10
993	5
994	7
995	1
996	16
997	2
998	2
999	7
1000	0
1001	7
1002	10
1003	5
1004	3
1005	1
1006	21
1007	7
1008	8
1009	7
1010	11
1011	14
1012	1
1013	8
1014	2
1015	13
1016	1
1017	8
1018	4
1019	11
1020	2
1021	4
1022	4
1023	5
1024	12
1025	4
1026	15
1027	1
1028	8
1029	3
1030	8
1031	7
1032	6
1033	1
1034	5
1035	7
1036	3
1037	7

```
1038    10
1039     1
1040     1
1041     4
1042     8
1043    11
1044     2
1045     2
1046     5
1047     7
1048     0
1049     3
1050    10
1051     3
1052     9
1053     2
1054    10
1055     3
1056     4
1057    12
1058     3
1059    11
1060    11
1061    10
1062     2
1063     8
1064     3
1065     1
1066    10
1067     8
1068     3
1069    10
1070     3
1071    15
1072    10
1073     6
1074     6
1075     0
1076     8
1077    12
1078     4
1079    10
1080     6
1081     7
1082    13
1083     6
1084     2
1085     0
1086     6
1087    13
1088    10
1089     2
1090     1
1091     8
1092     6
1093     3
1094     3
1095     3
1096     8
1097    10
```

1098	7
1099	9
1100	10
1101	5
1102	7
1103	15
1104	8
1105	4
1106	0
1107	18
1108	9
1109	1
1110	11
1111	2
1112	4
1113	11
1114	9
1115	3
1116	5
1117	6
1118	0
1119	4
1120	2
1121	3
1122	12
1123	1
1124	7
1125	14
1126	7
1127	9
1128	10
1129	8
1130	6
1131	9
1132	4
1133	1
1134	3
1135	17
1136	9
1137	16
1138	4
1139	0
1140	2
1141	17
1142	1
1143	4
1144	9
1145	7
1146	1
1147	10
1148	7
1149	2
1150	3
1151	7
1152	1
1153	0
1154	4
1155	15
1156	8
1157	13

```
1158     6
1159     1
1160     4
1161     5
1162     4
1163     1
1164    11
1165     6
1166     2
1167     1
1168     4
1169     0
1170     7
1171     2
1172    12
1173    11
1174    16
1175     6
1176     4
1177    12
1178     2
1179    10
1180    10
1181     1
1182     9
1183    14
1184     4
1185     6
1186    11
1187     0
1188    11
1189     2
1190     7
1191     4
1192    10
1193     7
1194     1
1195     2
1196     8
1197    10
1198     2
1199     3
1200     7
1201     4
1202     2
1203     7
1204     4
1205     3
1206     6
1207     0
1208     3
1209     4
1210     6
1211    11
1212     3
1213     8
1214     7
1215     2
1216     0
1217     1
```

1218	1
1219	5
1220	9
1221	6
1222	13
1223	11
1224	8
1225	6
1226	5
1227	7
1228	2
1229	6
1230	6
1231	3
1232	9
1233	9
1234	12
1235	8
1236	2
1237	7
1238	9
1239	12
1240	3
1241	3
1242	3
1243	24
1244	2
1245	10
1246	6
1247	4
1248	2
1249	4
1250	4
1251	6
1252	1
1253	3
1254	8
1255	1
1256	0
1257	3
1258	3
1259	1
1260	7
1261	13
1262	5
1263	8
1264	12
1265	3
1266	2
1267	9
1268	1
1269	6
1270	3
1271	0
1272	8
1273	2
1274	12
1275	0
1276	7
1277	1

1278	6
1279	2
1280	6
1281	6
1282	1
1283	12
1284	8
1285	5
1286	8
1287	1
1288	2
1289	4
1290	2
1291	12
1292	1
1293	1
1294	4
1295	4
1296	4
1297	11
1298	3
1299	1
1300	13
1301	1
1302	2
1303	7
1304	3
1305	5
1306	7
1307	5
1308	3
1309	8
1310	9
1311	3
1312	14
1313	3
1314	4
1315	2
1316	4
1317	10
1318	9
1319	9
1320	3
1321	2
1322	14
1323	6
1324	1
1325	4
1326	6
1327	8
1328	7
1329	8
1330	10
1331	10
1332	2
1333	12
1334	7
1335	11
1336	13
1337	5

1338	1
1339	4
1340	3
1341	3
1342	4
1343	8
1344	0
1345	0
1346	3
1347	6
1348	14
1349	2
1350	3
1351	4
1352	9
1353	4
1354	10
1355	11
1356	10
1357	4
1358	3
1359	12
1360	2
1361	6
1362	7
1363	4
1364	8
1365	6
1366	5
1367	8
1368	0
1369	7
1370	15
1371	10
1372	12
1373	9
1374	11
1375	8
1376	8
1377	8
1378	8
1379	15
1380	0
1381	3
1382	1
1383	3
1384	5
1385	0
1386	16
1387	18
1388	4
1389	5
1390	6
1391	7
1392	0
1393	5
1394	9
1395	0
1396	1
1397	5

1398	5
1399	1
1400	4
1401	6
1402	7
1403	5
1404	2
1405	8
1406	14
1407	11
1408	12
1409	3
1410	4
1411	7
1412	12
1413	7
1414	10
1415	11
1416	1
1417	3
1418	2
1419	8
1420	4
1421	3
1422	11
1423	10
1424	6
1425	2
1426	0
1427	2
1428	11
1429	15
1430	4
1431	2
1432	9
1433	8
1434	3
1435	3
1436	2
1437	1
1438	2
1439	2
1440	4
1441	3
1442	0
1443	16
1444	1
1445	7
1446	1
1447	1
1448	4
1449	6
1450	16
1451	4
1452	3
1453	2
1454	11
1455	6
1456	2
1457	5

1458	13
1459	3
1460	1
1461	6
1462	3
1463	12
1464	12
1465	11
1466	7
1467	6
1468	4
1469	2
1470	9
1471	7
1472	6
1473	2
1474	3
1475	7
1476	5
1477	4
1478	9
1479	3
1480	5
1481	20
1482	9
1483	2
1484	5
1485	4
1486	7
1487	6
1488	4
1489	4
1490	4
1491	0
1492	8
1493	4
1494	8
1495	6
1496	4
1497	8
1498	15
1499	7
1500	4
1501	9
1502	6
1503	0
1504	4
1505	10
1506	13
1507	11
1508	4
1509	10
1510	1
1511	9
1512	2
1513	14
1514	1
1515	2
1516	2
1517	3

1518	10
1519	5
1520	10
1521	8
1522	6
1523	5
1524	1
1525	1
1526	11
1527	6
1528	2
1529	3
1530	5
1531	7
1532	10
1533	7
1534	8
1535	3
1536	5
1537	9
1538	9
1539	1
1540	4
1541	12
1542	11
1543	4
1544	2
1545	10
1546	4
1547	15
1548	15
1549	3
1550	6
1551	1
1552	3
1553	11
1554	19
1555	4
1556	5
1557	6
1558	6
1559	3
1560	10
1561	5
1562	5
1563	1
1564	5
1565	4
1566	5
1567	12
1568	8
1569	12
1570	12
1571	4
1572	8
1573	7
1574	5
1575	2
1576	1
1577	3

1578	6
1579	3
1580	9
1581	7
1582	0
1583	5
1584	6
1585	4
1586	2
1587	2
1588	6
1589	2
1590	2
1591	12
1592	4
1593	8
1594	0
1595	4
1596	5
1597	1
1598	4
1599	3
1600	11
1601	20
1602	8
1603	3
1604	7
1605	7
1606	10
1607	6
1608	1
1609	13
1610	7
1611	1
1612	8
1613	10
1614	5
1615	8
1616	4
1617	9
1618	7
1619	5
1620	3
1621	1
1622	9
1623	4
1624	6
1625	6
1626	2
1627	2
1628	3
1629	6
1630	3
1631	4
1632	6
1633	11
1634	3
1635	5
1636	5
1637	5

```
1638    17
1639     4
1640    13
1641    11
1642     4
1643    12
1644    17
1645    12
1646    10
1647     3
1648     6
1649     5
1650     3
1651     3
1652    14
1653     5
1654     2
1655     3
1656     5
1657    12
1658     2
1659     2
1660     5
1661    15
1662     9
1663     2
1664    10
1665     3
1666     9
1667     0
1668     7
1669     2
1670     0
1671    22
1672     4
1673    17
1674     5
1675     9
1676    13
1677     1
1678     4
1679    11
1680     2
1681     3
1682     1
1683     7
1684     1
1685     3
1686     6
1687     5
1688     4
1689     4
1690     1
1691    11
1692    12
1693     2
1694     4
1695     2
1696     4
1697     9
```

1698	5
1699	13
1700	1
1701	2
1702	9
1703	3
1704	2
1705	11
1706	11
1707	3
1708	6
1709	2
1710	9
1711	7
1712	6
1713	7
1714	2
1715	10
1716	11
1717	8
1718	16
1719	6
1720	3
1721	0
1722	6
1723	4
1724	2
1725	5
1726	7
1727	8
1728	2
1729	7
1730	13
1731	6
1732	19
1733	16
1734	6
1735	0
1736	12
1737	2
1738	3
1739	2
1740	16
1741	2
1742	7
1743	4
1744	3
1745	3
1746	7
1747	15
1748	5
1749	6
1750	17
1751	10
1752	12
1753	11
1754	5
1755	8
1756	15
1757	9

1758	3
1759	5
1760	9
1761	5
1762	3
1763	7
1764	4
1765	1
1766	4
1767	7
1768	2
1769	9
1770	3
1771	5
1772	0
1773	0
1774	7
1775	9
1776	9
1777	7
1778	6
1779	9
1780	7
1781	5
1782	1
1783	2
1784	6
1785	1
1786	4
1787	14
1788	3
1789	3
1790	6
1791	7
1792	11
1793	7
1794	6
1795	1
1796	0
1797	2
1798	1
1799	0
1800	13
1801	8
1802	2
1803	3
1804	5
1805	14
1806	7
1807	2
1808	7
1809	4
1810	5
1811	0
1812	2
1813	3
1814	13
1815	9
1816	5
1817	8

1818	1
1819	3
1820	2
1821	6
1822	6
1823	3
1824	5
1825	7
1826	2
1827	3
1828	1
1829	3
1830	5
1831	11
1832	1
1833	7
1834	8
1835	3
1836	3
1837	5
1838	5
1839	14
1840	7
1841	8
1842	10
1843	10
1844	5
1845	4
1846	3
1847	10
1848	9
1849	6
1850	8
1851	3
1852	1
1853	3
1854	0
1855	10
1856	22
1857	2
1858	3
1859	4
1860	3
1861	7
1862	1
1863	3
1864	8
1865	3
1866	13
1867	3
1868	7
1869	7
1870	18
1871	8
1872	4
1873	3
1874	5
1875	7
1876	2
1877	7

```
1878     6
1879    10
1880     1
1881     6
1882     5
1883    11
1884     0
1885     6
1886     8
1887     0
1888     1
1889     6
1890    14
1891     5
1892    14
1893     3
1894     6
1895     7
1896     3
1897     3
1898     5
1899     4
1900     4
1901     2
1902     0
1903     4
1904     3
1905     3
1906     9
1907    14
1908     1
1909     4
1910     1
1911     1
1912     4
1913    10
1914     4
1915     6
1916     8
1917     4
1918     5
1919     0
1920     3
1921    11
1922     5
1923     2
1924     3
1925     1
1926     3
1927     3
1928     9
1929     4
1930     7
1931     7
1932     2
1933     2
1934     2
1935     4
1936     4
1937     6
```

1938	4
1939	13
1940	4
1941	10
1942	10
1943	15
1944	8
1945	7
1946	7
1947	14
1948	5
1949	3
1950	4
1951	27
1952	0
1953	2
1954	2
1955	2
1956	12
1957	10
1958	7
1959	11
1960	11
1961	7
1962	5
1963	9
1964	9
1965	6
1966	11
1967	17
1968	12
1969	5
1970	1
1971	9
1972	7
1973	7
1974	3
1975	4
1976	2
1977	5
1978	1
1979	4
1980	6
1981	5
1982	2
1983	8
1984	8
1985	9
1986	12
1987	7
1988	10
1989	11
1990	5
1991	2
1992	10
1993	6
1994	7
1995	4
1996	1
1997	5

1998	12
1999	2
2000	6
2001	3
2002	7
2003	14
2004	16
2005	1
2006	12
2007	5
2008	12
2009	8
2010	15
2011	11
2012	2
2013	4
2014	3
2015	11
2016	3
2017	14
2018	5
2019	2
2020	10
2021	3
2022	6
2023	5
2024	11
2025	4
2026	3
2027	4
2028	4
2029	5
2030	0
2031	5
2032	2
2033	1
2034	8
2035	2
2036	7
2037	12
2038	4
2039	3
2040	5
2041	15
2042	3
2043	4
2044	7
2045	6
2046	8
2047	3
2048	3
2049	2
2050	3
2051	4
2052	5
2053	1
2054	7
2055	3
2056	27
2057	4

```
2058    21
2059     7
2060    14
2061     2
2062     4
2063     8
2064     2
2065     4
2066     4
2067     3
2068     7
2069     4
2070    13
2071    11
2072     7
2073    16
2074     2
2075     7
2076    13
2077     6
2078     1
2079    10
2080    12
2081     4
2082     4
2083     8
2084     8
2085     1
2086     6
2087    11
2088     6
2089     8
2090    15
2091     7
2092     5
2093     5
2094     7
2095     3
2096     0
2097     6
2098    10
2099     6
2100     5
2101     6
2102     4
2103     5
2104     5
2105     7
2106    11
2107     5
2108    10
2109    12
2110     5
2111     7
2112     7
2113     9
2114     7
2115     7
2116     9
2117     6
```

```
2118    16
2119     8
2120     1
2121     6
2122     6
2123     6
2124     4
2125     4
2126     7
2127     3
2128    12
2129    11
2130     4
2131     3
2132     4
2133    15
2134     4
2135    16
2136     5
2137     3
2138     1
2139    21
2140     4
2141    16
2142     0
2143    10
2144    12
2145     7
2146     3
2147    11
2148     4
2149     0
2150     8
2151     2
2152     6
2153     9
2154     8
2155    20
2156     0
2157    14
2158     9
2159     1
2160     3
2161     4
2162     9
2163     7
2164     6
2165     5
2166     4
2167     0
2168     5
2169     9
2170     6
2171     9
2172     4
2173     4
2174     9
2175     2
2176     5
2177     2
```

2178	2
2179	9
2180	6
2181	12
2182	7
2183	6
2184	1
2185	3
2186	3
2187	7
2188	4
2189	3
2190	8
2191	11
2192	14
2193	1
2194	10
2195	14
2196	9
2197	6
2198	12
2199	8
2200	7
2201	5
2202	1
2203	6
2204	12
2205	7
2206	8
2207	7
2208	7
2209	5
2210	0
2211	19
2212	7
2213	5
2214	5
2215	4
2216	6
2217	8
2218	7
2219	2
2220	5
2221	5
2222	6
2223	7
2224	13
2225	6
2226	7
2227	14
2228	3
2229	4
2230	2
2231	1
2232	8
2233	21
2234	8
2235	14
2236	3
2237	10

```
2238    12
2239     2
2240     4
2241     4
2242    12
2243     7
2244     7
2245    11
2246     3
2247     6
2248     9
2249     6
2250     2
2251     8
2252     1
2253    32
2254     5
2255    12
2256     1
2257     7
2258     6
2259     7
2260     1
2261     8
2262    15
2263    27
2264     6
2265    10
2266     5
2267     8
2268     9
2269     6
2270    10
2271     9
2272     4
2273     1
2274     7
2275     6
2276    10
2277     6
2278     8
2279    14
2280    12
2281     5
2282     8
2283    18
2284     4
2285     0
2286     7
2287     8
2288     3
2289    14
2290    15
2291    21
2292     1
2293     5
2294     6
2295     5
2296     6
2297     6
```

2298	3
2299	6
2300	4
2301	7
2302	5
2303	11
2304	27
2305	9
2306	8
2307	3
2308	15
2309	8
2310	7
2311	7
2312	4
2313	5
2314	1
2315	21
2316	2
2317	8
2318	2
2319	6
2320	10
2321	8
2322	12
2323	6
2324	5
2325	2
2326	3
2327	5
2328	6
2329	4
2330	6
2331	14
2332	13
2333	9
2334	9
2335	19
2336	3
2337	4
2338	5
2339	8
2340	12
2341	6
2342	14
2343	8
2344	8
2345	12
2346	6
2347	6
2348	12
2349	7
2350	5
2351	6
2352	12
2353	8
2354	4
2355	6
2356	9
2357	13

2358	2
2359	10
2360	6
2361	5
2362	9
2363	22
2364	11
2365	5
2366	10
2367	8
2368	6
2369	7
2370	5
2371	13
2372	12
2373	6
2374	3
2375	12
2376	4
2377	5
2378	15
2379	6
2380	9
2381	9
2382	4
2383	3
2384	1
2385	8
2386	12
2387	16
2388	15
2389	10
2390	10
2391	11
2392	15
2393	16
2394	4
2395	4
2396	6
2397	10
2398	13
2399	10
2400	1
2401	17
2402	3
2403	15
2404	4
2405	6
2406	7
2407	5
2408	5
2409	36
2410	3
2411	4
2412	6
2413	15
2414	6
2415	4
2416	10
2417	5

```
2418    17
2419     6
2420     0
2421     5
2422     4
2423    29
2424     4
2425     4
2426    12
2427     5
2428     8
2429     8
2430     3
2431    11
2432     8
2433    14
2434     7
2435     7
2436     8
2437    16
2438     8
2439    13
2440    10
2441    11
2442     4
2443    20
2444     7
2445    12
2446     6
2447    28
2448    12
2449    12
2450    13
2451     4
2452    21
2453     8
2454     8
2455     9
2456     6
2457     7
2458     6
2459    13
2460     9
2461    16
2462     8
2463    13
2464     9
2465     6
2466    14
2467    14
2468    14
2469    10
2470    12
2471     4
2472     9
2473     2
2474     6
2475     3
2476    11
2477     2
```

2478	11
2479	4
2480	10
2481	18
2482	3
2483	3
2484	18
2485	5
2486	5
2487	9
2488	6
2489	11
2490	19
2491	23
2492	8
2493	4
2494	21
2495	19
2496	11
2497	10
2498	6
2499	8
2500	16
2501	3
2502	6
2503	5
2504	5
2505	10
2506	7
2507	12
2508	8
2509	5
2510	2
2511	5
2512	5
2513	11
2514	8
2515	5
2516	10
2517	8
2518	13
2519	13
2520	26
2521	9
2522	3
2523	9
2524	6
2525	3
2526	24
2527	9
2528	5
2529	5
2530	16
2531	7
2532	5
2533	2
2534	9
2535	12
2536	11
2537	6

```
2538    16
2539    11
2540    10
2541     4
2542    11
2543    12
2544     6
2545     8
2546    12
2547    21
2548    15
2549    15
2550    22
2551     9
2552     7
2553     4
2554     8
2555     4
2556     7
2557    10
2558     8
2559     9
2560     9
2561     5
2562     6
2563     4
2564     7
2565     7
2566     8
2567     3
2568    15
2569    12
2570     9
2571     9
2572     5
2573     8
2574    14
2575    11
2576     4
2577     3
2578    10
2579     8
2580    39
2581     6
2582     7
2583    11
2584    17
2585     5
2586    16
2587     7
2588    10
2589    10
2590     6
2591     8
2592    21
2593     6
2594    12
2595    10
2596     4
2597     8
```

```
2598    11
2599     6
2600     7
2601    17
2602    27
2603     6
2604     7
2605     8
2606     8
2607    10
2608     8
2609     5
2610    10
2611     6
2612    11
2613    13
2614     6
2615     5
2616    14
2617    12
2618    20
2619    14
2620     7
2621    16
2622     9
2623    17
2624    24
2625    11
2626     6
2627    11
2628    10
2629    14
2630     7
2631     8
2632    32
2633    10
2634     5
2635    12
2636     5
2637    18
2638    14
2639    16
2640     6
2641    14
2642     3
2643    27
2644    14
2645     1
2646    22
2647     5
2648     2
2649     9
2650    16
2651    10
2652     9
2653     3
2654    17
2655     7
2656    13
2657     7
```

2658	5
2659	7
2660	7
2661	4
2662	23
2663	22
2664	12
2665	11
2666	4
2667	12
2668	6
2669	8
2670	11
2671	6
2672	17
2673	3
2674	4
2675	9
2676	6
2677	10
2678	6
2679	5
2680	10
2681	5
2682	13
2683	6
2684	22
2685	5
2686	7
2687	4
2688	9
2689	5
2690	21
2691	7
2692	5
2693	2
2694	6
2695	6
2696	6
2697	3
2698	9
2699	8
2700	12
2701	4
2702	10
2703	10
2704	15
2705	6
2706	9
2707	9
2708	10
2709	12
2710	7
2711	16
2712	22
2713	14
2714	6
2715	14
2716	27
2717	13

2718	37
2719	17
2720	7
2721	4
2722	22
2723	12
2724	10
2725	23
2726	15
2727	17
2728	6
2729	19
2730	9
2731	6
2732	9
2733	6
2734	18
2735	26
2736	13
2737	8
2738	19
2739	11
2740	8
2741	20
2742	8
2743	31
2744	25
2745	7
2746	18
2747	12
2748	21
2749	19
2750	11
2751	9
2752	18
2753	17
2754	9
2755	17
2756	6
2757	12
2758	25
2759	11
2760	29
2761	26
2762	22
2763	8
2764	8
2765	13
2766	17
2767	10
2768	6
2769	13
2770	12
2771	18
2772	15
2773	6
2774	12
2775	23
2776	22
2777	21

2778	10
2779	10
2780	26
2781	13
2782	5
2783	11
2784	10
2785	23
2786	4
2787	10
2788	22
2789	5
2790	18
2791	23
2792	7
2793	23
2794	11
2795	35
2796	8
2797	13
2798	16
2799	11
2800	15
2801	9
2802	4
2803	6
2804	4
2805	14
2806	25
2807	31
2808	13
2809	18
2810	10
2811	22
2812	16
2813	32
2814	6
2815	13
2816	13
2817	4
2818	19
2819	6
2820	8
2821	13
2822	26
2823	21
2824	27
2825	17
2826	13
2827	10
2828	25
2829	10
2830	21
2831	28
2832	17
2833	20
2834	8
2835	14
2836	7
2837	6

```
2838    21
2839    11
2840     9
2841    20
2842    30
2843    20
2844     9
2845    11
2846    19
2847    23
2848    15
2849    16
2850    16
2851     7
2852     9
2853    27
2854    17
2855     4
2856    12
2857    24
2858    23
2859    15
2860    17
2861    15
2862    19
2863    19
2864    25
2865    16
2866     0
2867    11
2868     3
2869    14
2870    16
2871    15
2872    22
2873    13
2874    20
2875     7
2876    24
2877    13
2878    22
2879    19
2880    15
2881     5
2882    10
2883    24
2884    13
2885    30
2886    19
2887    24
2888    11
2889     9
2890    11
2891    16
2892    20
2893    10
2894     6
2895    15
2896    12
2897    16
```

2898	11
2899	24
2900	9
2901	22
2902	16
2903	26
2904	3
2905	18
2906	10
2907	28
2908	30
2909	28
2910	20
2911	13
2912	31
2913	22
2914	17
2915	29
2916	15
2917	25
2918	17
2919	27
2920	25
2921	25
2922	14
2923	19
2924	7
2925	16
2926	34
2927	19
2928	9
2929	26
2930	11
2931	23
2932	18
2933	6
2934	18
2935	25
2936	21
2937	17
2938	6
2939	14
2940	19
2941	14
2942	8
2943	25
2944	28
2945	19
2946	22
2947	21
2948	24
2949	22
2950	21
2951	6
2952	14
2953	25
2954	29
2955	20
2956	21
2957	25

2958	11
2959	28
2960	24
2961	14
2962	27
2963	25
2964	18
2965	25
2966	12
2967	13
2968	19
2969	22
2970	9
2971	19
2972	6
2973	23
2974	23
2975	19
2976	29
2977	31
2978	15
2979	31
2980	15
2981	13
2982	27
2983	14
2984	21
2985	20
2986	11
2987	21
2988	8
2989	7
2990	18
2991	18
2992	20
2993	15
2994	13
2995	19
2996	20
2997	10
2998	30
2999	26
3000	29
3001	6
3002	28
3003	31
3004	12
3005	23
3006	23
3007	23
3008	22
3009	22
3010	31
3011	28
3012	8
3013	16
3014	29
3015	21
3016	21
3017	28

```
3018    27
3019    29
3020    20
3021    28
3022    32
3023    35
3024     6
3025    15
3026    27
3027    31
3028    17
3029    20
3030    16
3031    29
3032    14
3033    14
3034    17
3035    30
3036    26
3037    28
3038    30
3039    28
3040    24
3041    34
3042    22
3043    11
3044    22
3045    12
3046    35
3047    22
3048    21
3049    26
3050    28
3051    31
3052    33
3053    31
3054    29
3055    28
3056    19
3057    30
3058    19
3059    27
3060    16
3061    33
3062    29
3063     4
3064    26
3065     3
3066    23
3067    10
3068    23
3069    24
3070    31
3071    25
3072    19
3073    31
3074    14
3075    23
3076    26
3077    32
```

```
3078    31
3079    31
3080    33
3081    20
3082    23
3083     5
3084    30
3085    33
3086    24
3087    27
3088    26
3089    20
3090    28
3091    23
3092    22
3093    26
3094    17
3095    31
3096    30
3097     5
3098    21
3099    11
3100    27
3101    12
3102    11
3103    25
3104    27
3105    34
3106    29
3107    19
3108    31
3109    42
3110    23
3111    28
3112    31
3113    14
3114    28
3115    25
3116    33
3117    27
3118    29
3119    29
3120    10
3121    24
3122    12
3123     9
3124    28
3125    32
3126    26
3127    34
3128    24
3129    13
3130    22
3131    22
3132    19
3133    26
3134    20
3135    26
3136    32
3137    17
```

```
3138    29
3139    28
3140    30
3141    21
3142    31
3143    30
3144    32
3145    26
3146    10
3147    20
3148    19
3149    31
3150    32
3151    27
3152    32
3153    34
3154    29
3155    36
3156    23
3157    28
3158    16
3159    17
3160    13
3161    31
3162    22
3163    28
3164    30
3165    30
3166    26
3167    32
3168    30
3169    32
3170    27
3171    29
3172    27
3173    45
3174    25
3175    15
3176    28
3177    29
3178    15
3179    13
3180    15
3181    30
3182    39
3183    29
3184    28
3185    20
3186    34
3187    15
3188    36
3189    19
3190    24
3191    33
3192    27
3193    29
3194    31
3195    18
3196    21
3197    30
```

3198	26
3199	25
3200	17
3201	15
3202	32
3203	24
3204	19
3205	20
3206	28
3207	13
3208	28
3209	28
3210	28
3211	30
3212	28
3213	31
3214	23
3215	8
3216	26
3217	18
3218	37
3219	6
3220	24
3221	29
3222	31
3223	25
3224	23
3225	24
3226	26
3227	28
3228	36
3229	26
3230	22
3231	28
3232	31
3233	32
3234	26
3235	24
3236	29
3237	28
3238	23
3239	34
3240	31
3241	30
3242	32
3243	32
3244	19
3245	27
3246	38
3247	29
3248	32
3249	32
3250	28
3251	19
3252	29
3253	16
3254	31
3255	29
3256	32
3257	36

```
3258    27
3259    22
3260    31
3261    32
3262    34
3263    31
3264    29
3265    26
3266    23
3267    25
3268    30
3269    24
3270    26
3271    17
3272    23
3273    28
3274    32
3275    36
3276    39
3277    16
3278    14
3279    31
3280    30
3281    30
3282    33
3283    29
3284    22
3285    31
3286    30
3287    32
3288    31
3289    31
3290    25
3291    30
3292    30
3293    33
3294    29
3295    41
3296    35
3297    12
3298    23
3299    33
3300    30
3301    34
3302    32
3303    22
3304    23
3305    28
3306    34
3307    31
3308    30
3309    29
3310    10
3311    27
3312    29
3313    27
3314    33
3315    32
3316    25
3317    25
```

3318	41
3319	32
3320	33
3321	21
3322	33
3323	31
3324	36
3325	32
3326	33
3327	33
3328	26
3329	32
3330	30
3331	30
3332	34
3333	33
3334	30
3335	33
3336	28
3337	30
3338	32
3339	31
3340	34
3341	33
3342	32
3343	30
3344	28
3345	28
3346	32
3347	31
3348	33
3349	30
3350	18
3351	31
3352	28
3353	34
3354	30
3355	30
3356	32
3357	34
3358	25
3359	22
3360	35
3361	30
3362	33
3363	34
3364	29
3365	33
3366	33
3367	33
3368	36
3369	33
3370	31
3371	33
3372	35
3373	32
3374	30
3375	27
3376	34
3377	37

```
3378    24
3379    27
3380    28
3381    30
3382    31
3383    44
3384    28
3385    28
3386    30
3387    30
3388    25
3389    36
3390    31
3391    27
3392    28
3393    31
3394    34
3395    32
3396    24
3397    37
3398    28
3399    31
3400    31
3401    39
3402    35
3403    34
3404    36
3405    32
3406    38
3407    36
3408    34
3409    35
3410    29
3411    26
3412    33
3413    26
3414    27
3415    30
3416    26
3417    31
3418    38
3419    26
3420    33
3421    31
3422    21
3423    19
3424    40
3425    35
3426    28
3427    32
3428    34
3429    32
3430    33
3431    33
3432    32
3433    30
3434    31
3435    29
3436    23
3437    32
```

```
3438    33
3439    35
3440    33
3441    25
3442    39
3443    29
3444    34
3445    30
3446    37
3447    35
3448    34
3449    30
3450    40
3451    29
3452    30
3453    34
3454    34
3455    27
3456    31
3457    24
3458    26
3459    30
3460    31
3461    29
3462    35
3463    21
3464    31
3465    31
3466    32
3467    33
3468    33
3469    34
3470    33
3471    37
3472    35
3473    34
3474    24
3475    35
3476    37
3477    30
3478    29
3479    35
3480    38
3481    25
3482    38
3483    37
3484    29
3485    28
3486    21
3487    31
3488    34
3489    32
3490    35
3491    41
3492    27
3493    39
3494    33
3495    33
3496    26
3497    32
```

```
3498    37
3499    27
3500    36
3501    31
3502    29
3503    33
3504    33
3505    26
3506    35
3507    33
3508    23
3509    25
3510    35
3511    39
3512    34
3513    33
3514    30
3515    35
3516    42
3517    35
3518    35
3519    32
3520    25
3521    34
3522    32
3523    30
3524    36
3525    37
3526    30
3527    34
3528    33
3529    33
3530    35
3531    25
3532    35
3533    30
3534    40
3535    32
3536    36
3537    27
3538    31
3539    32
3540    36
3541    32
3542    27
3543    35
3544     3
3545    28
3546    32
3547    36
3548    30
3549    32
3550    27
3551    30
3552    36
3553    33
3554    37
3555    26
3556    35
3557    38
```

```
3558    27
3559    35
3560    39
3561    29
3562    36
3563    32
3564    36
3565    31
3566    36
3567    36
3568    27
3569    31
3570    34
3571    32
3572    26
3573    29
3574    33
3575    35
3576    39
3577    35
3578    19
3579    29
3580    35
3581    30
3582    36
3583    34
3584    30
3585    36
dtype: int64
```

In [26]:
```
##Let's now get only those records where number of missing values are less than 10%

df_concat_z = df_concat[df_concat.isnull().sum(axis = 1) <=6]
```

In [27]:
```
#Now let's check shape
df_concat_z.shape
```

Out[27]:
```
(1460, 65)
```

In [28]:
```
#Now, let see the default counts

df_concat_z['Default'].value_counts()
```

Out[28]:
```
0    1320
1     140
Name: Default, dtype: int64
```

Now we can see that out of 387 defaults initially, now when we filtered out only those records which has at least 90% of the data is filled then we only left with 140 defaults.

In [29]:
```
df_concat['Default'].value_counts()
```

Out[29]:
```
0    3199
1     387
Name: Default, dtype: int64
```

In [30]:
```
#That means if we calculate, we will get what percentage of defaults we have covered in

140/387
```

Out[30]:
```
0.36175710594315247
```

We will only drop records with more than 25% of the data missing

In [31]:
```
###Now first lets see which are the records has more than 25% missing data

df_concat.isnull().sum().sort_values(ascending = False)/df_concat.index.size
```

Out[31]:
```
ROG_Rev_exp_in_forex_perc          0.45
ROG_Rev_earn_in_forex_perc         0.37
Cash_Flow_From_Fin                 0.28
PAT                                0.27
Adjusted_PAT                       0.27
PBT                                0.26
APATM_perc_Latest                  0.26
Cash_Flow_From_Inv                 0.24
ROG_Gross_Block_perc               0.23
CP                                 0.23
PBDT                               0.23
Cash_Flow_From_Opr                 0.22
ROG_Net_Worth_perc                 0.21
Rev_earn_in_forex                  0.21
Interest_Cover_Ratio_Latest        0.20
CPM_perc_Latest                    0.20
PBIT                               0.20
PBITM_perc_Latest                  0.20
PBDTM_perc_Latest                  0.19
Capital_exp_in_forex               0.19
Rev_exp_in_forex                   0.19
OG_Cost_of_Prod_perc               0.19
ROG_Gross_Sales_perc               0.19
PBIDT                              0.19
ROG_Net_Sales_perc                 0.19
Networth                           0.18
Market_Capitalisation              0.18
ROG_CP_perc                        0.18
ROG_PBDT_perc                      0.18
Net_Working_Capital                0.17
ROG_PBIT_perc                      0.17
ROG_PBT_perc                       0.17
OG_PBIDT_perc                      0.17
Selling_Cost                       0.17
Other_Income                       0.17
CEPS_annualised_Unit_Curr          0.17
ROG_PAT_perc                       0.17
PBIDTM_perc_Latest                 0.17
Capital_Employed                   0.17
Total_Debt                         0.16
Curr_Liab_and_Prov                 0.16
Curr_Assets                        0.16
Total_Assets_to_Liab               0.16
ROG_Capital_Employed_perc          0.16
Curr_Ratio_Latest                  0.16
Cost_of_Prod                       0.16
```

```
Value_Of_Output                    0.16
Net_Sales                          0.16
Gross_Sales                        0.15
Gross_Block                        0.15
ROG_Market_Capitalisation_perc     0.14
Fixed_Assets_Ratio_Latest          0.14
Book_Value_Adj_Unit_Curr           0.14
Book_Value_Unit_Curr               0.14
ROG_Total_Assets_perc              0.13
Value_of_Output_to_Gross_Block     0.13
Equity_Paid_Up                     0.12
Debtors_Vel_Days                   0.11
Creditors_Vel_Days                 0.11
Inventory_Ratio_Latest             0.10
Debtors_Ratio_Latest               0.10
Inventory_Vel_Days                 0.10
Total_Asset_Turnover_Ratio_Latest  0.06
Value_of_Output_to_Total_Assets    0.04
Default                            0.00
dtype: float64
```

In [32]:
```
#So after getting the above results we are removing these below features

df_concat_new = df_concat.drop(['ROG_Rev_exp_in_forex_perc','ROG_Rev_earn_in_forex_perc
                ,'Adjusted_PAT','PBT','APATM_perc_Latest'], axis = 1)
```

In [33]:
```
df_concat_new.shape
```

Out[33]:
```
(3586, 58)
```

Now we will do scaling

In [34]:
```
#### But first let's separate predictors and response

predictors = df_concat_new.drop('Default', axis = 1)
response = df_concat_new['Default']
```

In [35]:
```
from sklearn.preprocessing import StandardScaler
SS = StandardScaler()
predictors_scaled = pd.DataFrame(SS.fit_transform(predictors), columns = predictors.col
```

In [36]:
```
df_z = pd.concat([predictors_scaled, response], axis = 1)
```

In [37]:
```
df_z.head()
```

Out[37]:

	Equity_Paid_Up	Networth	Capital_Employed	Total_Debt	Gross_Block	Net_Working_Capital	Curr_Ass
0	NaN	NaN	NaN	NaN	NaN	NaN	-(
1	NaN	NaN	NaN	NaN	NaN	NaN	N
2	NaN	NaN	NaN	NaN	NaN	NaN	N

	Equity_Paid_Up	Networth	Capital_Employed	Total_Debt	Gross_Block	Net_Working_Capital	Curr_Ass
3	NaN	NaN	NaN	NaN	NaN	NaN	N
4	NaN	NaN	NaN	NaN	NaN	NaN	N

5 rows × 58 columns

In [38]:
```
## Now imputing the other remaining missing values using KNN Imputer


from sklearn.impute import KNNImputer

imputer = KNNImputer(n_neighbors = 10)
```

In [39]:
```
df_imputed = pd.DataFrame(imputer.fit_transform(df_z), columns = df_z.columns)
```

In [40]:
```
#Now checking null values

df_imputed.isnull().sum()
```

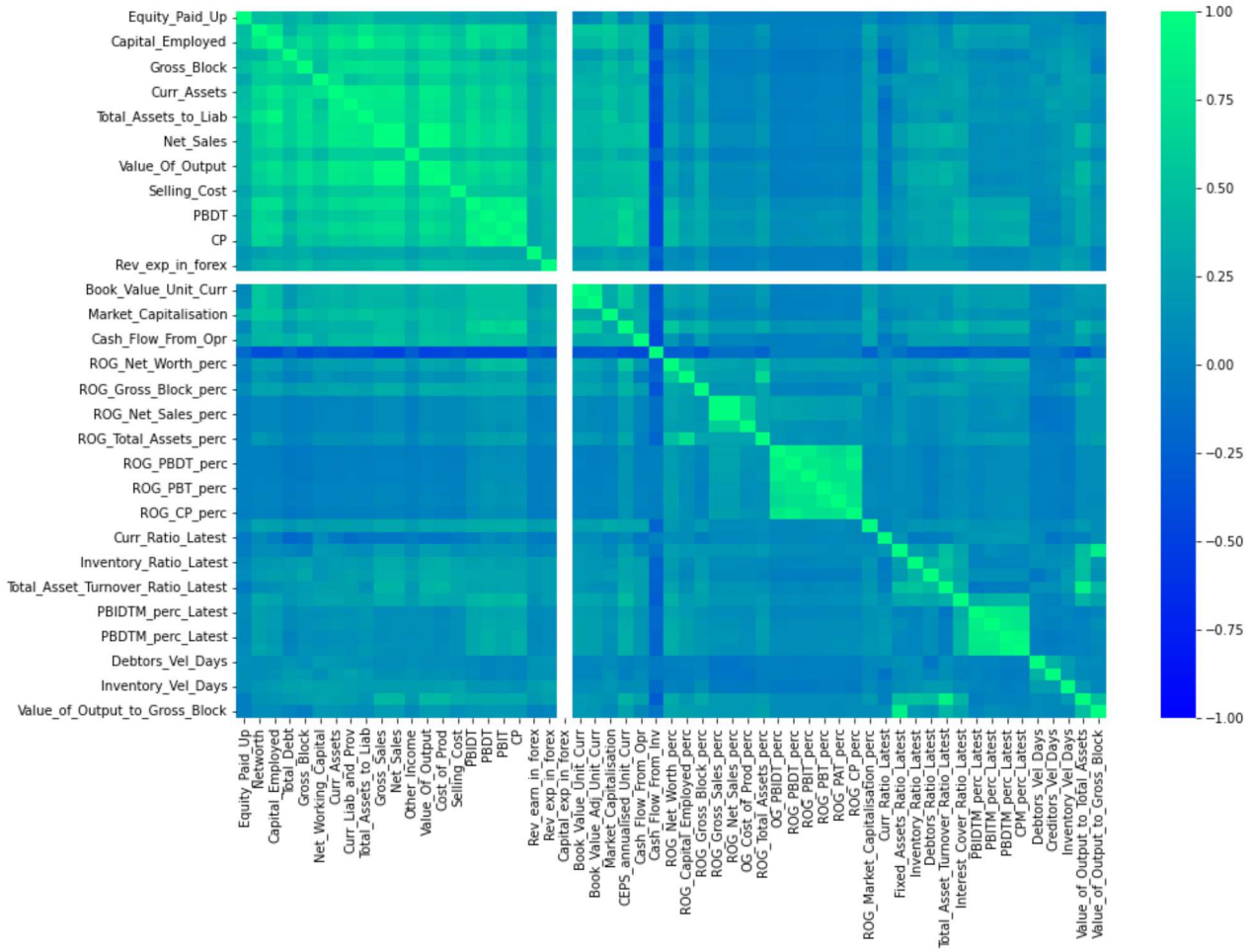
Out[40]:
```
Equity_Paid_Up                0
Networth                      0
Capital_Employed              0
Total_Debt                    0
Gross_Block                   0
Net_Working_Capital           0
Curr_Assets                   0
Curr_Liab_and_Prov            0
Total_Assets_to_Liab          0
Gross_Sales                   0
Net_Sales                     0
Other_Income                  0
Value_Of_Output               0
Cost_of_Prod                  0
Selling_Cost                  0
PBIDT                         0
PBDT                          0
PBIT                          0
CP                            0
Rev_earn_in_forex             0
Rev_exp_in_forex              0
Capital_exp_in_forex          0
Book_Value_Unit_Curr          0
Book_Value_Adj_Unit_Curr      0
Market_Capitalisation         0
CEPS_annualised_Unit_Curr     0
Cash_Flow_From_Opr            0
Cash_Flow_From_Inv            0
ROG_Net_Worth_perc            0
ROG_Capital_Employed_perc     0
ROG_Gross_Block_perc          0
ROG_Gross_Sales_perc          0
ROG_Net_Sales_perc            0
```

```
OG_Cost_of_Prod_perc                0
ROG_Total_Assets_perc               0
OG_PBIDT_perc                       0
ROG_PBDT_perc                       0
ROG_PBIT_perc                       0
ROG_PBT_perc                        0
ROG_PAT_perc                        0
ROG_CP_perc                         0
ROG_Market_Capitalisation_perc      0
Curr_Ratio_Latest                   0
Fixed_Assets_Ratio_Latest           0
Inventory_Ratio_Latest              0
Debtors_Ratio_Latest                0
Total_Asset_Turnover_Ratio_Latest   0
Interest_Cover_Ratio_Latest         0
PBIDTM_perc_Latest                  0
PBITM_perc_Latest                   0
PBDTM_perc_Latest                   0
CPM_perc_Latest                     0
Debtors_Vel_Days                    0
Creditors_Vel_Days                  0
Inventory_Vel_Days                  0
Value_of_Output_to_Total_Assets     0
Value_of_Output_to_Gross_Block      0
Default                             0
dtype: int64
```

Now after treating the outliers and missing values, lets look at the correlation

In [41]:
```
plt.figure(figsize = (15,10))
correlation = df_imputed.drop('Default', axis=1).corr()
sns.heatmap(correlation, cmap = 'winter', vmin = -1, vmax = 1)
```

Out[41]:
```
<AxesSubplot:>
```



In [42]:
```
predictors = df_imputed.drop('Default', axis = 1)
response = df_imputed['Default']
```

Now we will split the dataset into train and test set

In [43]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE #Recursive Feature Elimination
from sklearn.linear_model import LogisticRegression
```

In [44]:
```
#split should be done in 67:33 and random_state = 42 as given already

X_train, X_test, y_train, y_test = train_test_split(predictors, response, test_size = 0
```

In [86]:
```
#shape of X_train

X_train.shape
```

Out[86]: (2402, 57)

In [87]:
```
#shape of X_test

X_test.shape
```

Out[87]: (1184, 57)

In [45]:
```
#After splitting we are using Logistic Regression as a model with recursive feature eli


LR = LogisticRegression()
selector = RFE(estimator = LR, n_features_to_select = 20, step = 1)

selector = selector.fit(X_train, y_train)

selector.n_features_
```

Out[45]: 20

In [46]:
```
#Now let see ranking

selector.ranking_
```

Out[46]:
```
array([25,  1,  1,  2,  1,  7,  1,  1,  1,  1, 21,  6,  1,  1, 18,  1,  1,
        1,  1, 33, 10, 38,  1,  1,  9, 34, 35, 17,  1,  1, 29, 14, 15, 16,
        1, 36, 31, 23, 26, 22, 37, 32,  1,  3, 12, 27,  5,  1, 20, 13, 28,
       19, 24, 11,  8,  4, 30])
```

In [47]:
```
#Now let see which are those 20 best features

df_new = pd.DataFrame({'Feature':predictors_scaled.columns, 'Rank':selector.ranking_})
```

In [48]:
```
df_new[df_new['Rank']==1] #features at Rank 1
```

Out[48]:

	Feature	Rank
1	Networth	1
2	Capital_Employed	1
4	Gross_Block	1
6	Curr_Assets	1
7	Curr_Liab_and_Prov	1
8	Total_Assets_to_Liab	1
9	Gross_Sales	1
12	Value_Of_Output	1
13	Cost_of_Prod	1
15	PBIDT	1
16	PBDT	1
17	PBIT	1
18	CP	1
22	Book_Value_Unit_Curr	1

	Feature	Rank
23	Book_Value_Adj_Unit_Curr	1
28	ROG_Net_Worth_perc	1
29	ROG_Capital_Employed_perc	1
34	ROG_Total_Assets_perc	1
42	Curr_Ratio_Latest	1
47	Interest_Cover_Ratio_Latest	1

In [49]:
```
#### Validating the model on test and train
```

In [50]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

In [51]:
```
train_predictor = selector.predict(X_train)
test_predictor = selector.predict(X_test)
```

In [67]:
```
print(confusion_matrix(y_train, train_predictor))
```

```
[[2137   20]
 [  96  149]]
```

In [53]:
```
print(classification_report(y_train, train_predictor))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.99      0.97      2157
         1.0       0.88      0.61      0.72       245

    accuracy                           0.95      2402
   macro avg       0.92      0.80      0.85      2402
weighted avg       0.95      0.95      0.95      2402
```

In [54]:
```
print(confusion_matrix(y_test, test_predictor))
```

```
[[1028   14]
 [  48   94]]
```

In [55]:
```
print(classification_report(y_test, test_predictor))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.99      0.97      1042
         1.0       0.87      0.66      0.75       142

    accuracy                           0.95      1184
   macro avg       0.91      0.82      0.86      1184
```

```
weighted avg       0.95      0.95      0.94      1184
```

The recall score is not so good. Maybe the data is still imbalance. We will balance the data using SMOTE

In [56]:
```
from imblearn.over_sampling import SMOTE
SM = SMOTE(random_state=33)
X_res, y_res = SM.fit_resample(X_train, y_train)
```

In [57]:
```
selector_smote = selector.fit(X_res, y_res)
```

In [58]:
```
selector_smote.n_features_
```

Out[58]:
```
20
```

In [59]:
```
train_predictor_smote = selector_smote.predict(X_res)
test_predictor_smote = selector_smote.predict(X_test)
```

In [60]:
```
print(classification_report(y_res, train_predictor_smote))
```

```
              precision    recall  f1-score   support

         0.0       0.93      0.91      0.92      2157
         1.0       0.91      0.93      0.92      2157

    accuracy                           0.92      4314
   macro avg       0.92      0.92      0.92      4314
weighted avg       0.92      0.92      0.92      4314
```

In [61]:
```
print(classification_report(y_test, test_predictor_smote))
```

```
              precision    recall  f1-score   support

         0.0       0.99      0.87      0.93      1042
         1.0       0.50      0.94      0.65       142

    accuracy                           0.88      1184
   macro avg       0.74      0.91      0.79      1184
weighted avg       0.93      0.88      0.89      1184
```

Now our recall has been improved so much for both test and train.

CONCLUSION:

1. We got better recall of 94% for 1, that means 94% of the actual value of defaults are the right prediction.
2. Precision of 50% means 50% of the predicted defaults are actual.

__END__