Bankruptcy prediction

Introduction

The dataset here presented shows several financial ratios of Taiwanese firm between 1999 and 2009 (here the dataset source from kaggle) , as feature for our model. The bankruptcy follow the definition from the Taiwan Stock Exchange.
We will first do some PCA to identify a limited number of feature for our model, the label we try to predict is the first column "Bankrupt?".

Data exploration and PCA

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import itertools
import seaborn as sns
```

In [2]:
```
df = pd.read_csv("data.csv")
df.dropna(axis=0,inplace=True)
display(df)
```

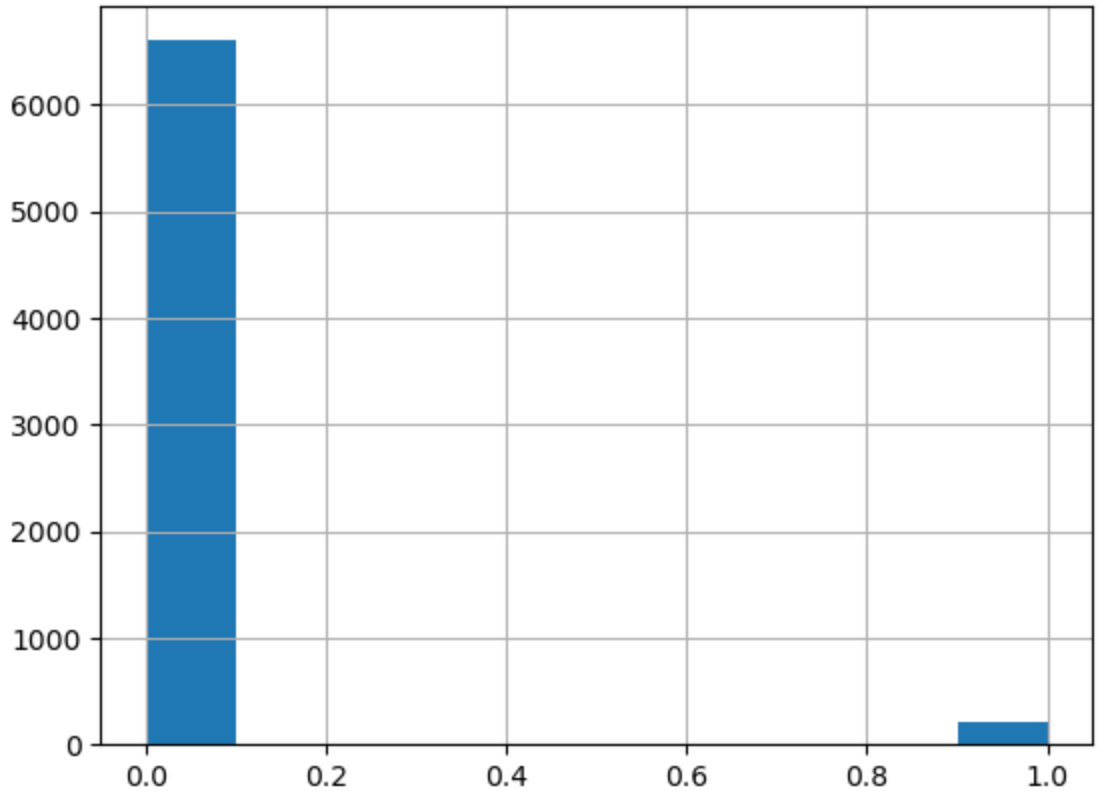
	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-t r Intere Ra
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887	0.8088
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380	0.8093
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403	0.8083
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967	0.8089
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366	0.8093
...	...	...	...	...	...	...	...	...	
6814	0	0.493687	0.539468	0.543230	0.604455	0.604462	0.998992	0.797409	0.8093
6815	0	0.475162	0.538269	0.524172	0.598308	0.598308	0.998992	0.797414	0.8093
6816	0	0.472725	0.533744	0.520638	0.610444	0.610213	0.998984	0.797401	0.8093
6817	0	0.506264	0.559911	0.554045	0.607850	0.607850	0.999074	0.797500	0.8093
6818	0	0.493053	0.570105	0.549548	0.627409	0.627409	0.998080	0.801987	0.8138

6819 rows × 96 columns

In [3]:
```
print(df[df.columns[0]].value_counts())
df[df.columns[0]].hist()
plt.show()
```

```
0    6599
1     220
```

```
Name: Bankrupt?, dtype: int64
```



The data available are largely skewd to not bankrupt account we will handle this problem when building the pipeline. Considering all the 96 feature we have a gran total of 4560 correlation plot. We will first exclude the highly correlated feature and then show the correlation plot.

In [4]:
```
print(df.info())
display(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6819 entries, 0 to 6818
Data columns (total 96 columns):
 #   Column                                                    Non-Null Count  Dtype  
---  ------                                                    --------------  -----  
 0   Bankrupt?                                                 6819 non-null   int64  
 1    ROA(C) before interest and depreciation before interest  6819 non-null   float64
 2    ROA(A) before interest and % after tax                   6819 non-null   float64
 3    ROA(B) before interest and depreciation after tax        6819 non-null   float64
 4    Operating Gross Margin                                   6819 non-null   float64
 5    Realized Sales Gross Margin                              6819 non-null   float64
 6    Operating Profit Rate                                    6819 non-null   float64
 7    Pre-tax net Interest Rate                                6819 non-null   float64
 8    After-tax net Interest Rate                              6819 non-null   float64
 9    Non-industry income and expenditure/revenue              6819 non-null   float64
 10   Continuous interest rate (after tax)                     6819 non-null   float64
 11   Operating Expense Rate                                   6819 non-null   float64
 12   Research and development expense rate                    6819 non-null   float64
 13   Cash flow rate                                           6819 non-null   float64
 14   Interest-bearing debt interest rate                      6819 non-null   float64
 15   Tax rate (A)                                             6819 non-null   float64
 16   Net Value Per Share (B)                                  6819 non-null   float64
 17   Net Value Per Share (A)                                  6819 non-null   float64
 18   Net Value Per Share (C)                                  6819 non-null   float64
 19   Persistent EPS in the Last Four Seasons                  6819 non-null   float64
 20   Cash Flow Per Share                                      6819 non-null   float64
 21   Revenue Per Share (Yuan ¥)                               6819 non-null   float64
 22   Operating Profit Per Share (Yuan ¥)                      6819 non-null   float64
 23   Per Share Net profit before tax (Yuan ¥)                 6819 non-null   float64
 24   Realized Sales Gross Profit Growth Rate                  6819 non-null   float64
```

25	Operating Profit Growth Rate	6819	non-null	float64
26	After-tax Net Profit Growth Rate	6819	non-null	float64
27	Regular Net Profit Growth Rate	6819	non-null	float64
28	Continuous Net Profit Growth Rate	6819	non-null	float64
29	Total Asset Growth Rate	6819	non-null	float64
30	Net Value Growth Rate	6819	non-null	float64
31	Total Asset Return Growth Rate Ratio	6819	non-null	float64
32	Cash Reinvestment %	6819	non-null	float64
33	Current Ratio	6819	non-null	float64
34	Quick Ratio	6819	non-null	float64
35	Interest Expense Ratio	6819	non-null	float64
36	Total debt/Total net worth	6819	non-null	float64
37	Debt ratio %	6819	non-null	float64
38	Net worth/Assets	6819	non-null	float64
39	Long-term fund suitability ratio (A)	6819	non-null	float64
40	Borrowing dependency	6819	non-null	float64
41	Contingent liabilities/Net worth	6819	non-null	float64
42	Operating profit/Paid-in capital	6819	non-null	float64
43	Net profit before tax/Paid-in capital	6819	non-null	float64
44	Inventory and accounts receivable/Net value	6819	non-null	float64
45	Total Asset Turnover	6819	non-null	float64
46	Accounts Receivable Turnover	6819	non-null	float64
47	Average Collection Days	6819	non-null	float64
48	Inventory Turnover Rate (times)	6819	non-null	float64
49	Fixed Assets Turnover Frequency	6819	non-null	float64
50	Net Worth Turnover Rate (times)	6819	non-null	float64
51	Revenue per person	6819	non-null	float64
52	Operating profit per person	6819	non-null	float64
53	Allocation rate per person	6819	non-null	float64
54	Working Capital to Total Assets	6819	non-null	float64
55	Quick Assets/Total Assets	6819	non-null	float64
56	Current Assets/Total Assets	6819	non-null	float64
57	Cash/Total Assets	6819	non-null	float64
58	Quick Assets/Current Liability	6819	non-null	float64
59	Cash/Current Liability	6819	non-null	float64
60	Current Liability to Assets	6819	non-null	float64
61	Operating Funds to Liability	6819	non-null	float64
62	Inventory/Working Capital	6819	non-null	float64
63	Inventory/Current Liability	6819	non-null	float64
64	Current Liabilities/Liability	6819	non-null	float64
65	Working Capital/Equity	6819	non-null	float64
66	Current Liabilities/Equity	6819	non-null	float64
67	Long-term Liability to Current Assets	6819	non-null	float64
68	Retained Earnings to Total Assets	6819	non-null	float64
69	Total income/Total expense	6819	non-null	float64
70	Total expense/Assets	6819	non-null	float64
71	Current Asset Turnover Rate	6819	non-null	float64
72	Quick Asset Turnover Rate	6819	non-null	float64
73	Working capitcal Turnover Rate	6819	non-null	float64
74	Cash Turnover Rate	6819	non-null	float64
75	Cash Flow to Sales	6819	non-null	float64
76	Fixed Assets to Assets	6819	non-null	float64
77	Current Liability to Liability	6819	non-null	float64
78	Current Liability to Equity	6819	non-null	float64
79	Equity to Long-term Liability	6819	non-null	float64
80	Cash Flow to Total Assets	6819	non-null	float64
81	Cash Flow to Liability	6819	non-null	float64
82	CFO to Assets	6819	non-null	float64
83	Cash Flow to Equity	6819	non-null	float64
84	Current Liability to Current Assets	6819	non-null	float64
85	Liability-Assets Flag	6819	non-null	int64
86	Net Income to Total Assets	6819	non-null	float64
87	Total assets to GNP price	6819	non-null	float64
88	No-credit Interval	6819	non-null	float64
89	Gross Profit to Sales	6819	non-null	float64
90	Net Income to Stockholder's Equity	6819	non-null	float64

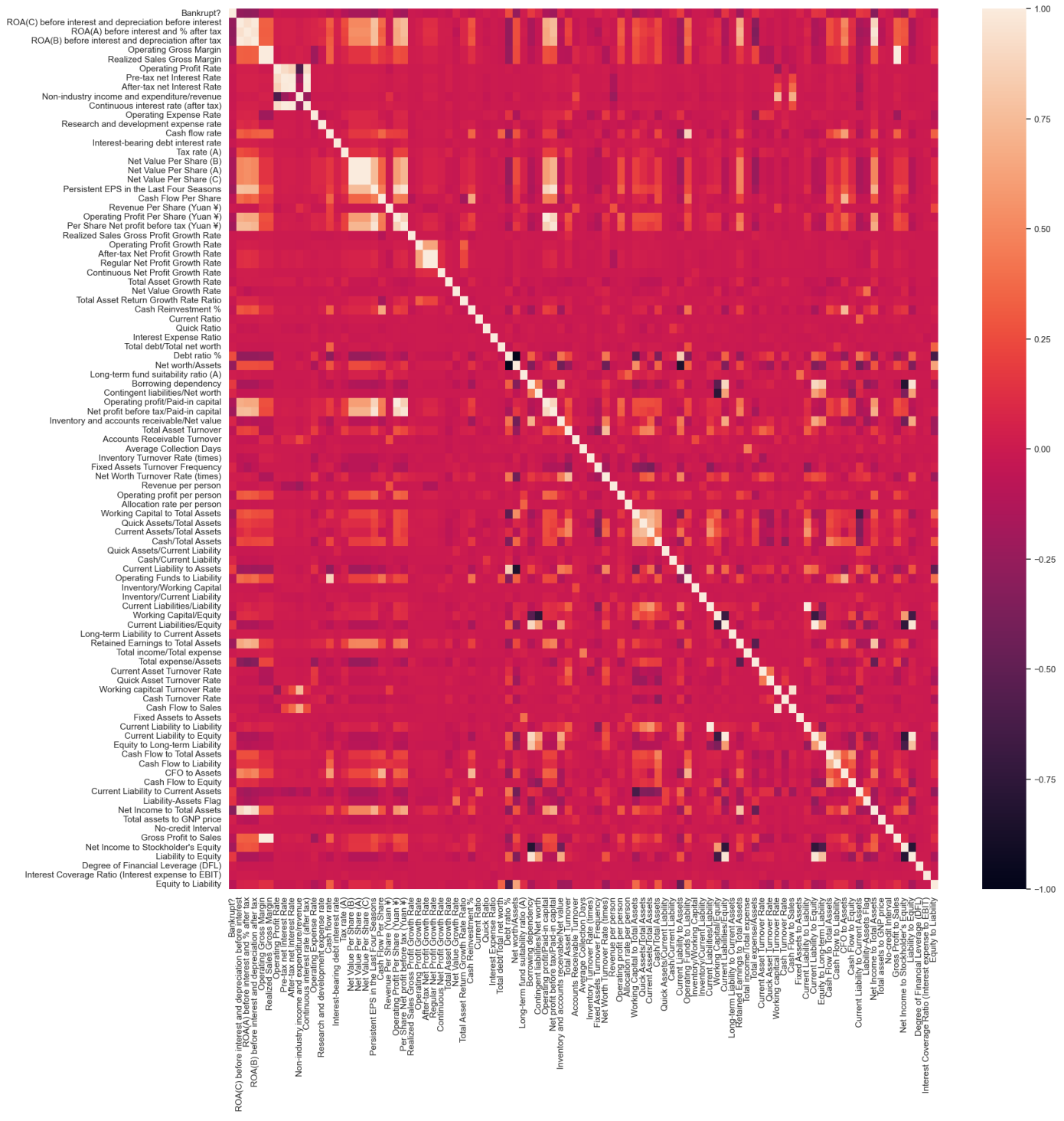
```
 91   Liability to Equity                                    6819 non-null   float64
 92   Degree of Financial Leverage (DFL)                     6819 non-null   float64
 93   Interest Coverage Ratio (Interest expense to EBIT)     6819 non-null   float64
 94   Net Income Flag                                        6819 non-null   int64
 95   Equity to Liability                                    6819 non-null   float64
dtypes: float64(93), int64(3)
memory usage: 5.0 MB
None
```

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	P
count	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	68
mean	0.032263	0.505180	0.558625	0.553589	0.607948	0.607929	0.998755	
std	0.176710	0.060686	0.065620	0.061595	0.016934	0.016916	0.013010	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.476527	0.535543	0.527277	0.600445	0.600434	0.998969	
50%	0.000000	0.502706	0.559802	0.552278	0.605997	0.605976	0.999022	
75%	0.000000	0.535563	0.589157	0.584105	0.613914	0.613842	0.999095	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 96 columns

After the exploratory analysis we focus on the correlation of the features. We do not need redundant variables so we will drop the columns with correlation higher than 0.8 or smaller than -0.8. We will also drop the column "Net Income Flag" since it is a constant with value 1.

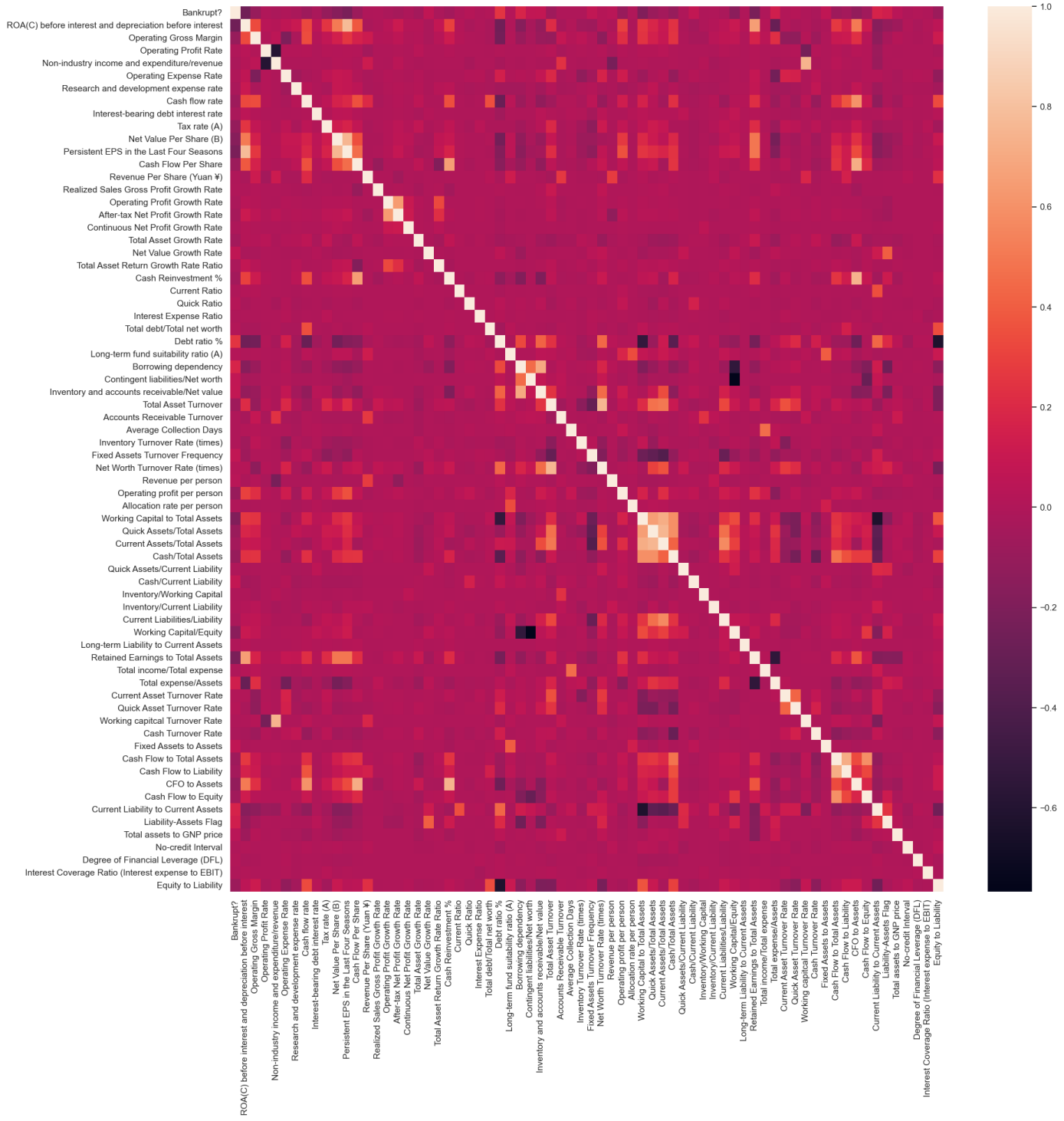
In [5]:
```
df = df.drop(df.columns[-2],axis=1)

def cor_heatmap(df):
  corr = df.corr()
  sns.set(rc={'figure.figsize':(20,20)})
  sns.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns)

cor_heatmap(df)
```



In [6]:
```
def drop_corr(x=0.8):
  upper_tri =corr.where(np.triu(np.ones(corr.shape),k=1).astype(bool))
  to_drop = [column for column in upper_tri.columns if any(upper_tri[column].abs() > x)]
  print(to_drop)
  new_df = df.drop(to_drop,axis=1)
  return new_df

corr = df.corr()
n_df = drop_corr(.8)
cor_heatmap(n_df)
```

```
[' ROA(A) before interest and % after tax', ' ROA(B) before interest and depreciation af
ter tax', ' Realized Sales Gross Margin', ' Pre-tax net Interest Rate', ' After-tax net
Interest Rate', ' Continuous interest rate (after tax)', ' Net Value Per Share (A)', ' N
et Value Per Share (C)', ' Operating Profit Per Share (Yuan ¥)', ' Per Share Net profit
before tax (Yuan ¥)', ' Regular Net Profit Growth Rate', ' Net worth/Assets', ' Operatin
g profit/Paid-in capital', ' Net profit before tax/Paid-in capital', ' Current Liability
to Assets', ' Operating Funds to Liability', ' Current Liabilities/Equity', ' Cash Flow
to Sales', ' Current Liability to Liability', ' Current Liability to Equity', ' Equity t
```

```
o Long-term Liability', ' Net Income to Total Assets', ' Gross Profit to Sales', " Net I
ncome to Stockholder's Equity", ' Liability to Equity']
```



In [7]:
```
display(n_df)
```

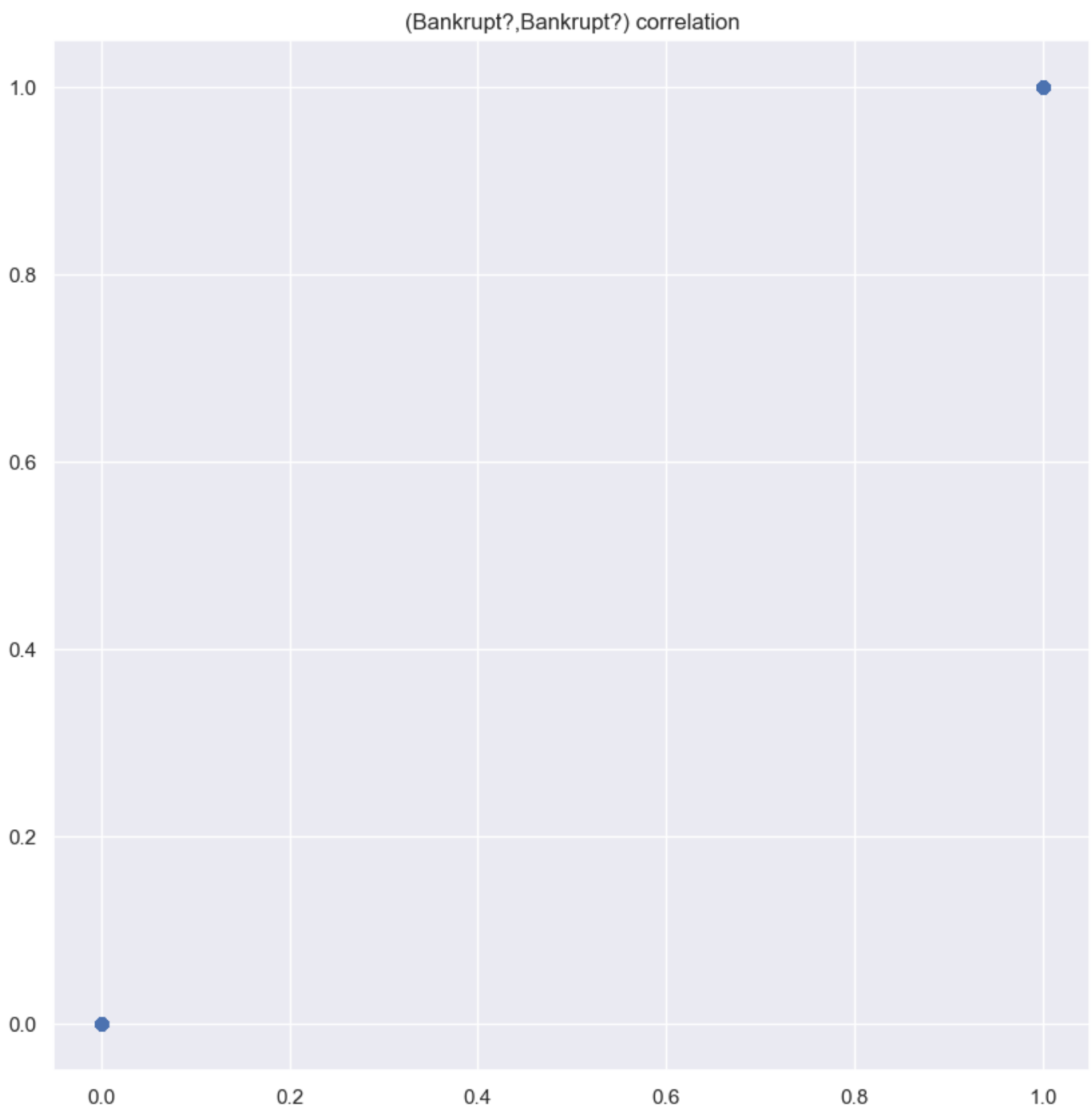
	Bankrupt?	ROA(C) before interest and depreciation before interest	Operating Gross Margin	Operating Profit Rate	Non-industry income and expenditure/revenue	Operating Expense Rate	Research and development expense rate
0	1	0.370594	0.601457	0.998969	0.302646	1.256969e-04	0.000000e+00
1	1	0.464291	0.610235	0.998946	0.303556	2.897851e-04	0.000000e+00
2	1	0.426071	0.601450	0.998857	0.302035	2.361297e-04	2.550000e+07
3	1	0.399844	0.583541	0.998700	0.303350	1.078888e-04	0.000000e+00
4	1	0.465022	0.598783	0.998973	0.303475	7.890000e+09	0.000000e+00

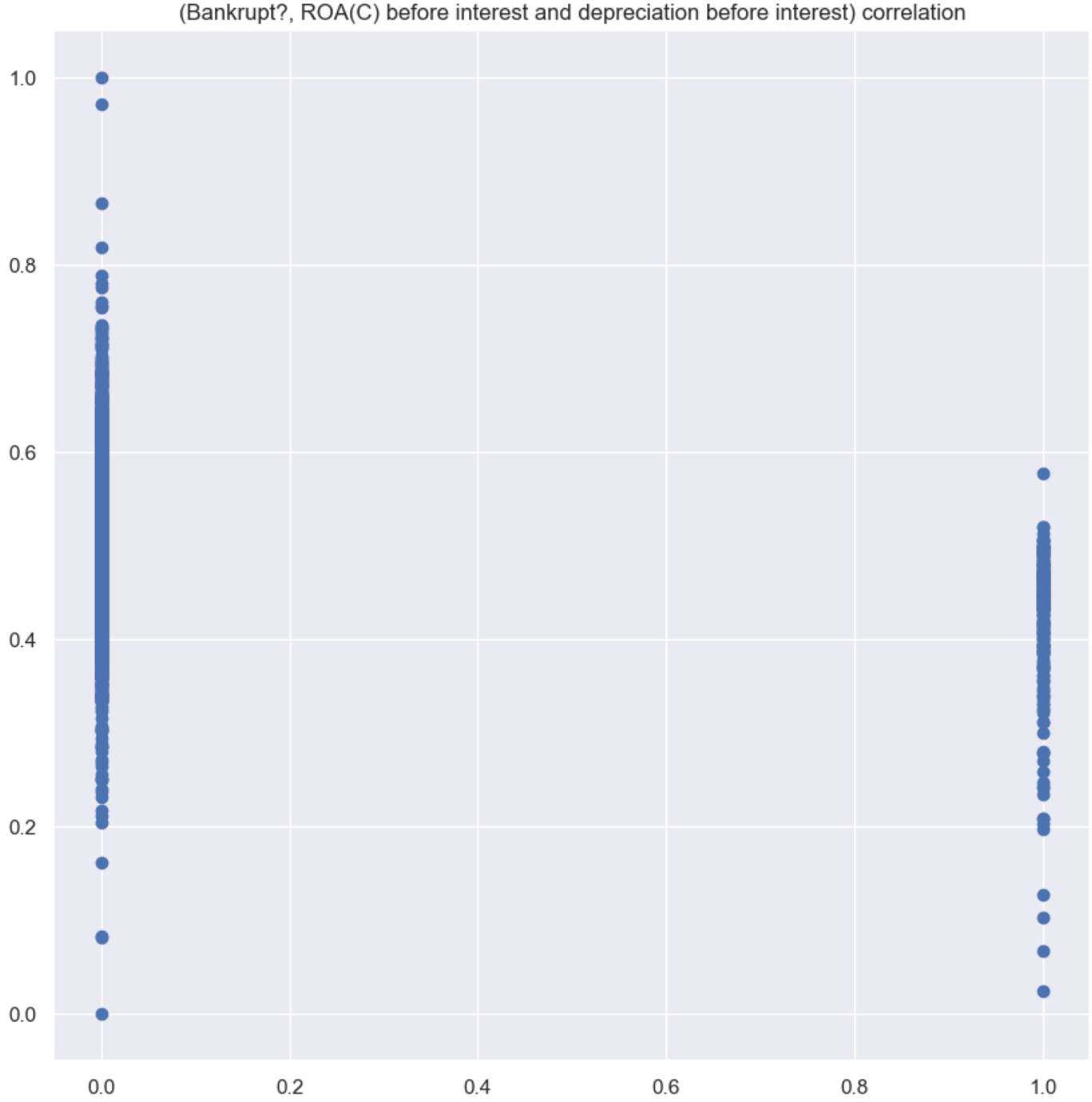
...	...	...	...	...	...	...	...
6814	0	0.493687	0.604455	0.998992	0.303510	1.510213e-04	4.500000e+09
6815	0	0.475162	0.598308	0.998992	0.303520	5.220000e+09	1.440000e+09
6816	0	0.472725	0.610444	0.998984	0.303512	2.509312e-04	1.039086e-04
6817	0	0.506264	0.607850	0.999074	0.303498	1.236154e-04	2.510000e+09
6818	0	0.493053	0.627409	0.998080	0.313415	1.431695e-03	0.000000e+00

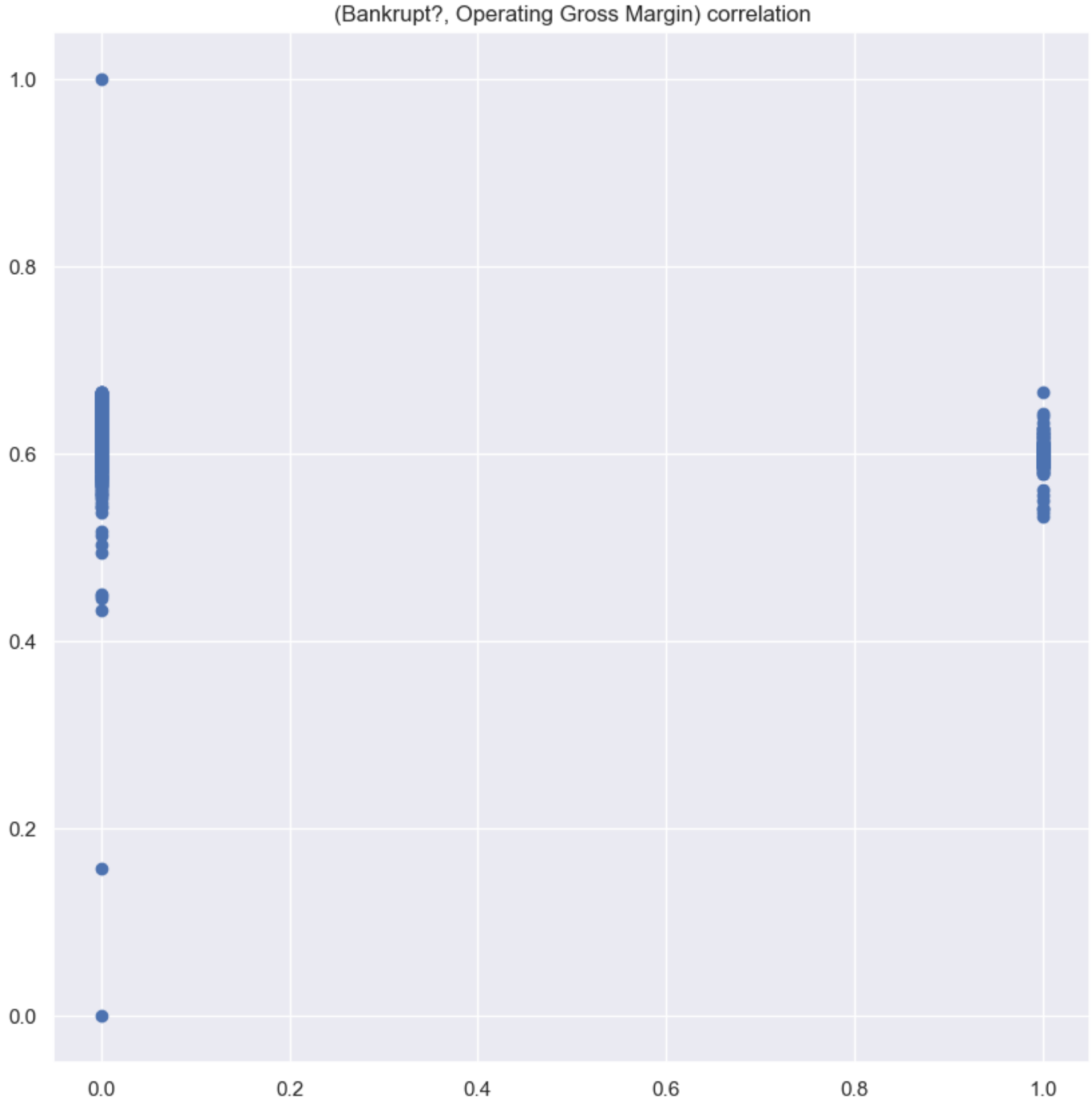
6819 rows × 70 columns

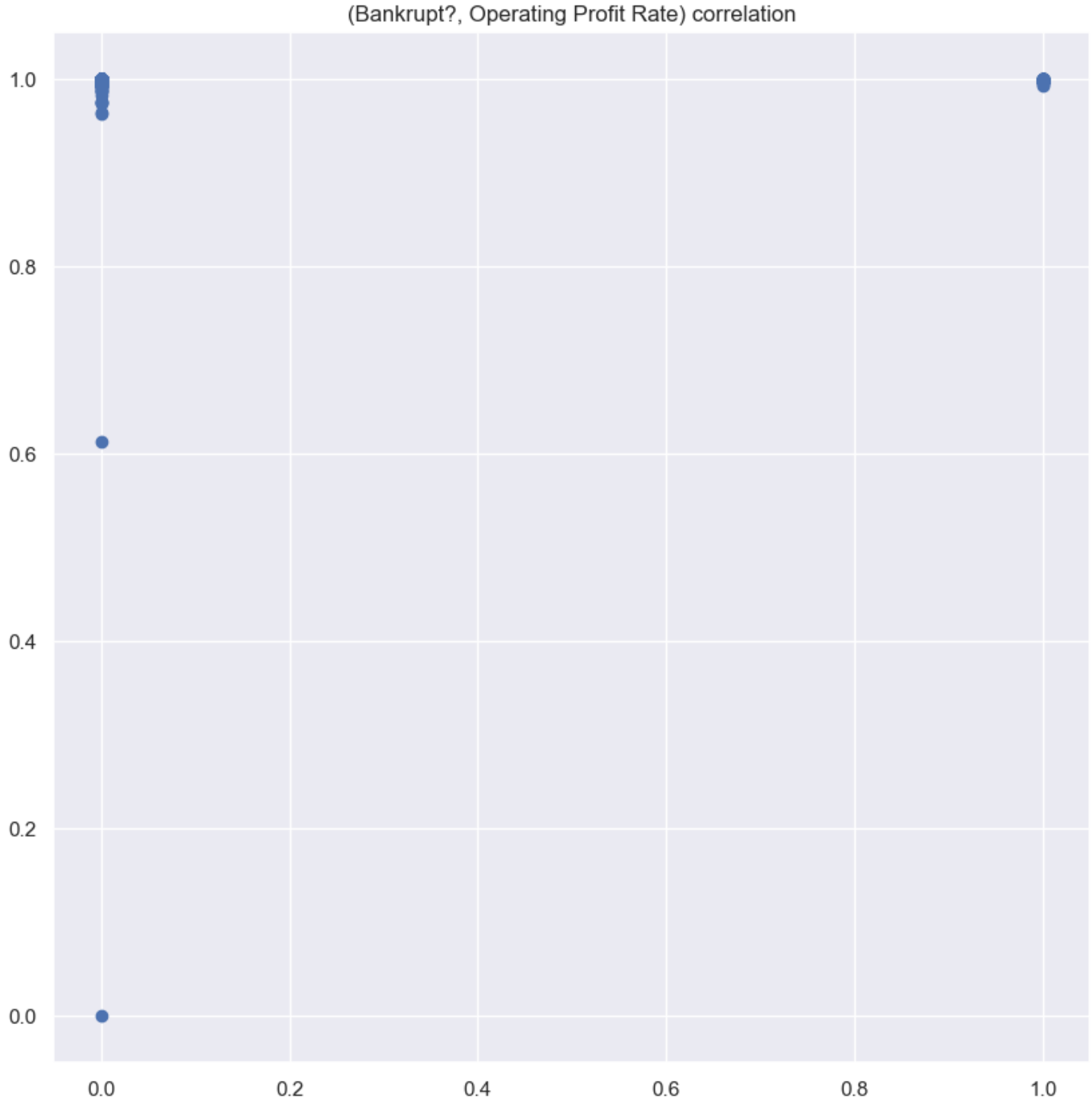
We have dropped 26 columns, now we can proceed with feature projection using PCA mehtod of scikit learn library. In order to explore the most important feature we are gonna plot the correlation between the label to predict and the other columns

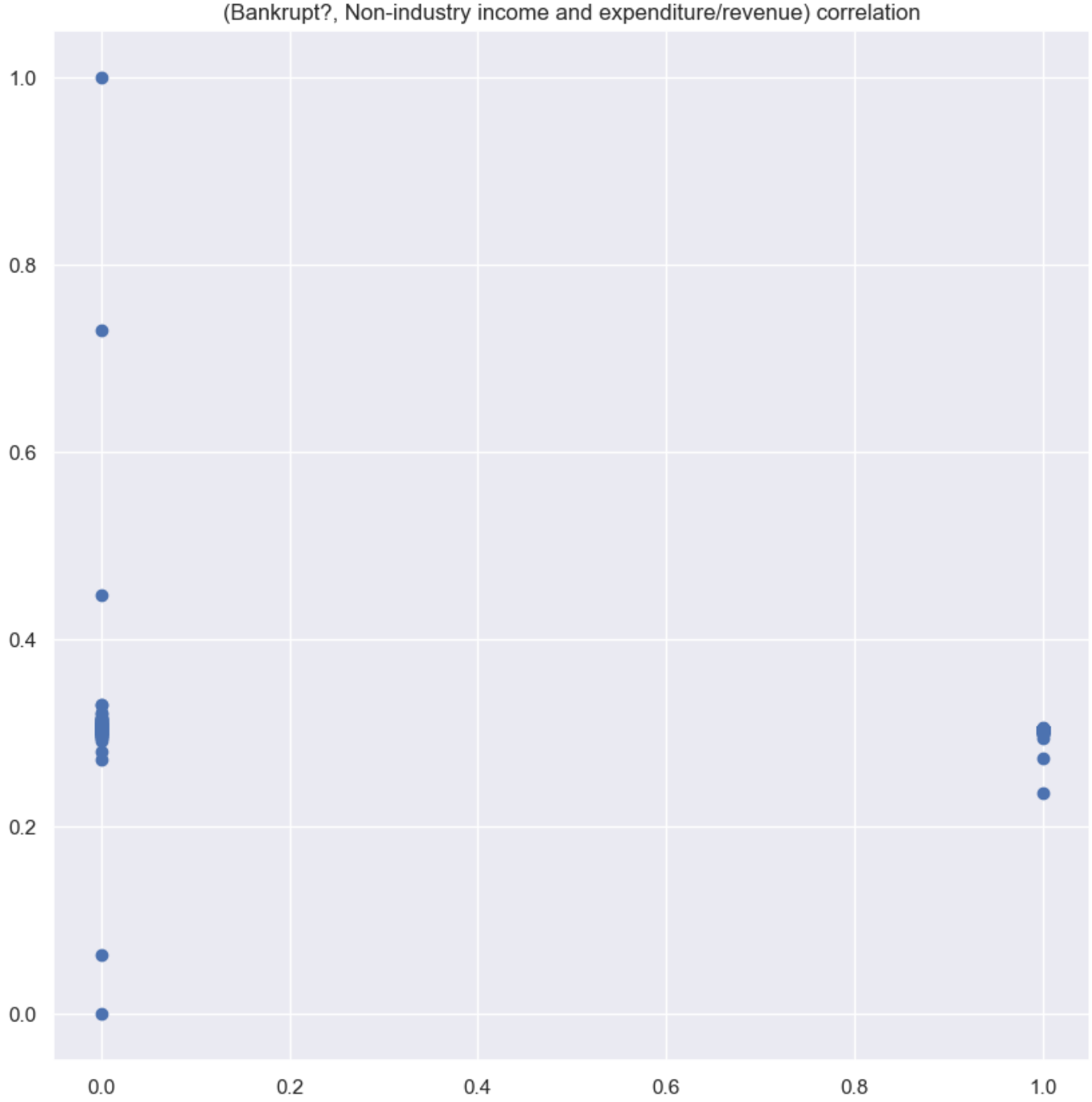
In [8]:
```
%matplotlib inline

def scatter_(c1,c2,name=True):
  plt.figure(figsize=(10,10))
  plt.scatter(c1,c2)
  if name==True:
    plt.title(f"({c1.name},{c2.name}) correlation")
  plt.show()
  rho = np.corrcoef(c1,c2)


corr_dict = {}
column_pair = [ (n_df.columns[0],i) for i in n_df.columns]

for c in column_pair:
    corr_dict[str(c)] = scatter_(df[c[0]],df[c[1]])

print(corr_dict)
```

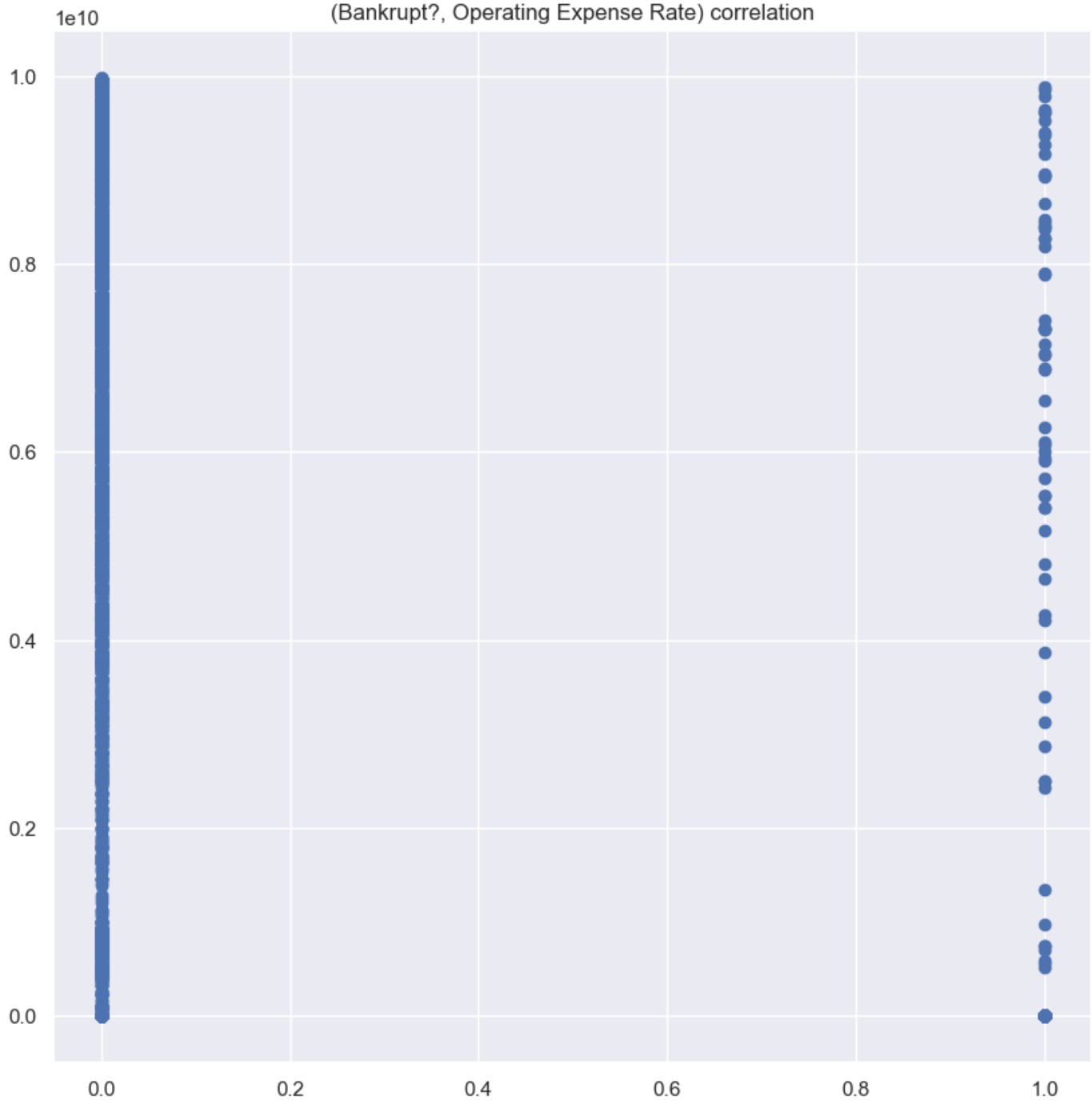

(Bankrupt?,Bankrupt?) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

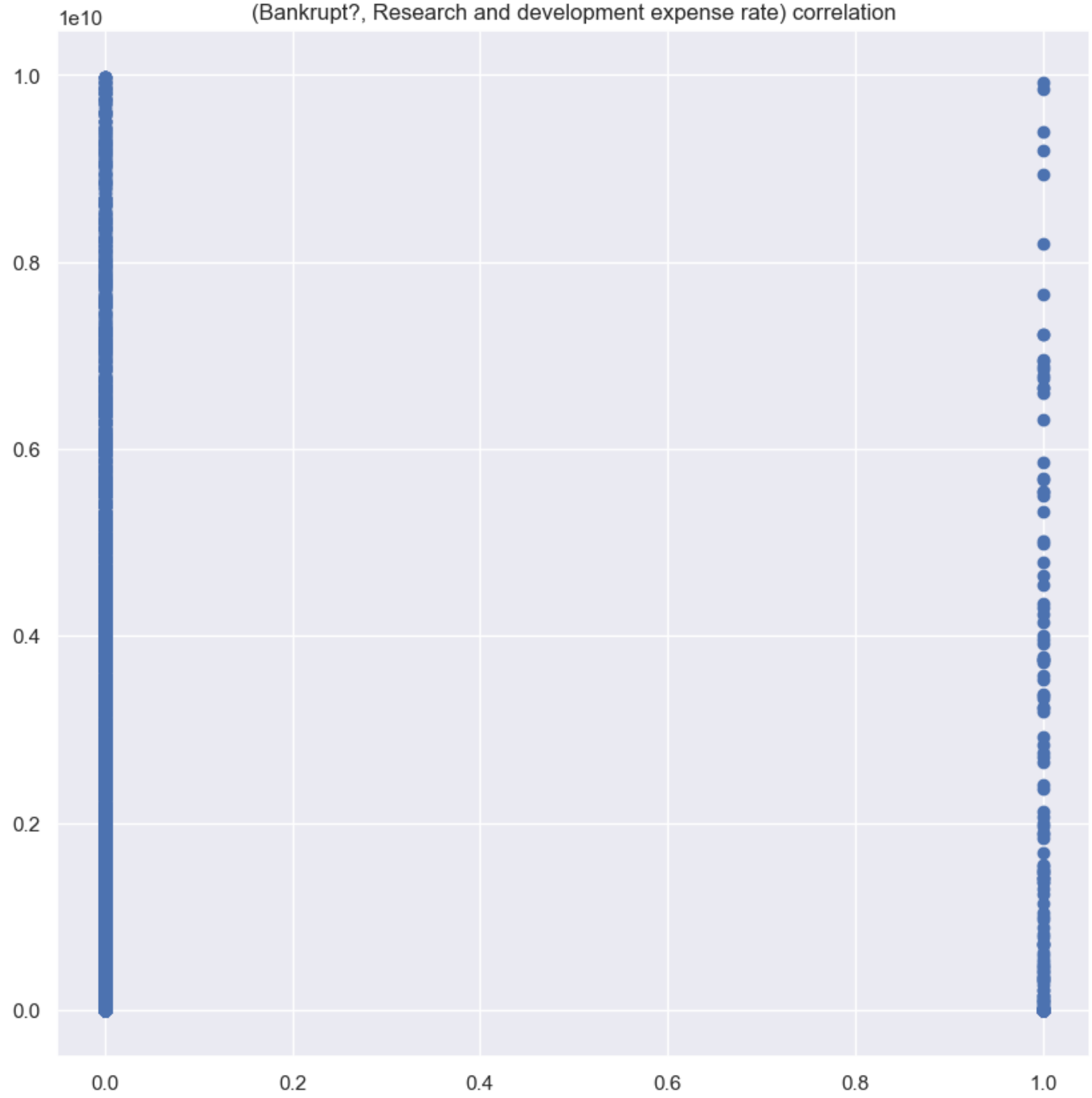

(Bankrupt?, ROA(C) before interest and depreciation before interest) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

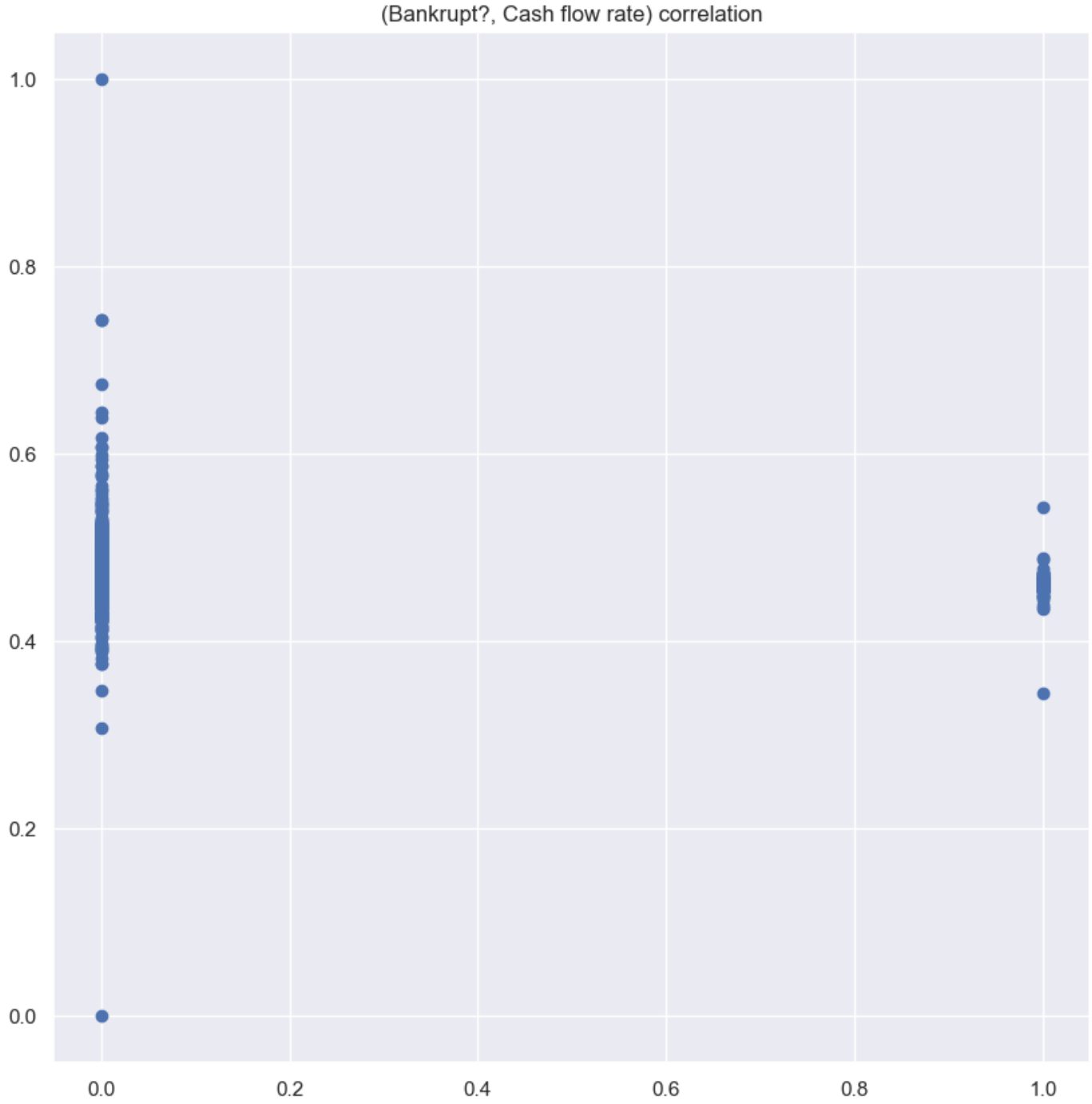

(Bankrupt?, Operating Gross Margin) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

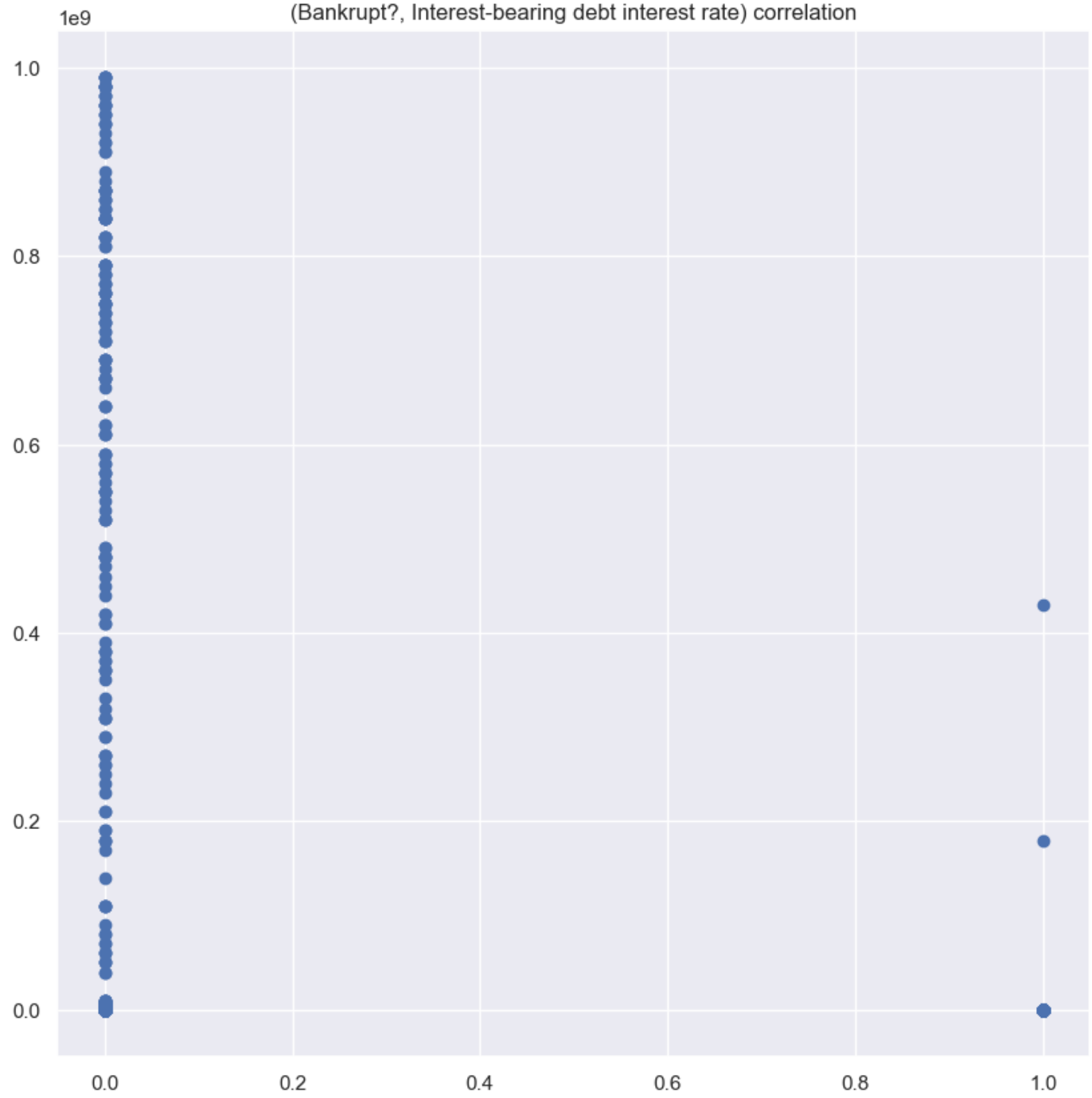

(Bankrupt?, Operating Profit Rate) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

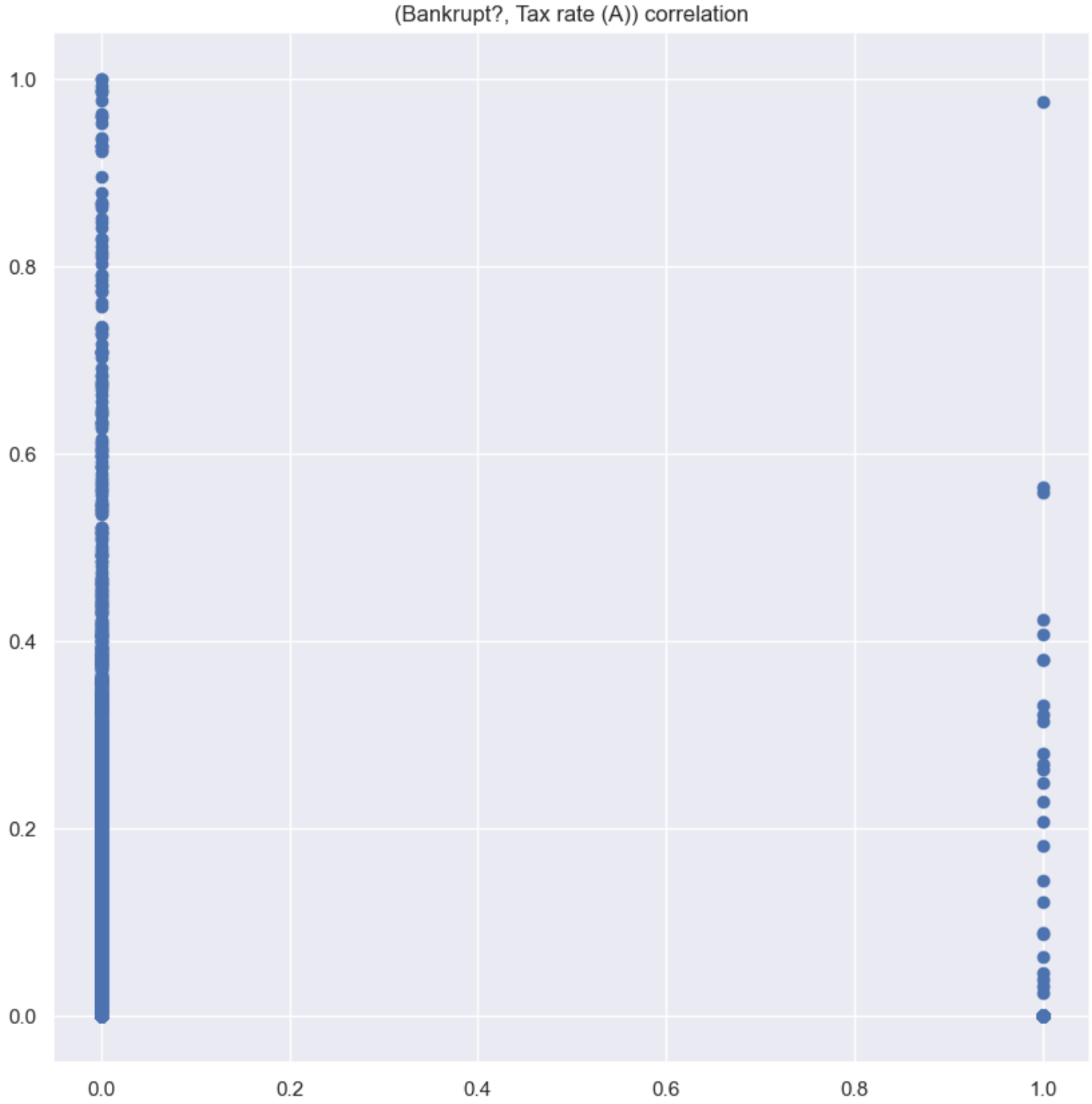

(Bankrupt?, Non-industry income and expenditure/revenue) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

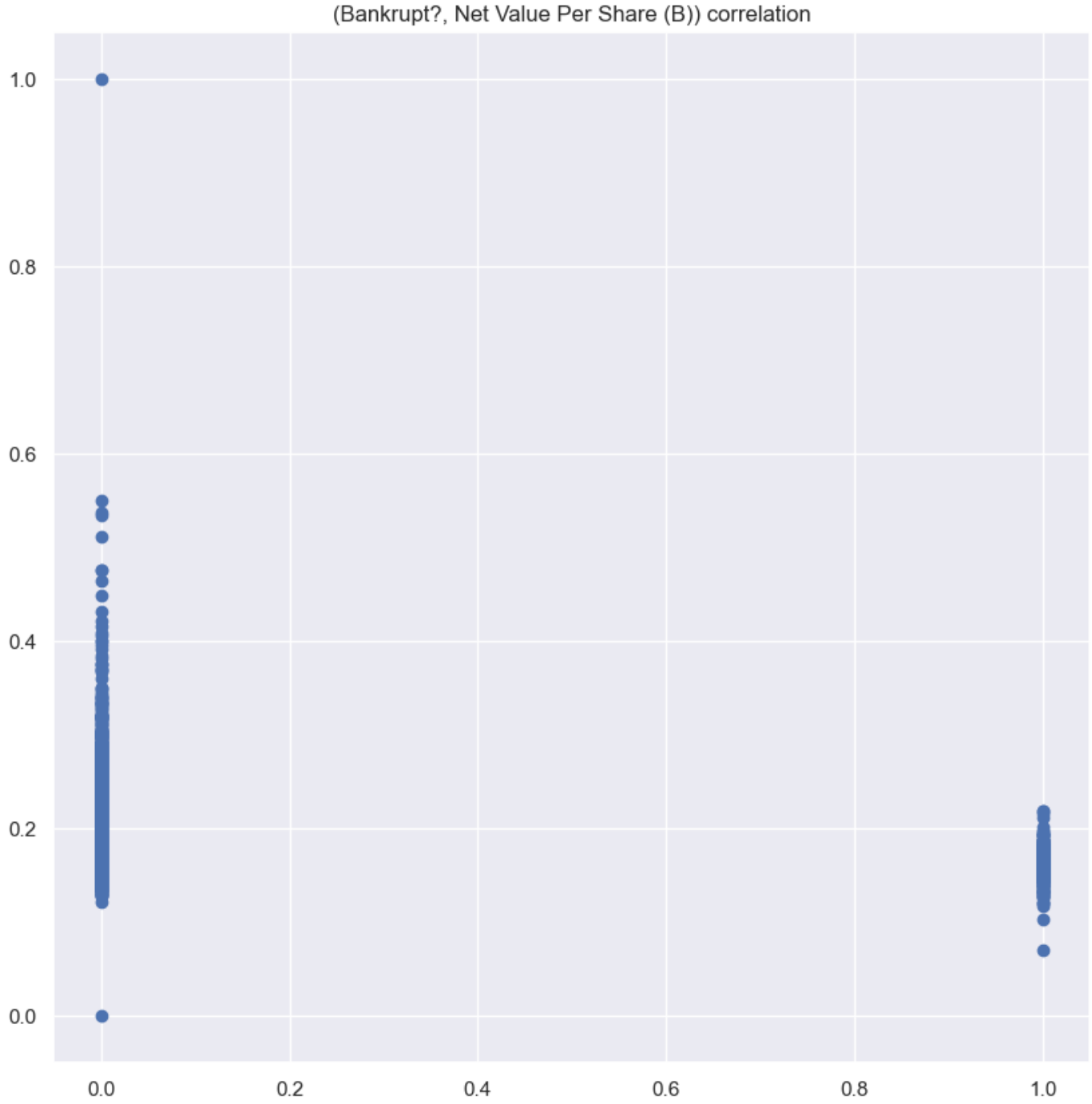

(Bankrupt?, Operating Expense Rate) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

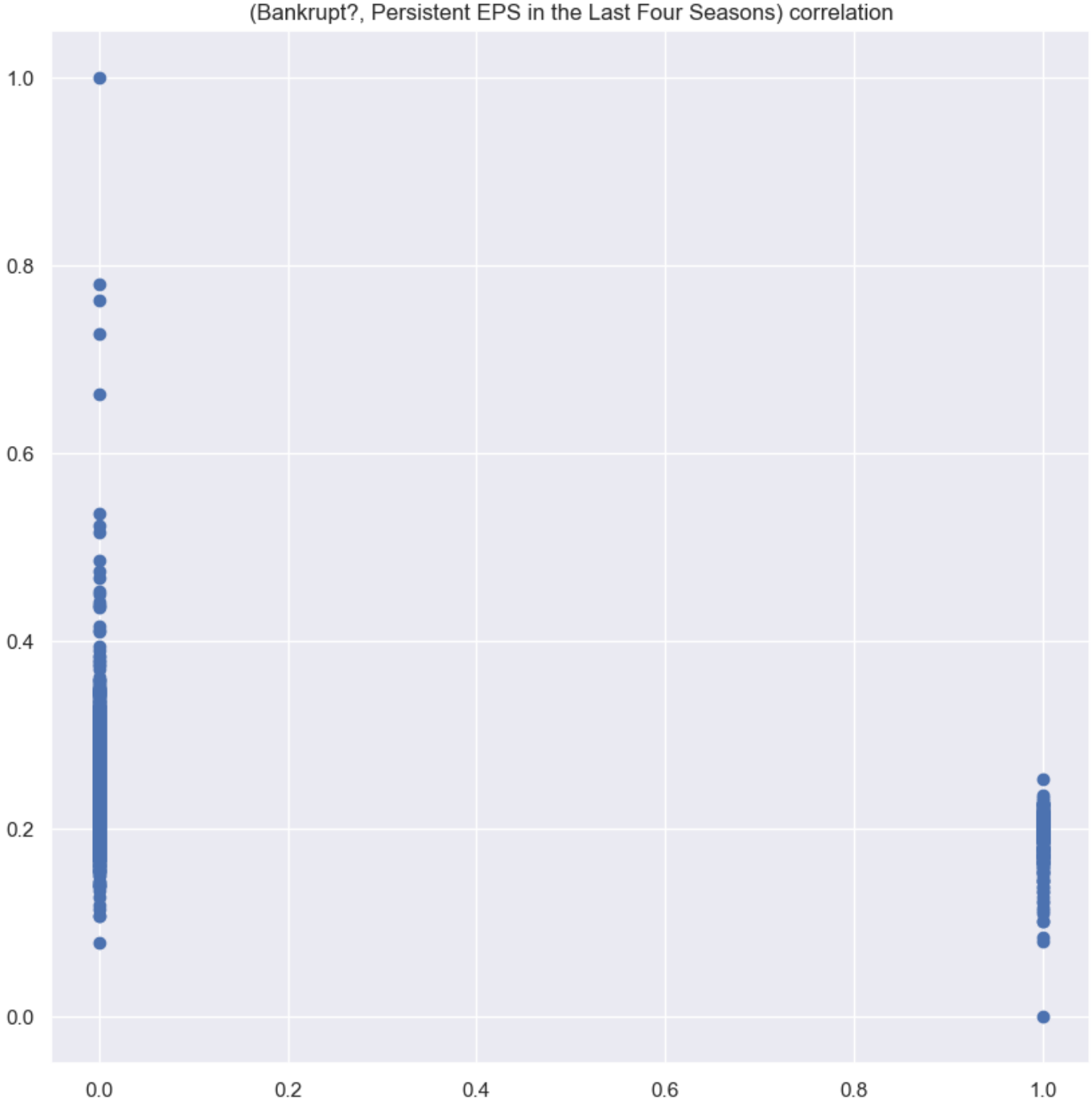

(Bankrupt?, Research and development expense rate) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

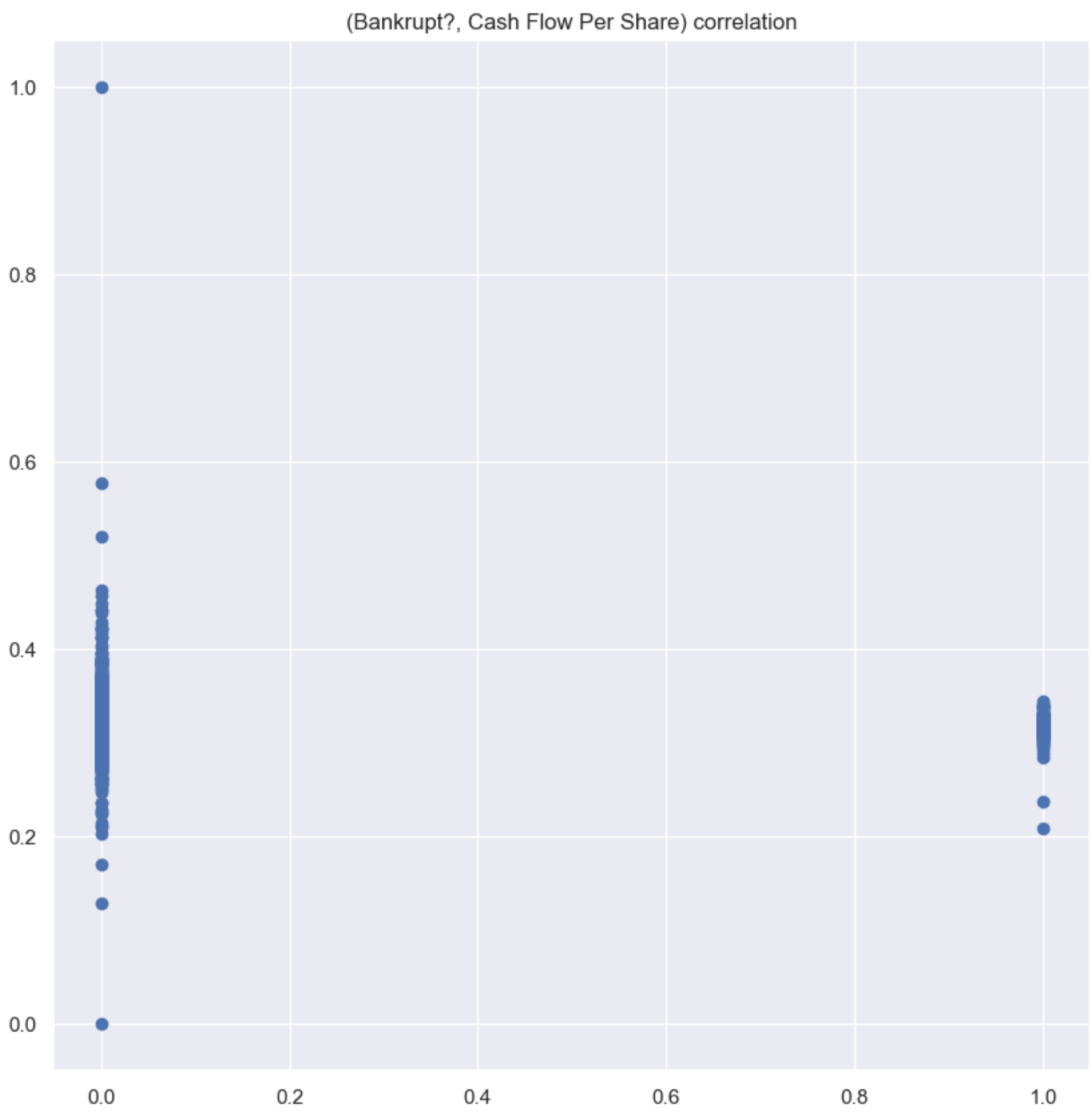

(Bankrupt?, Cash flow rate) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

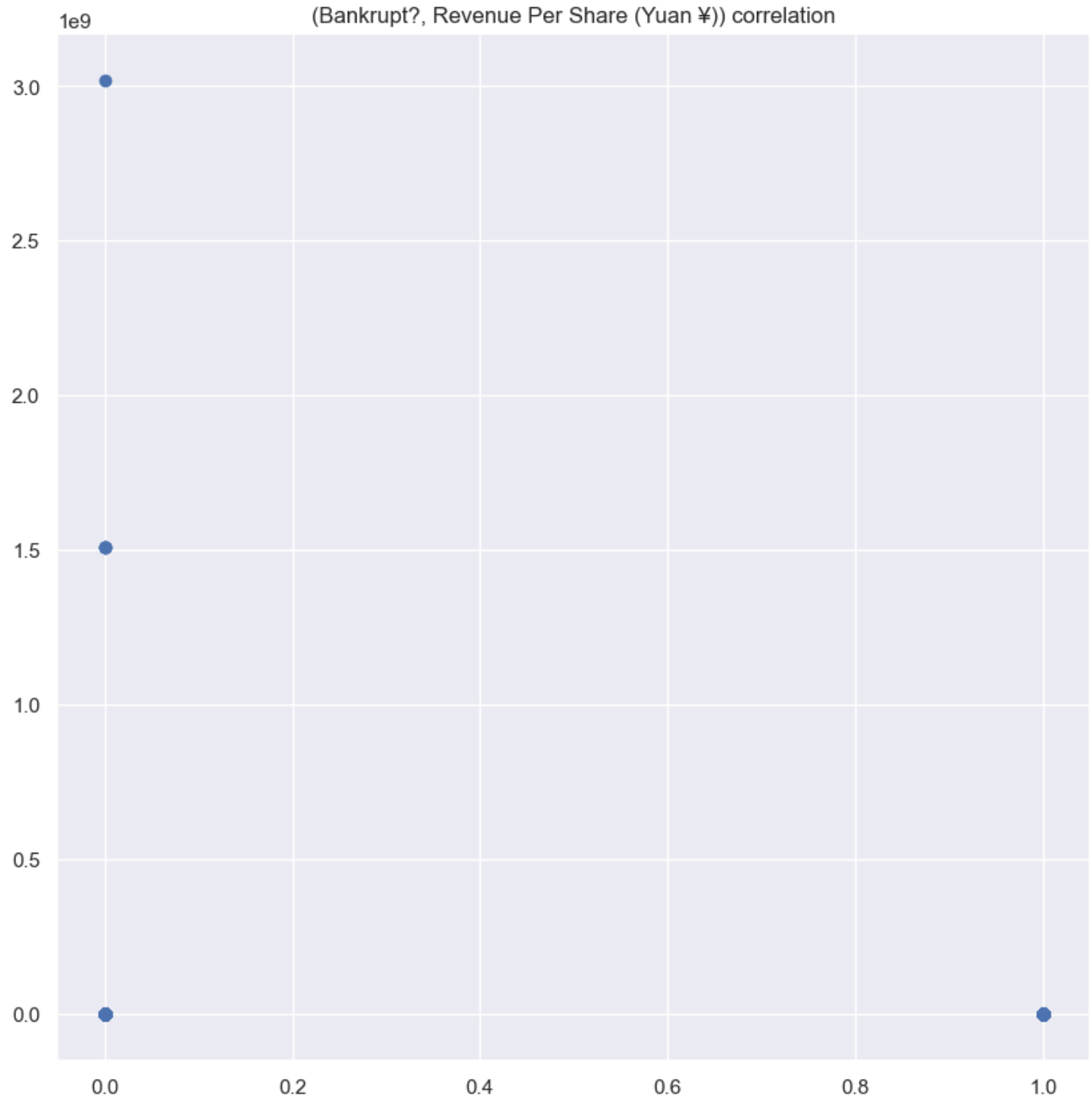

(Bankrupt?, Interest-bearing debt interest rate) correlation
1e9
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

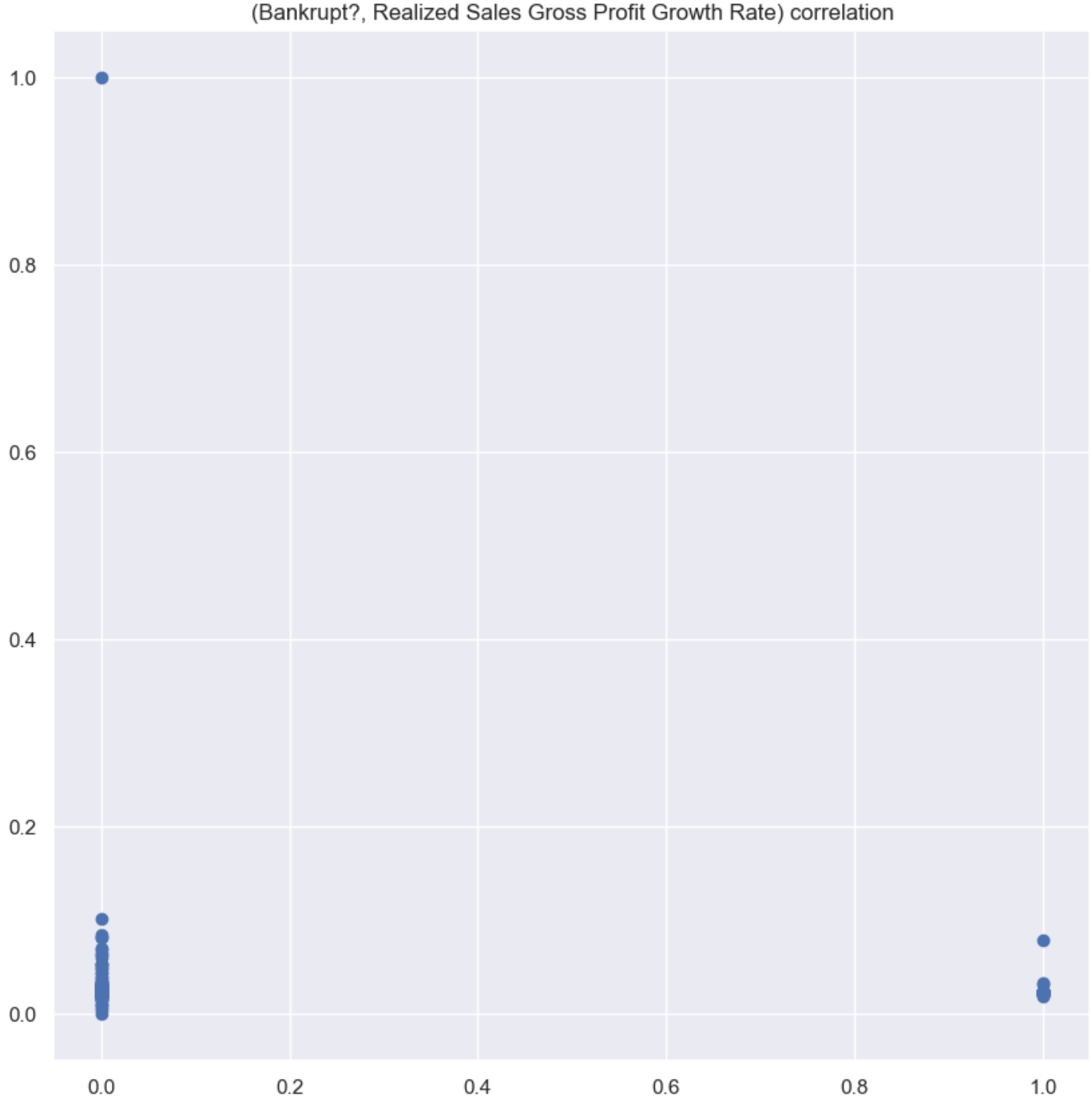

(Bankrupt?, Tax rate (A)) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

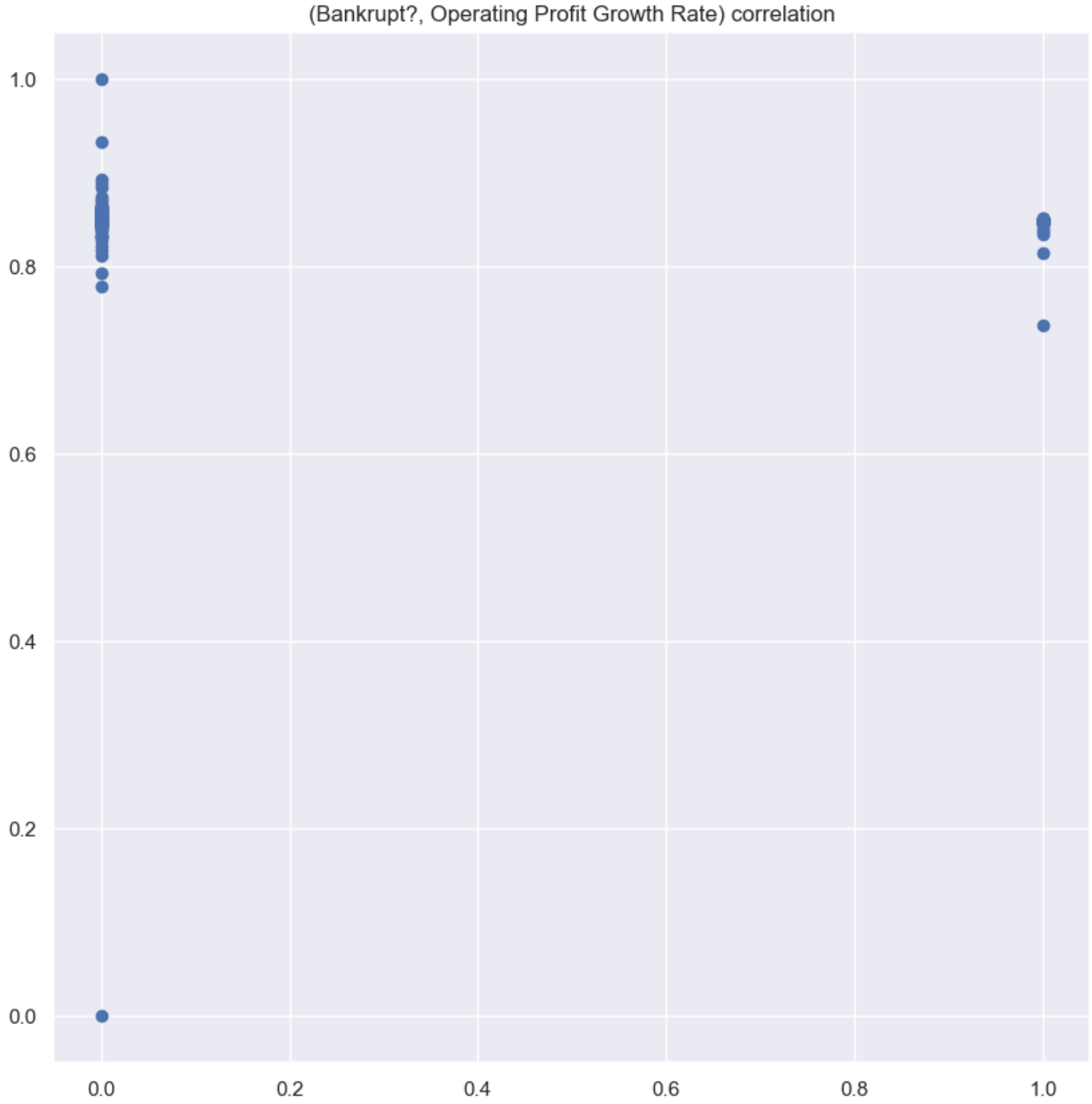

(Bankrupt?, Net Value Per Share (B)) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

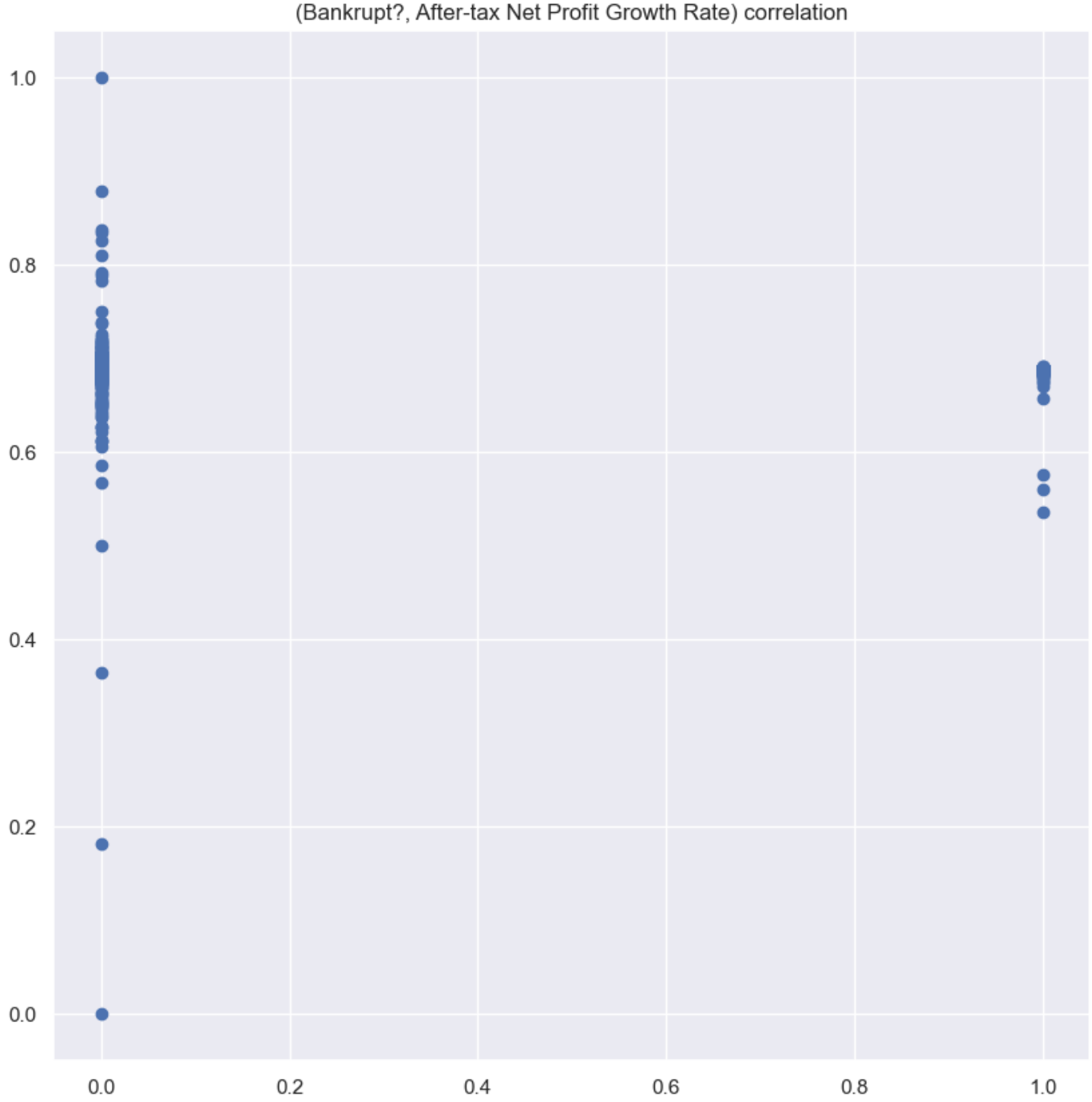

(Bankrupt?, Persistent EPS in the Last Four Seasons) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

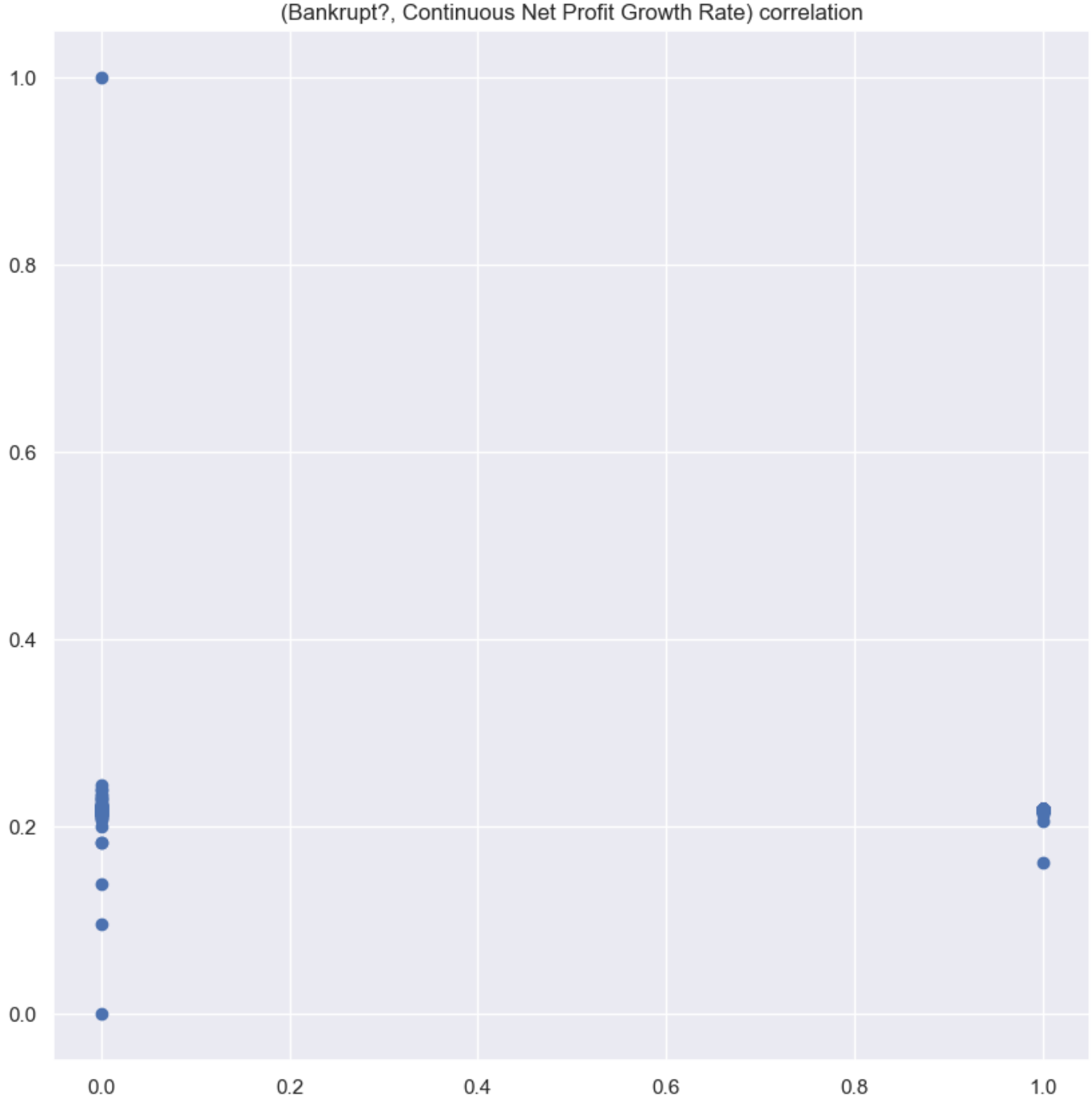

(Bankrupt?, Cash Flow Per Share) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

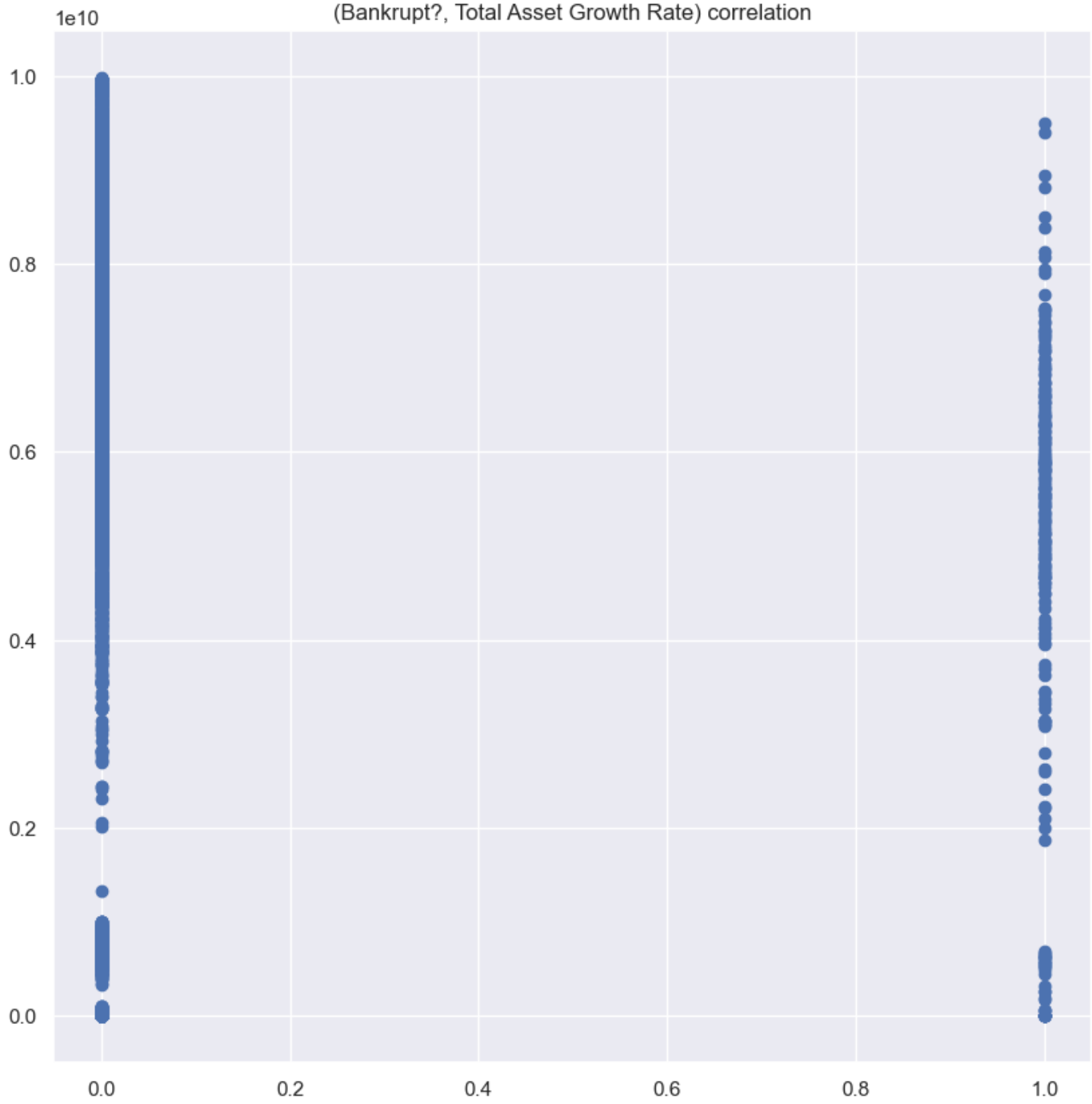

(Bankrupt?, Revenue Per Share (Yuan ¥)) correlation
1e9
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

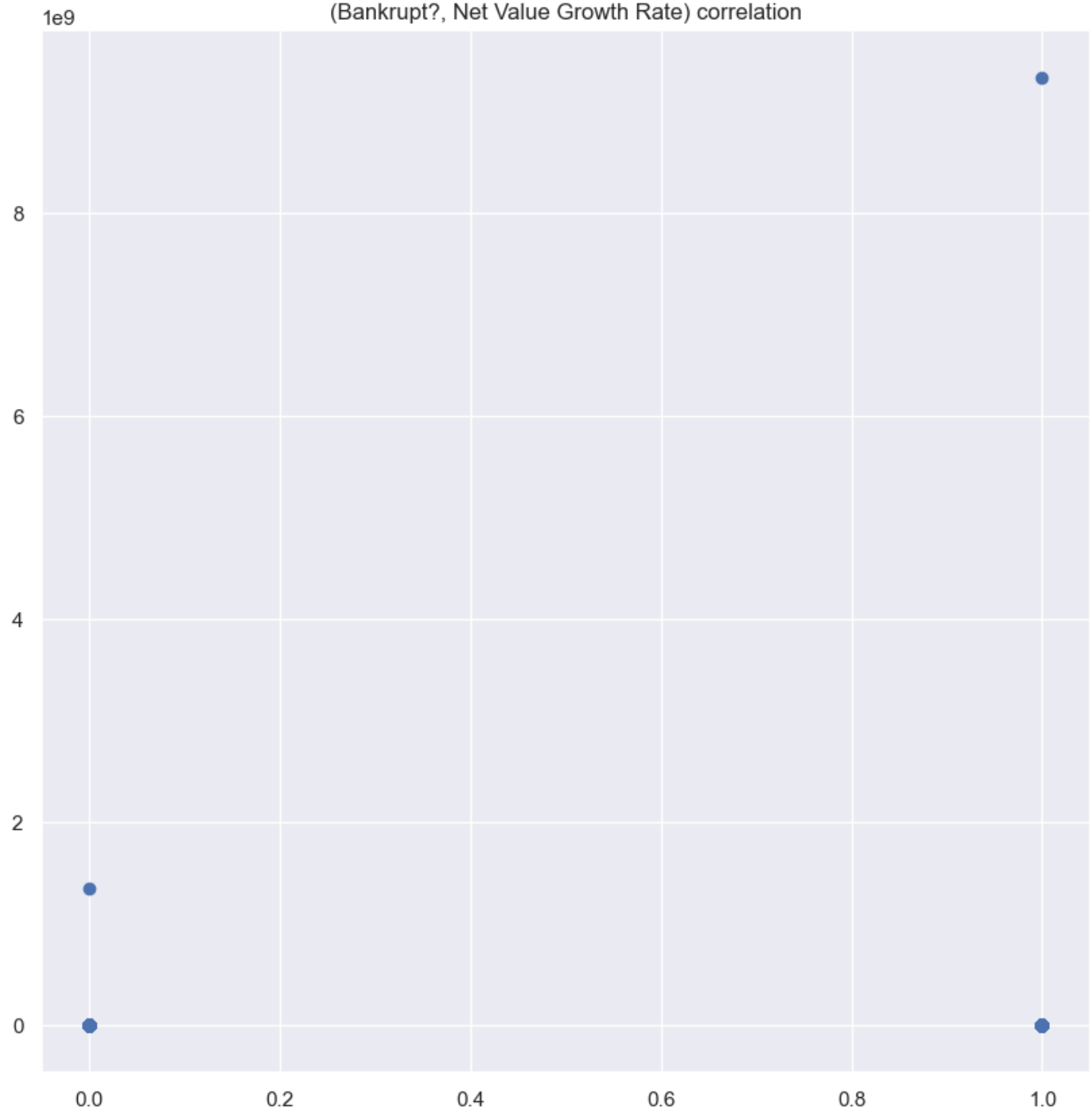

(Bankrupt?, Realized Sales Gross Profit Growth Rate) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

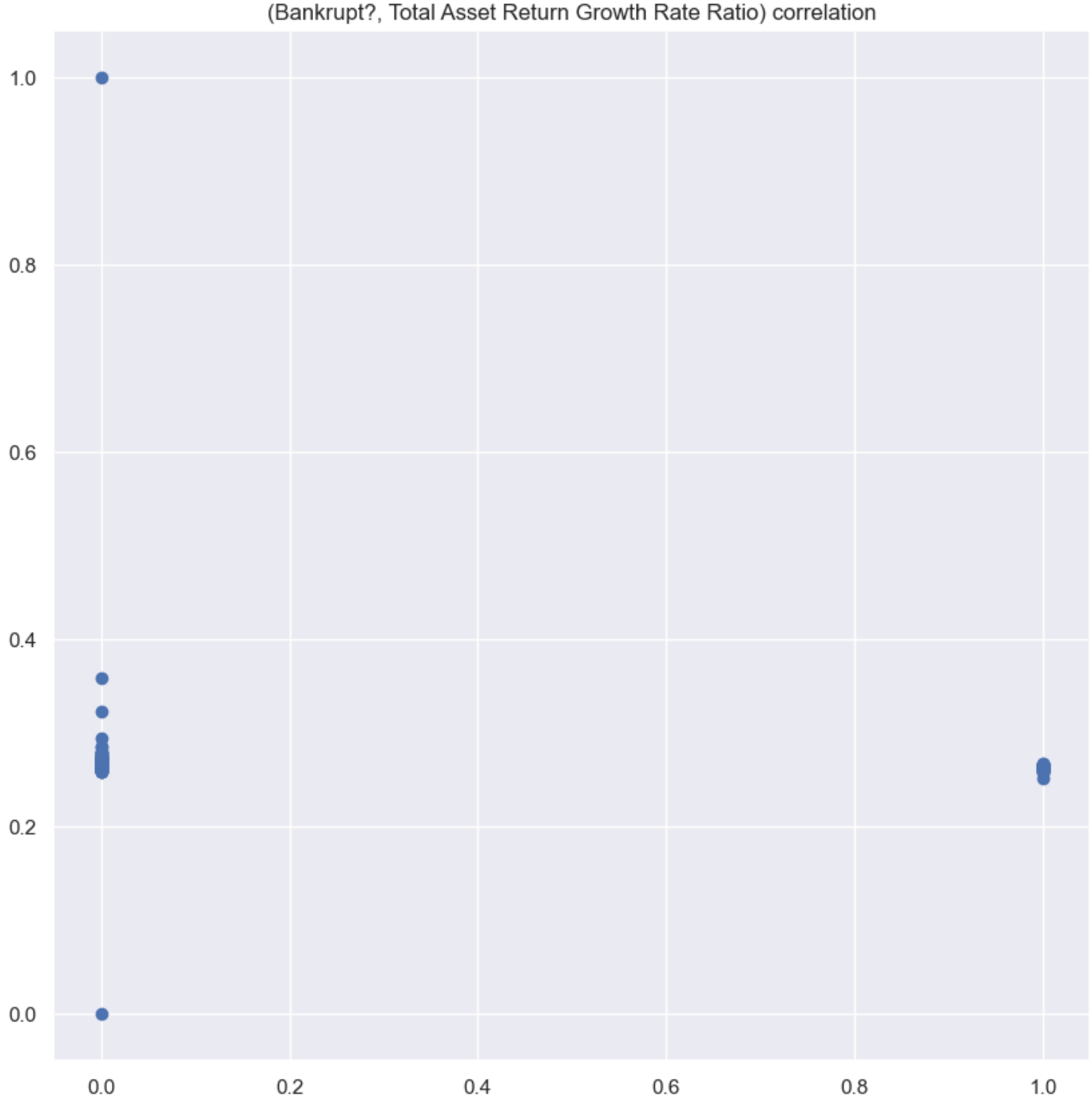

(Bankrupt?, Operating Profit Growth Rate) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

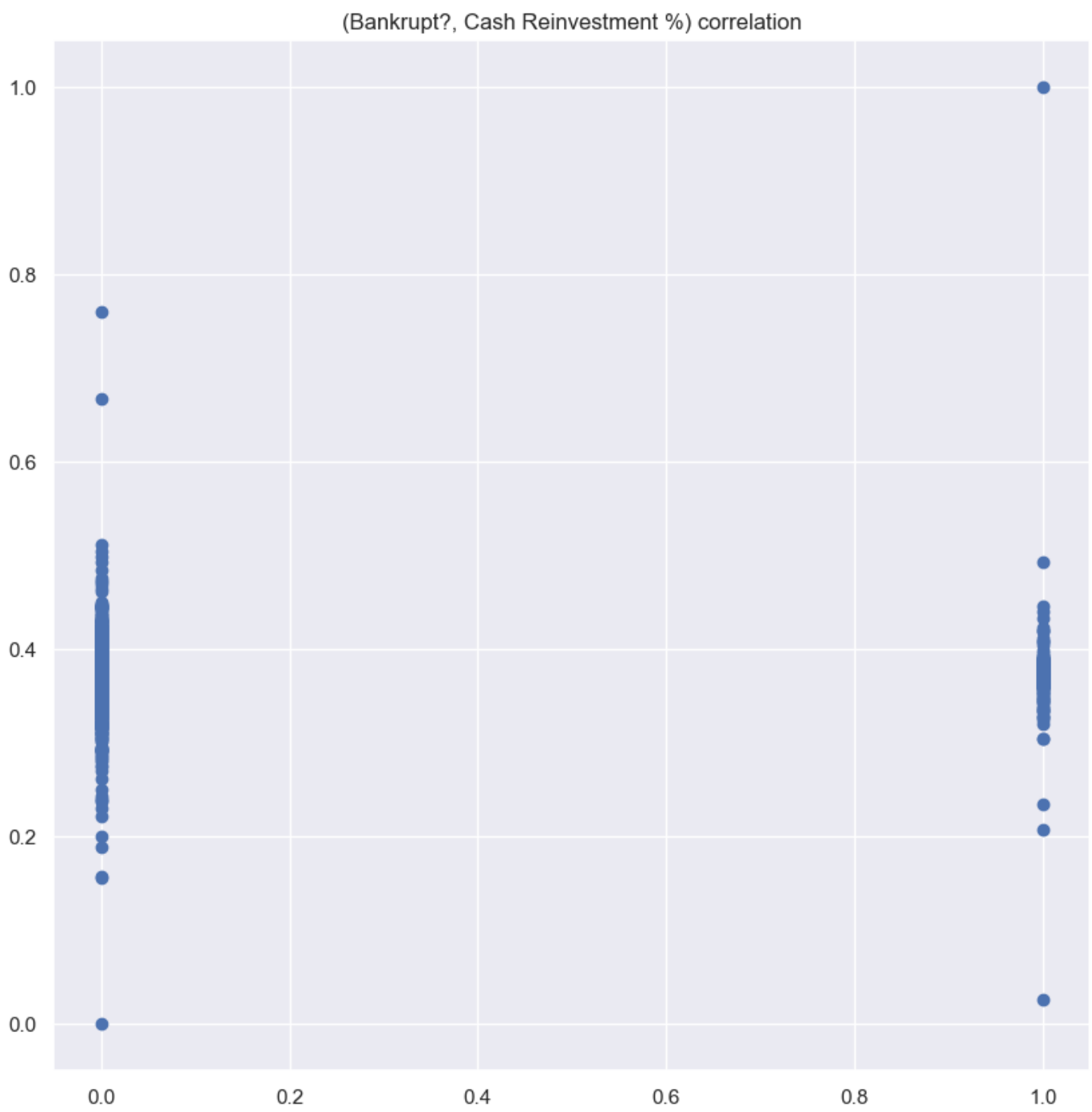

(Bankrupt?, After-tax Net Profit Growth Rate) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

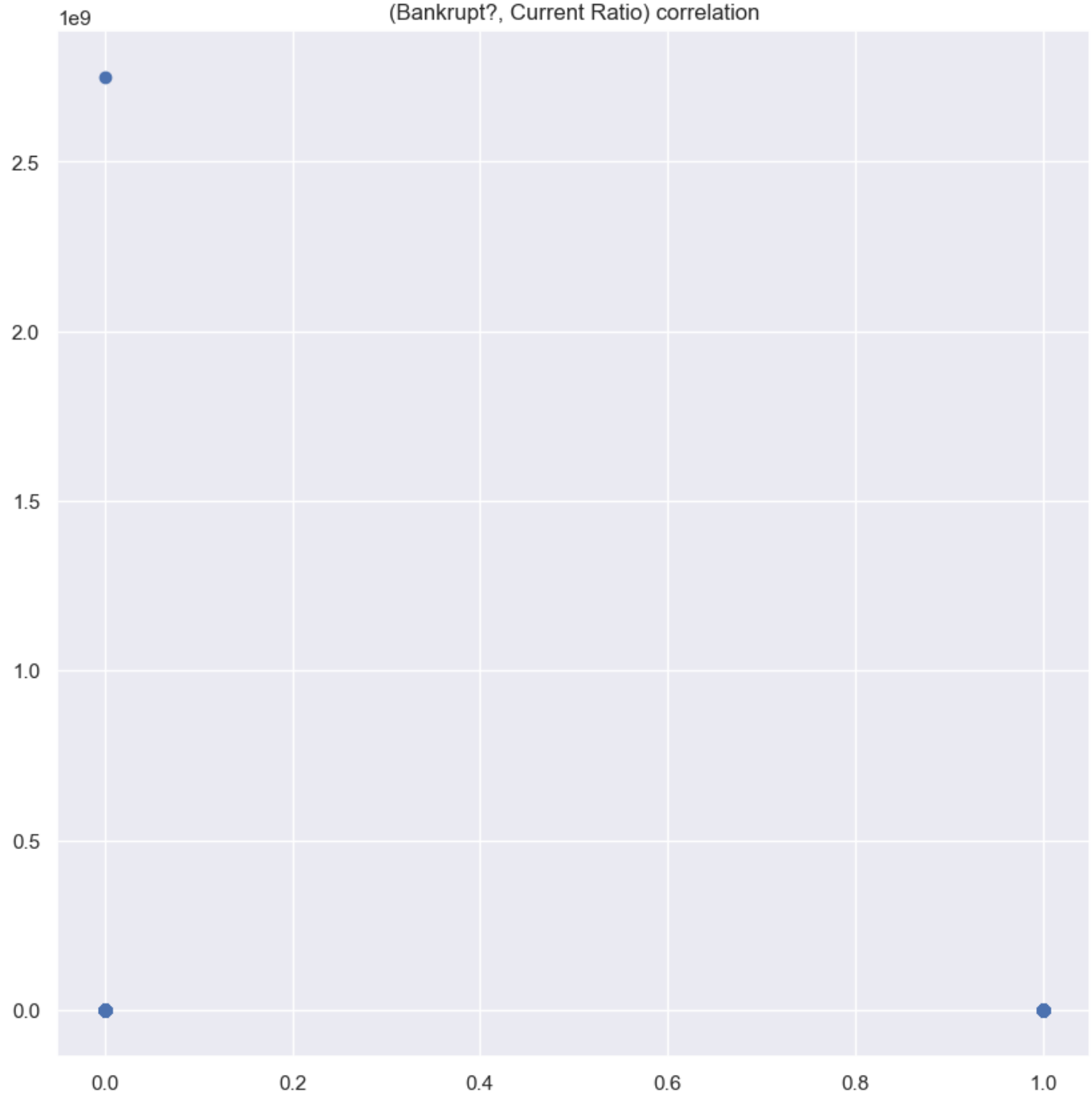

(Bankrupt?, Continuous Net Profit Growth Rate) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

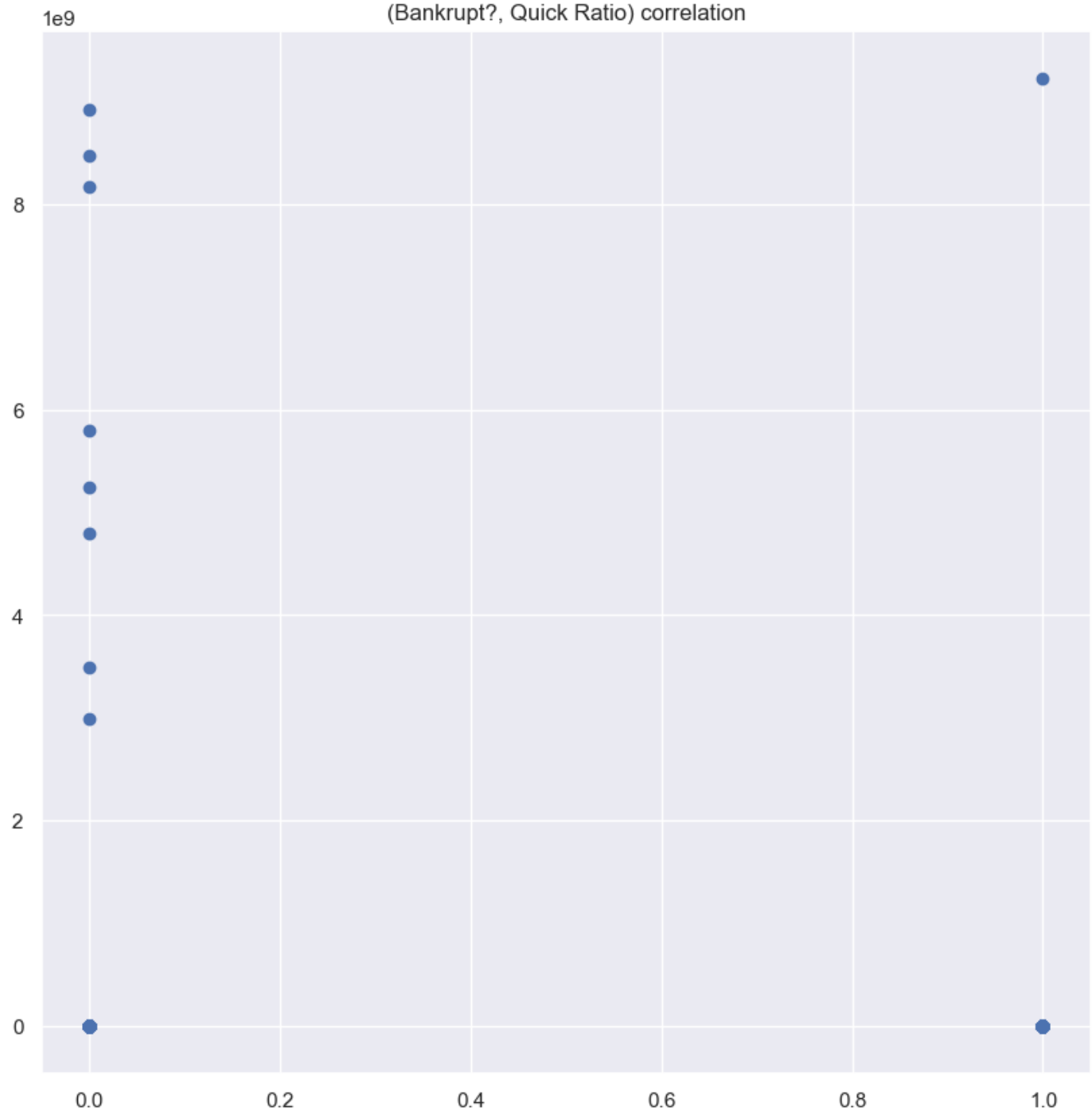

(Bankrupt?, Total Asset Growth Rate) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

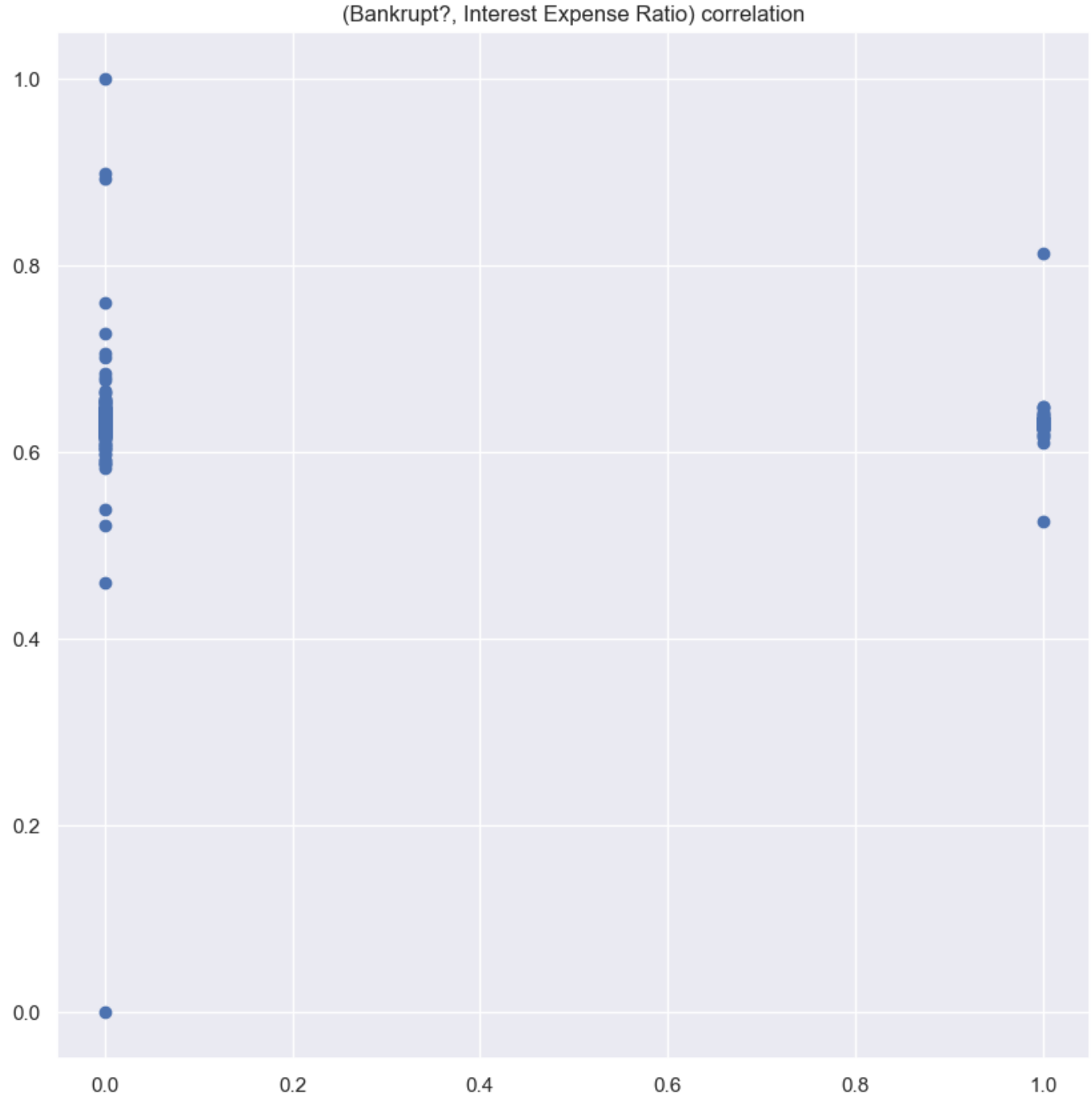

(Bankrupt?, Net Value Growth Rate) correlation
1e9
8
6
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0

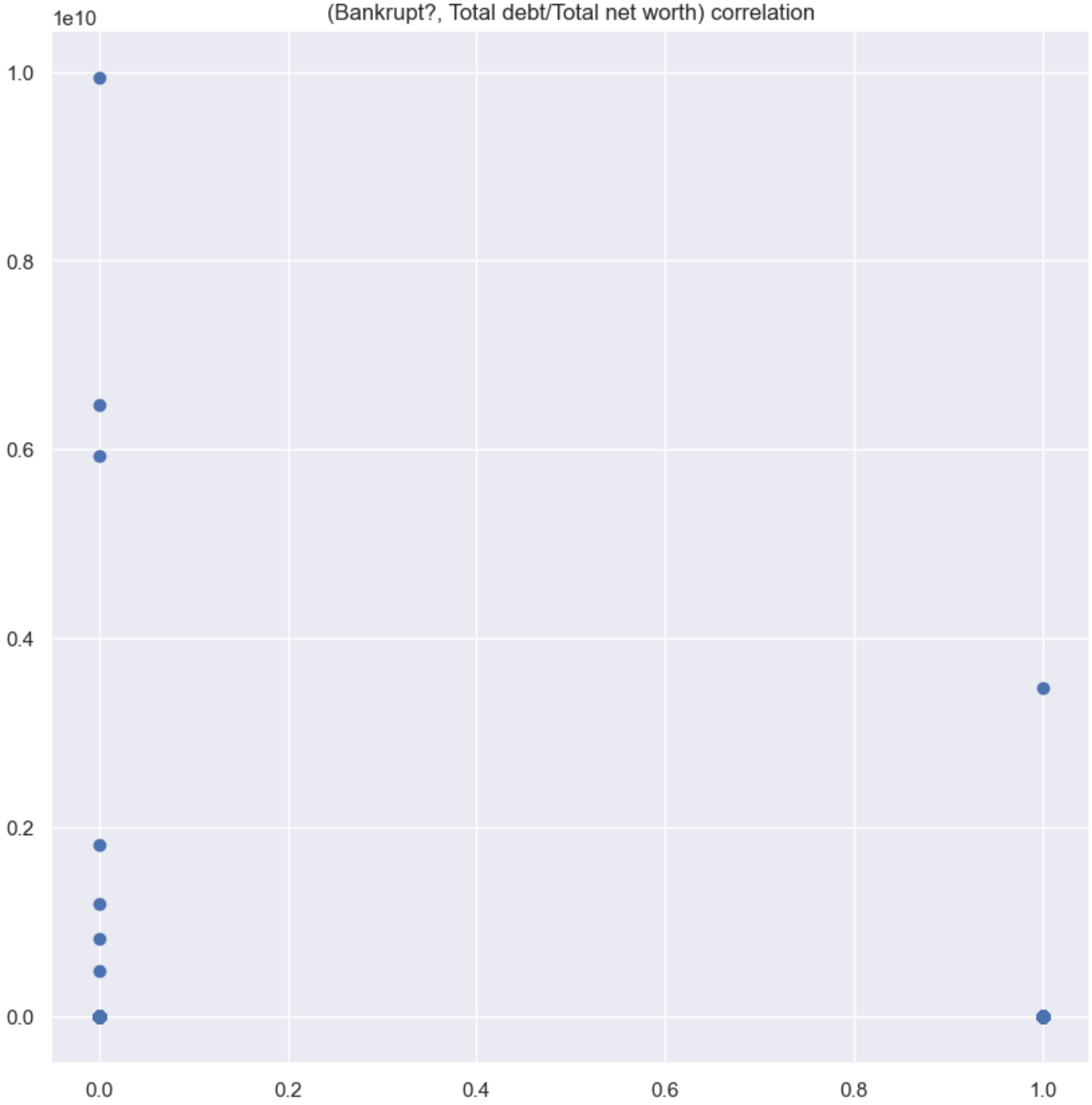

(Bankrupt?, Total Asset Return Growth Rate Ratio) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

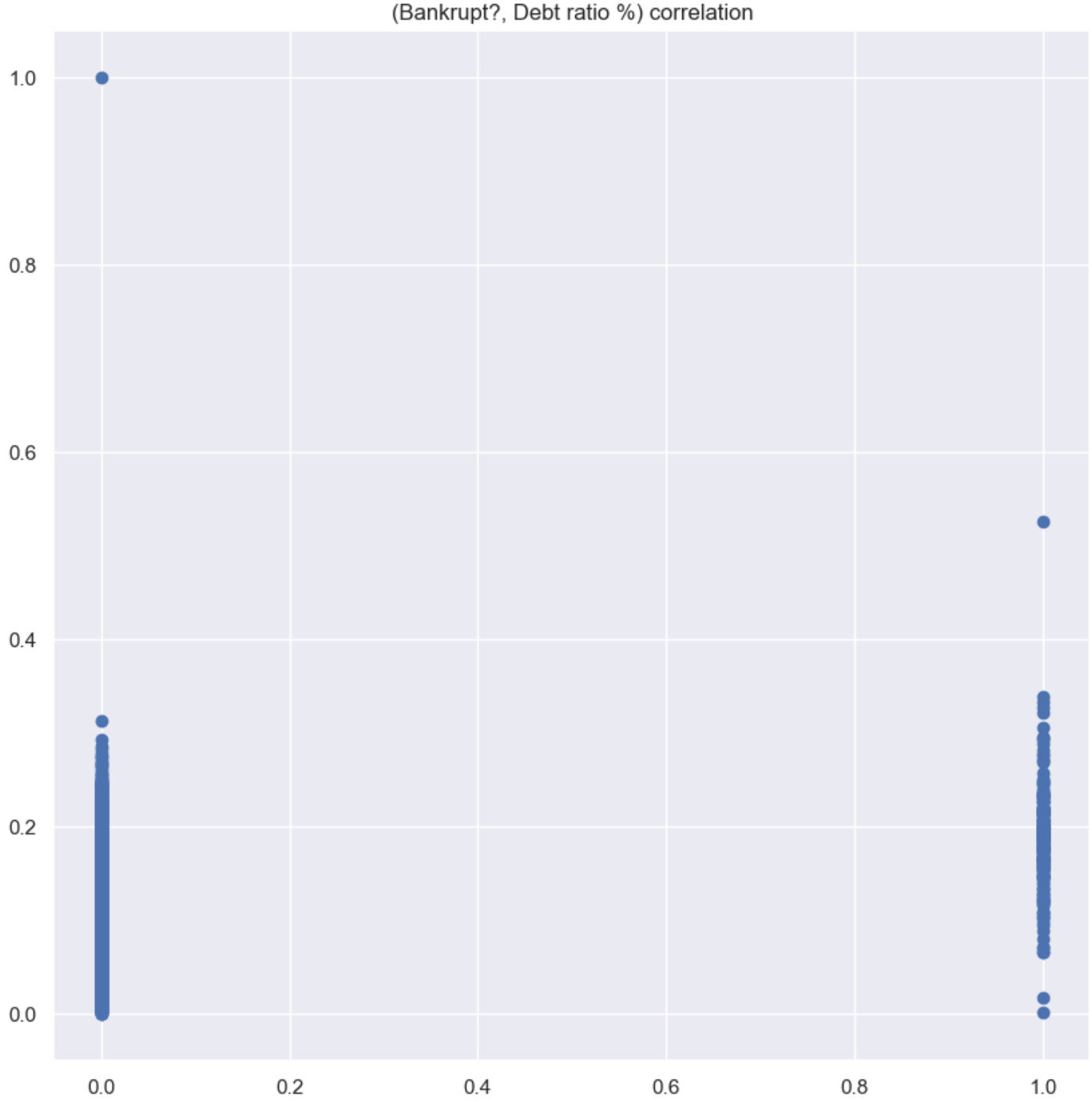

(Bankrupt?, Cash Reinvestment %) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

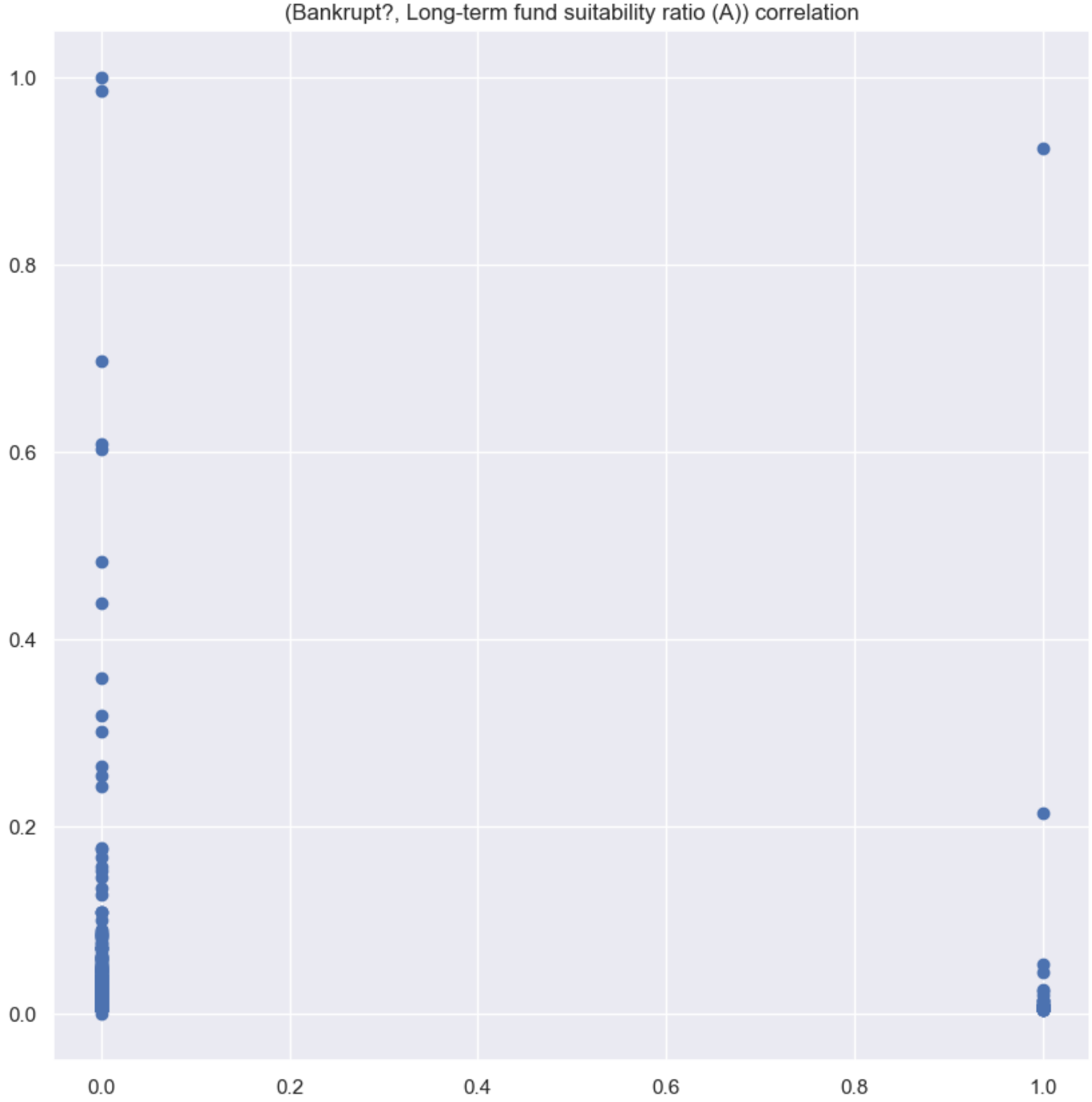

(Bankrupt?, Current Ratio) correlation
1e9
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0

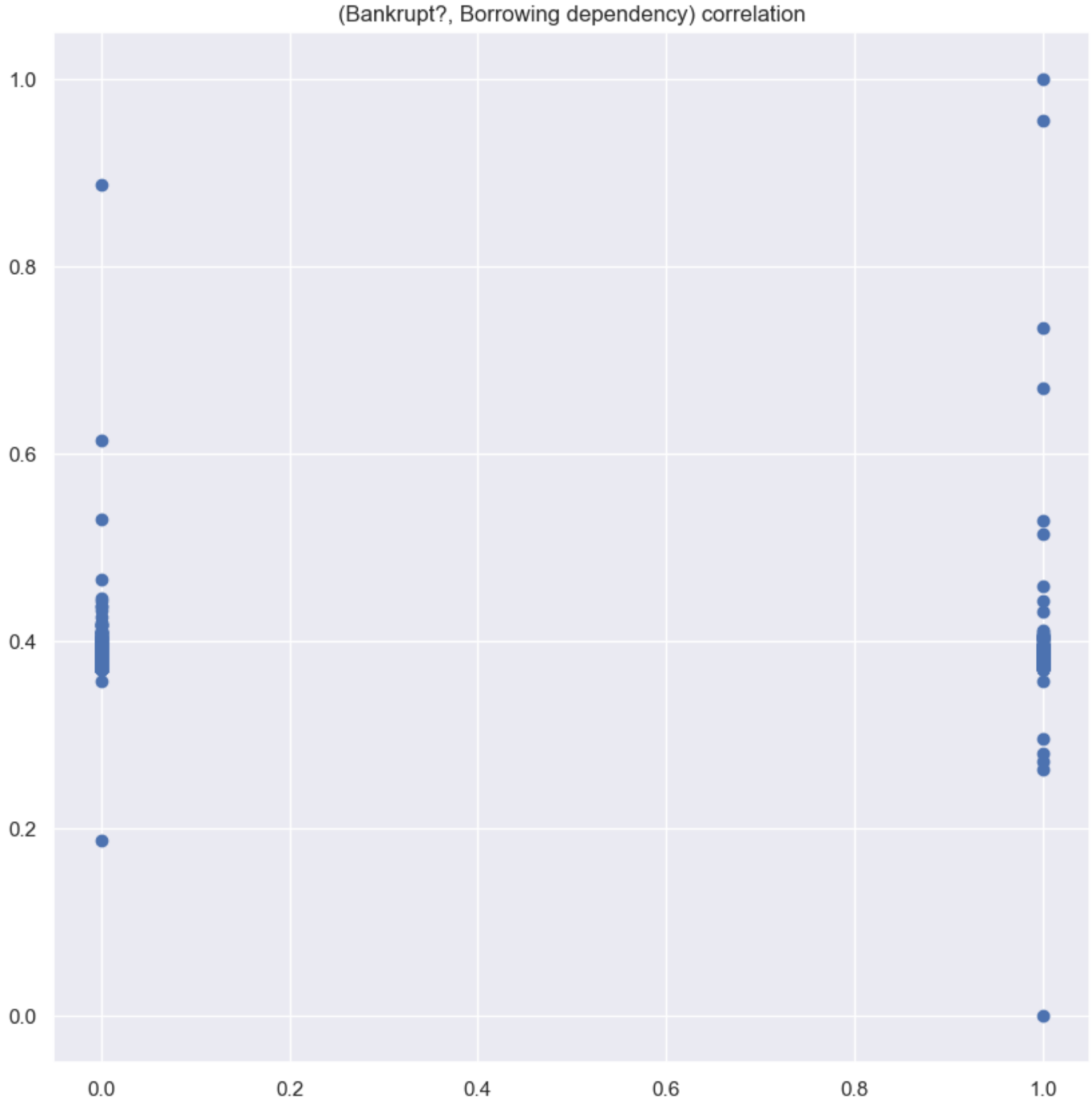

(Bankrupt?, Quick Ratio) correlation
1e9
8
6
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0

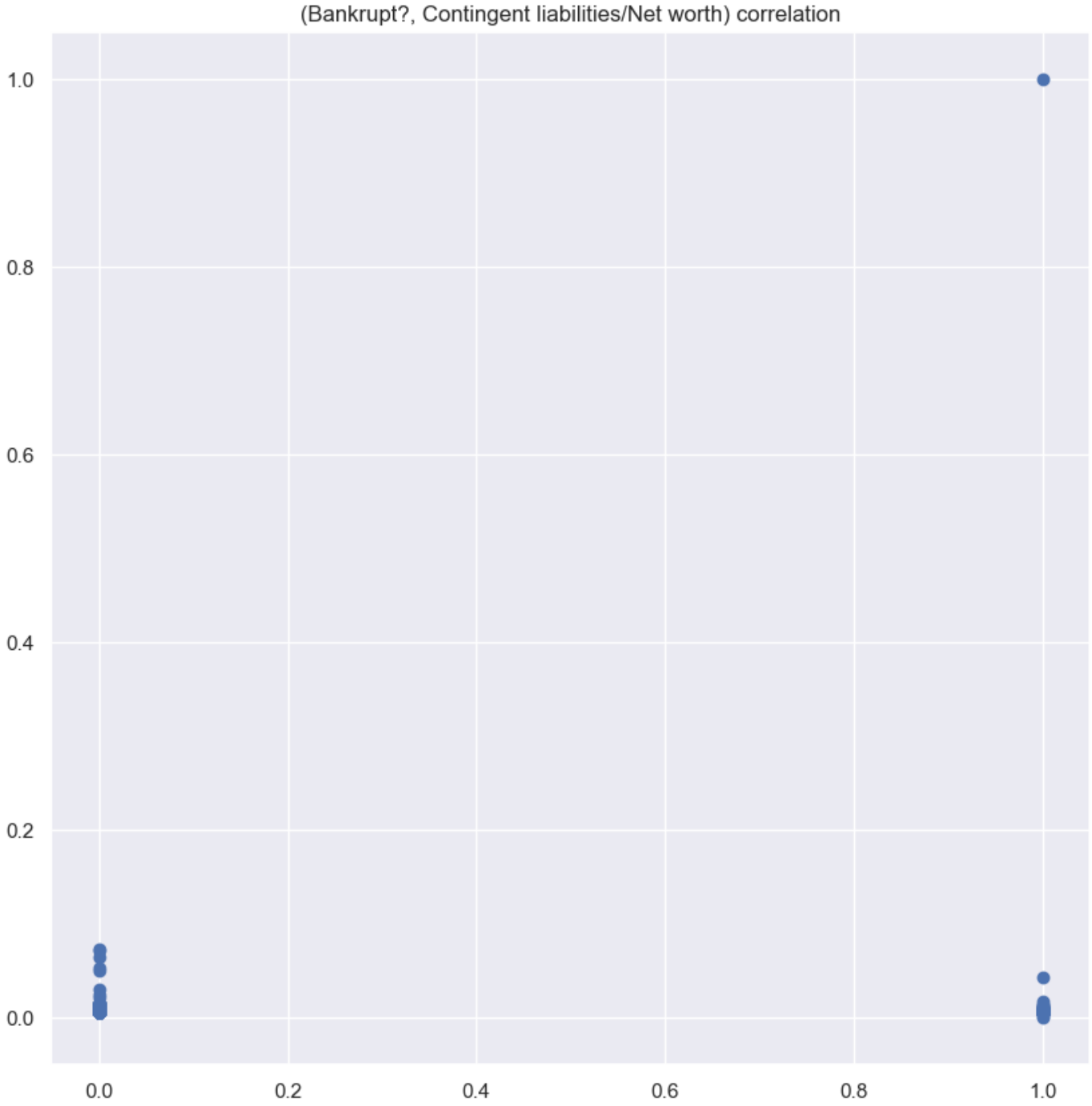

(Bankrupt?, Interest Expense Ratio) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

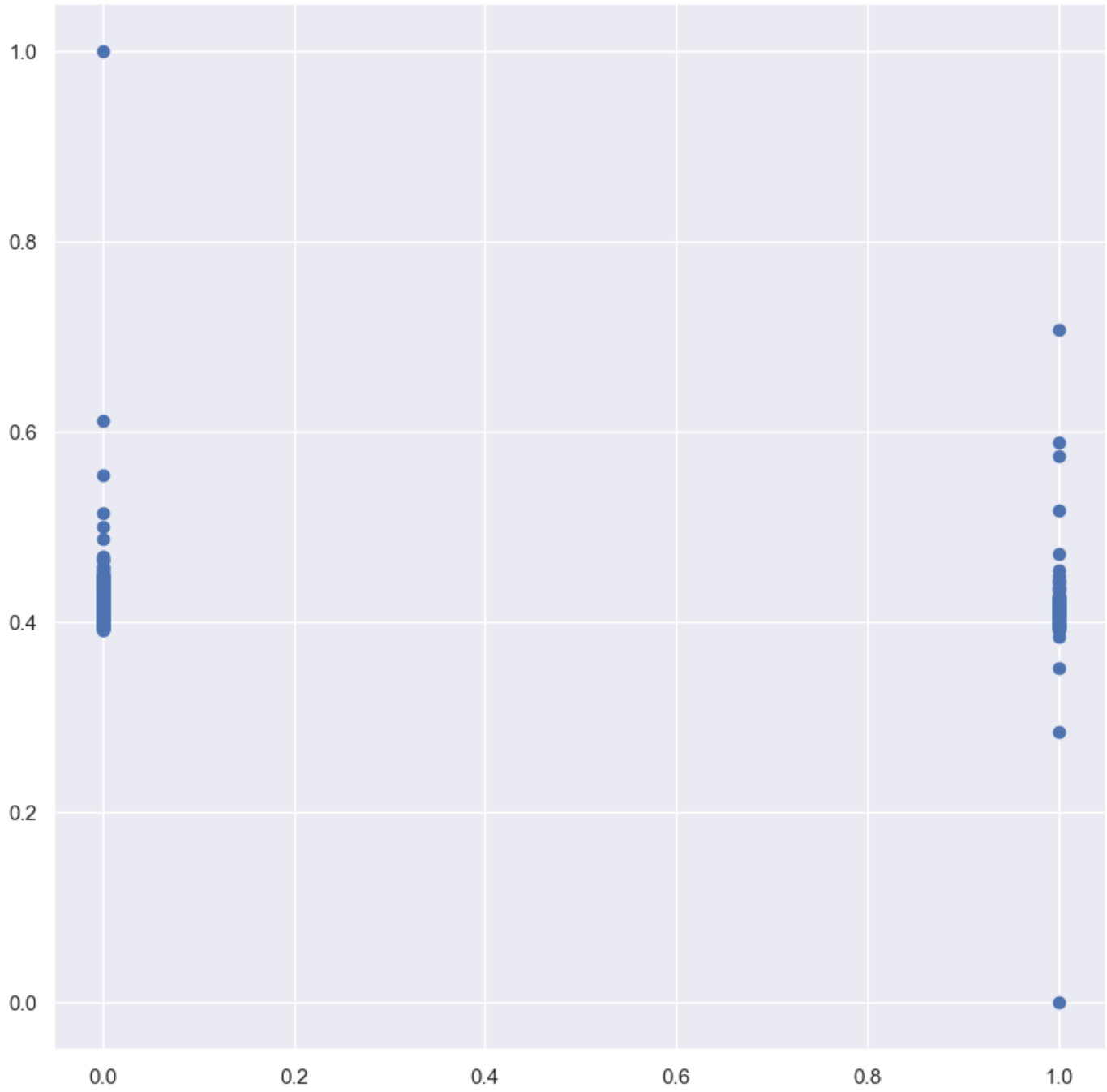

(Bankrupt?, Total debt/Total net worth) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

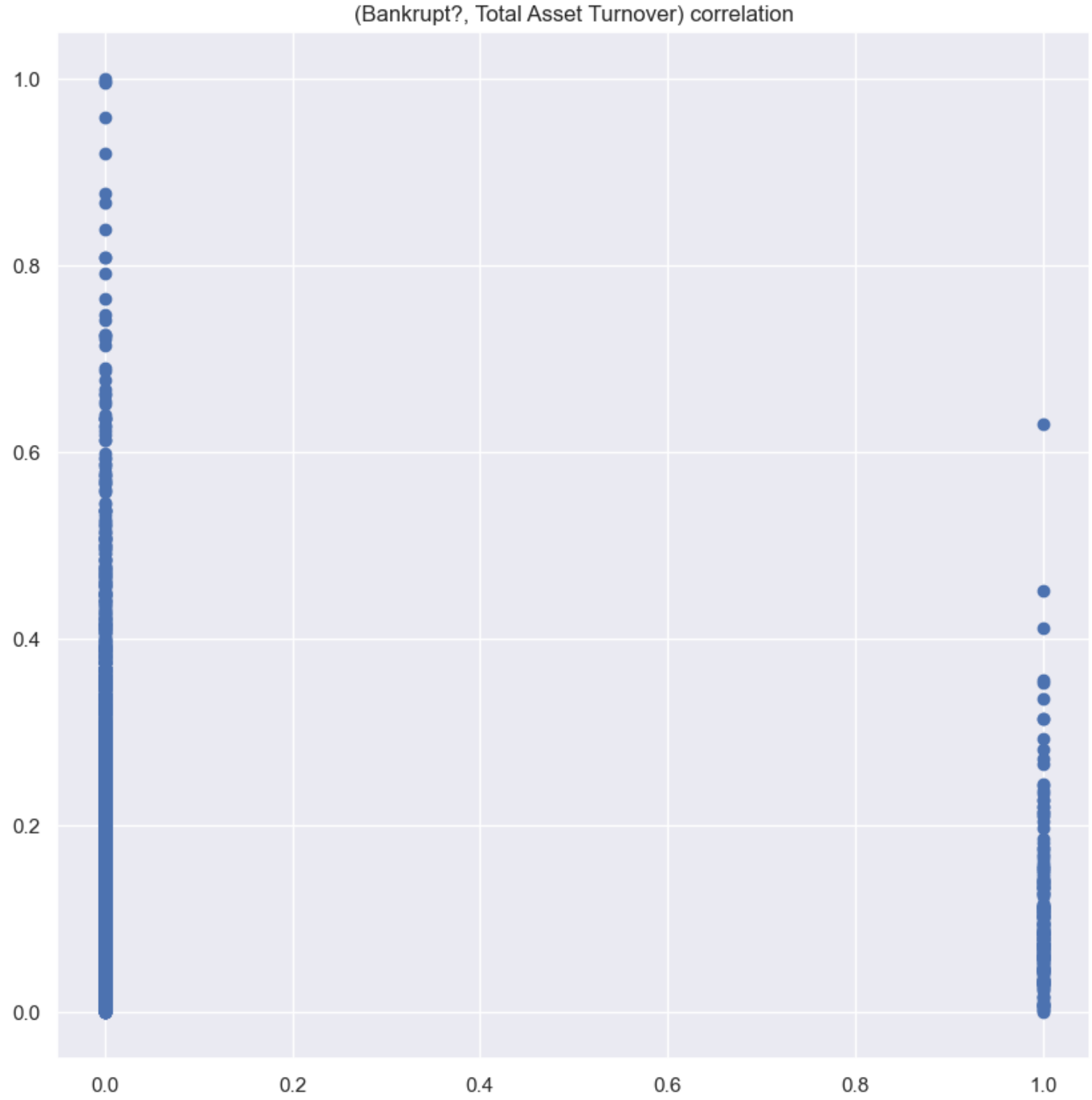

(Bankrupt?, Debt ratio %) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

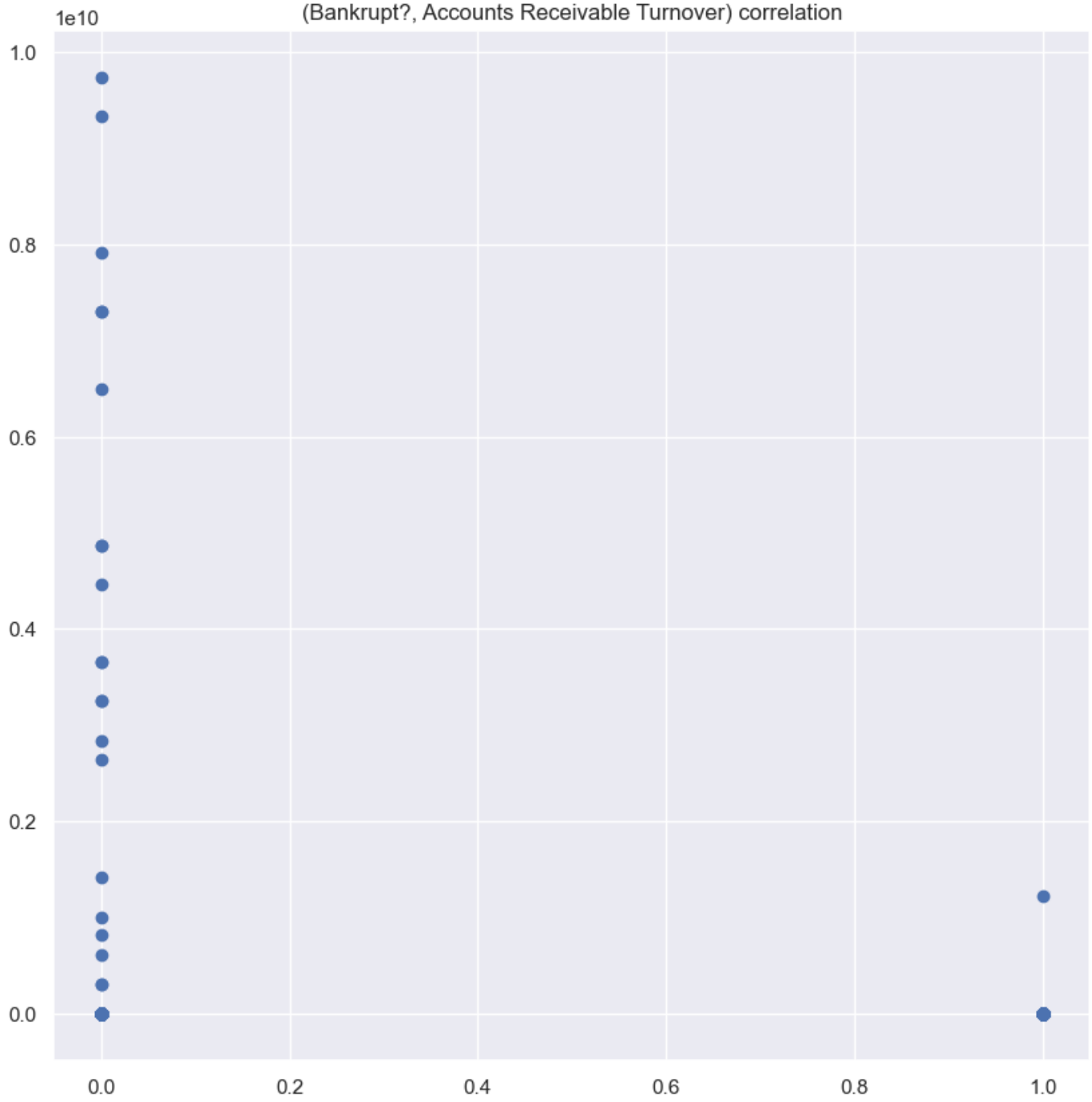

(Bankrupt?, Long-term fund suitability ratio (A)) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

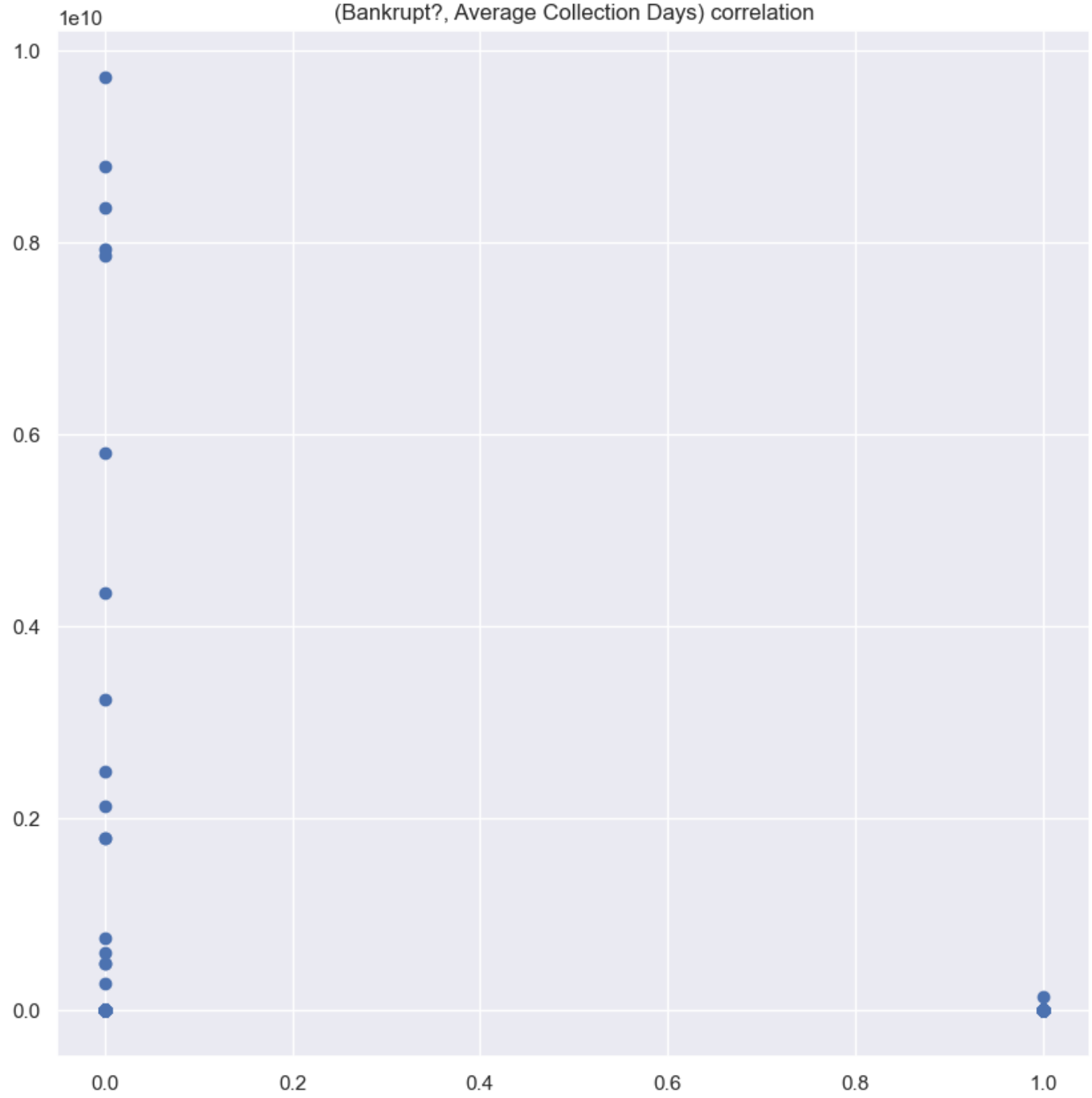

(Bankrupt?, Borrowing dependency) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

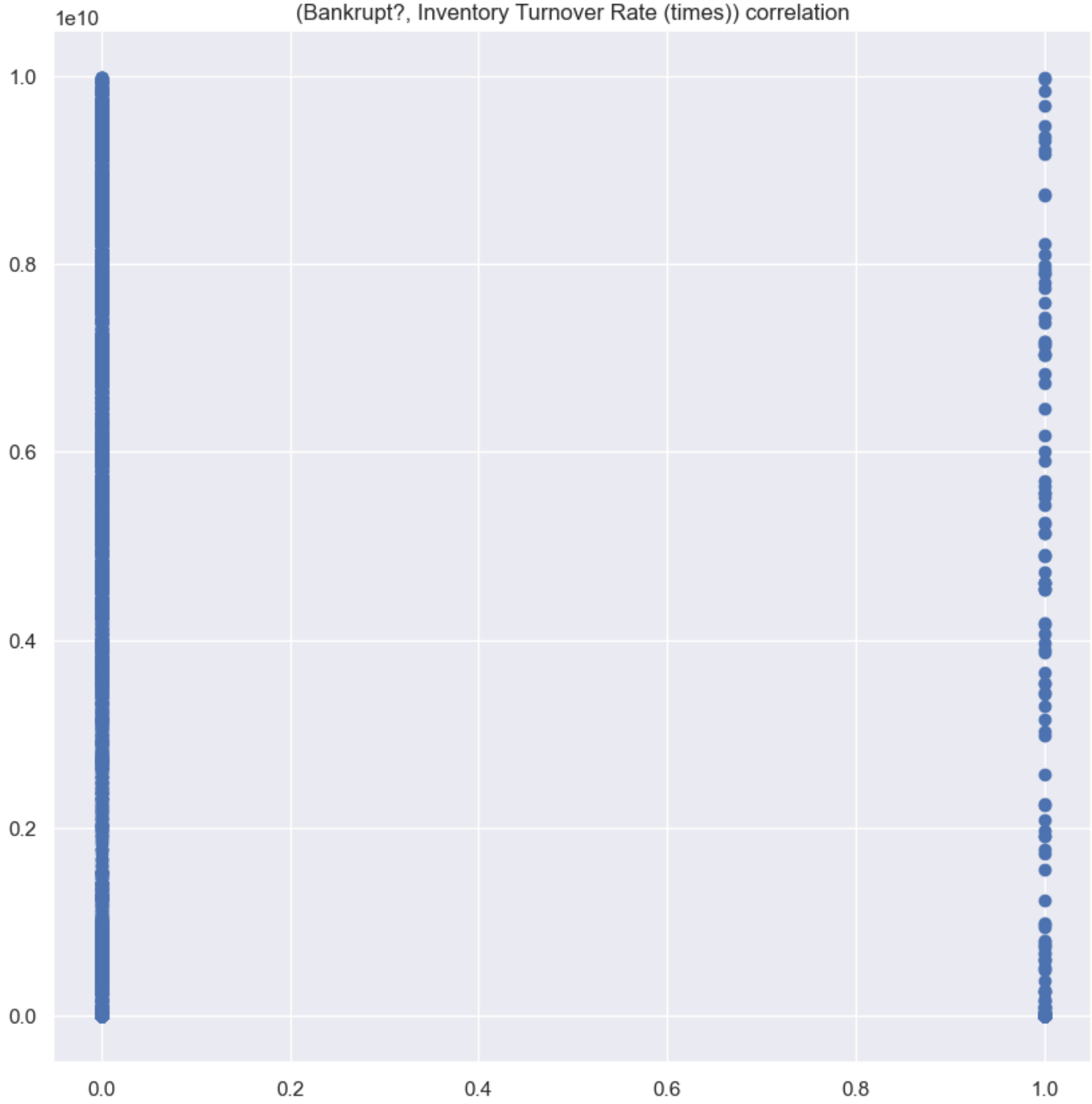

(Bankrupt?, Contingent liabilities/Net worth) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

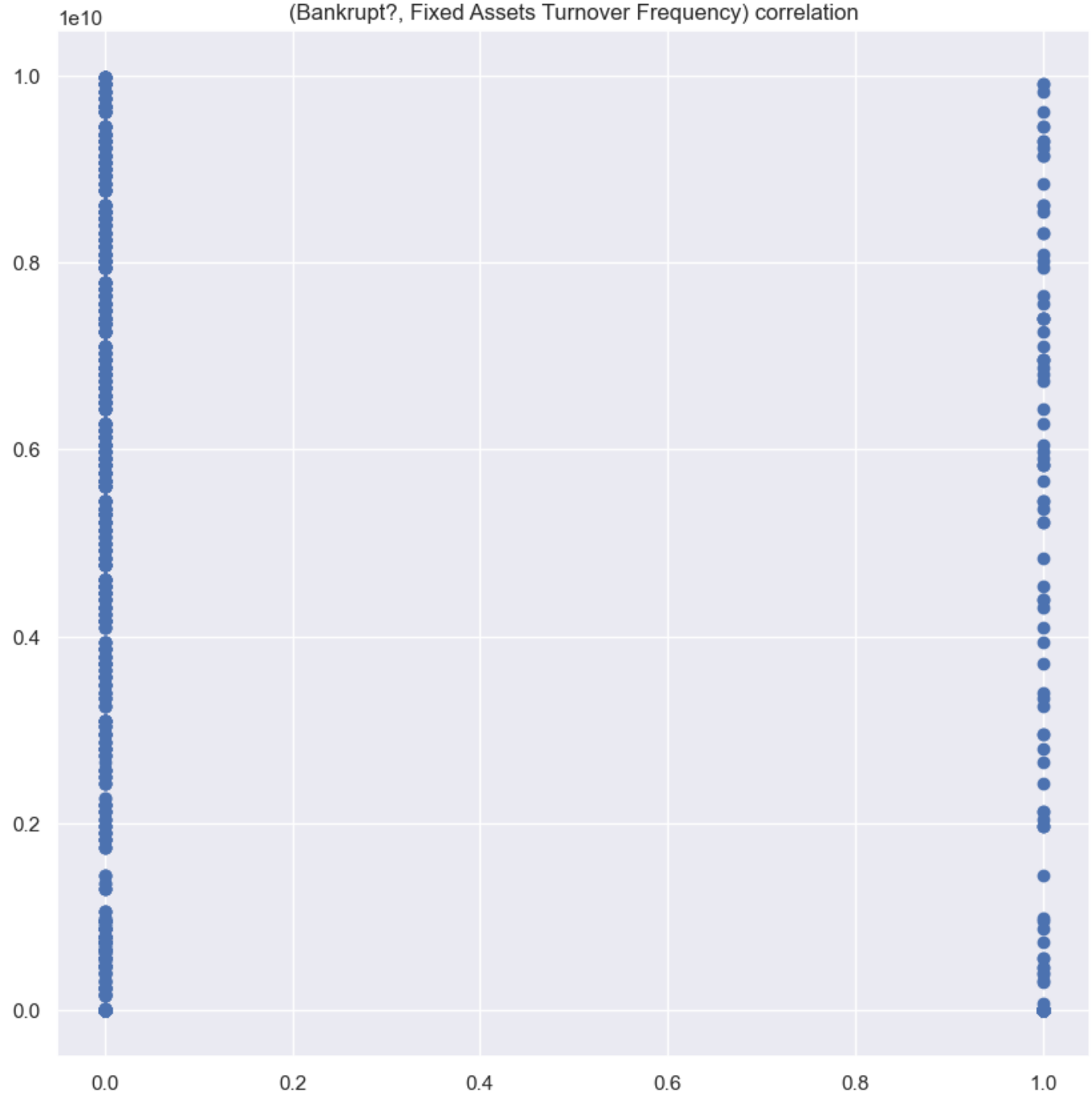

(Bankrupt?, Inventory and accounts receivable/Net value) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

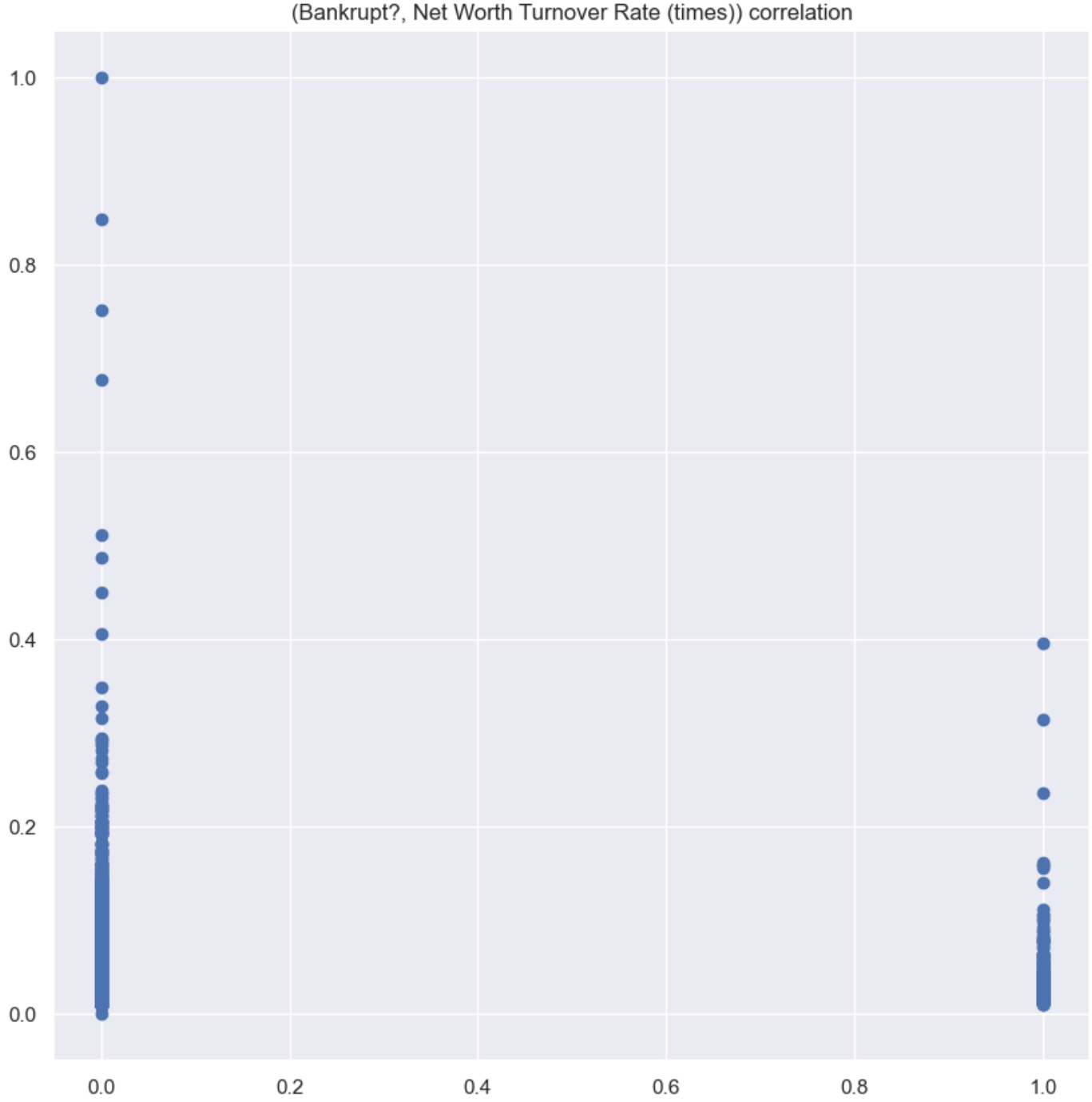

(Bankrupt?, Total Asset Turnover) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

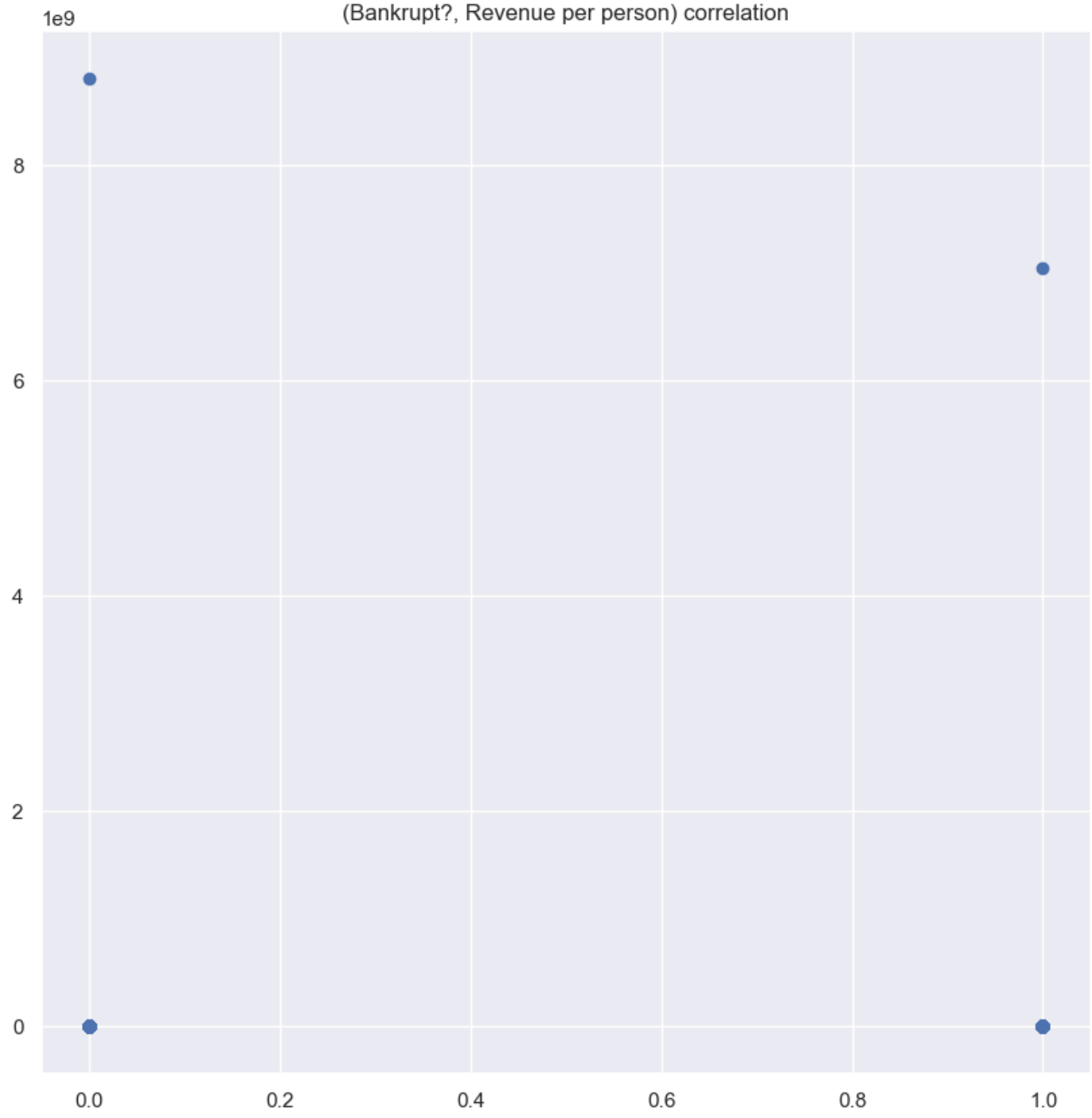

(Bankrupt?, Accounts Receivable Turnover) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

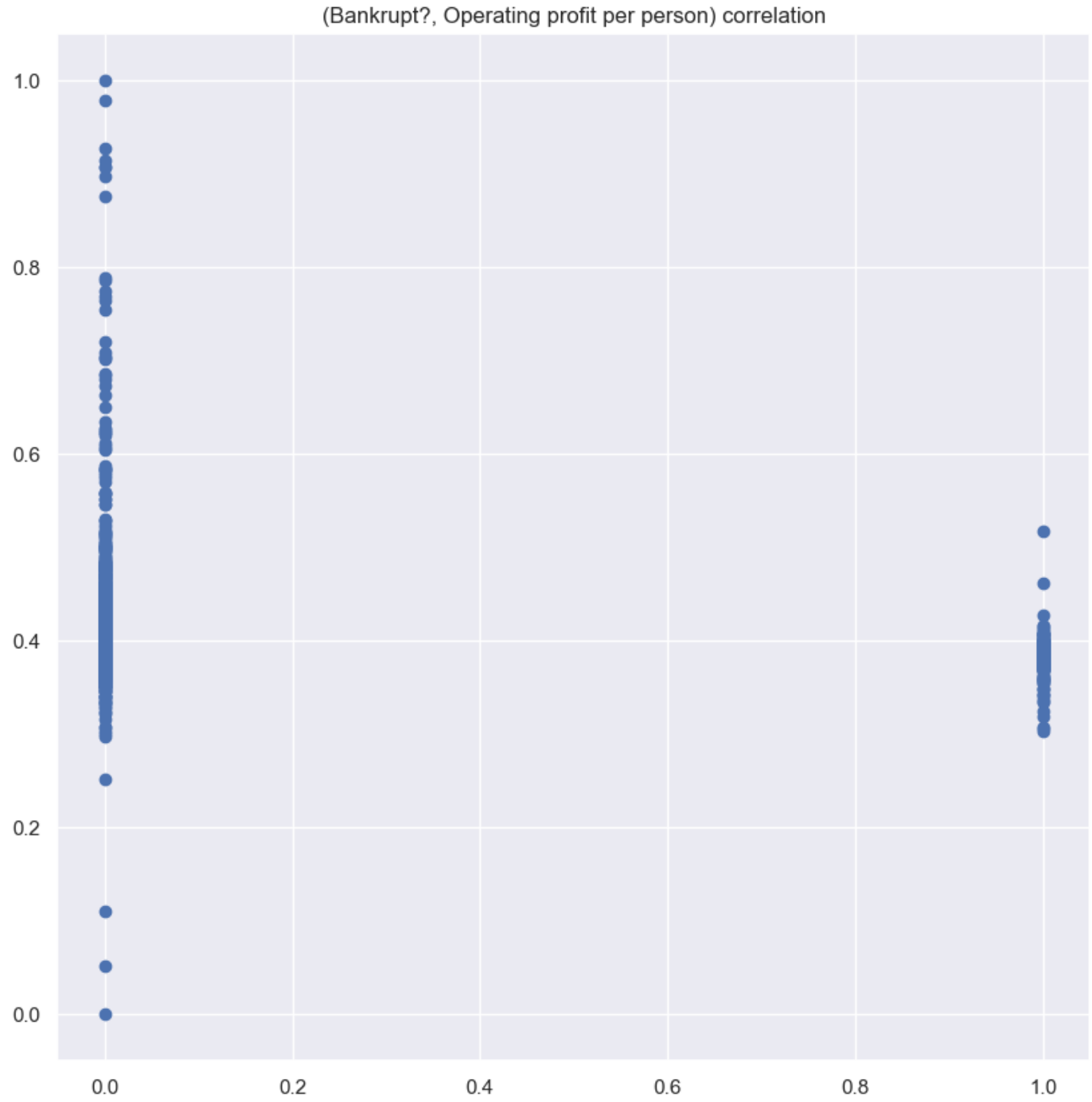

(Bankrupt?, Average Collection Days) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

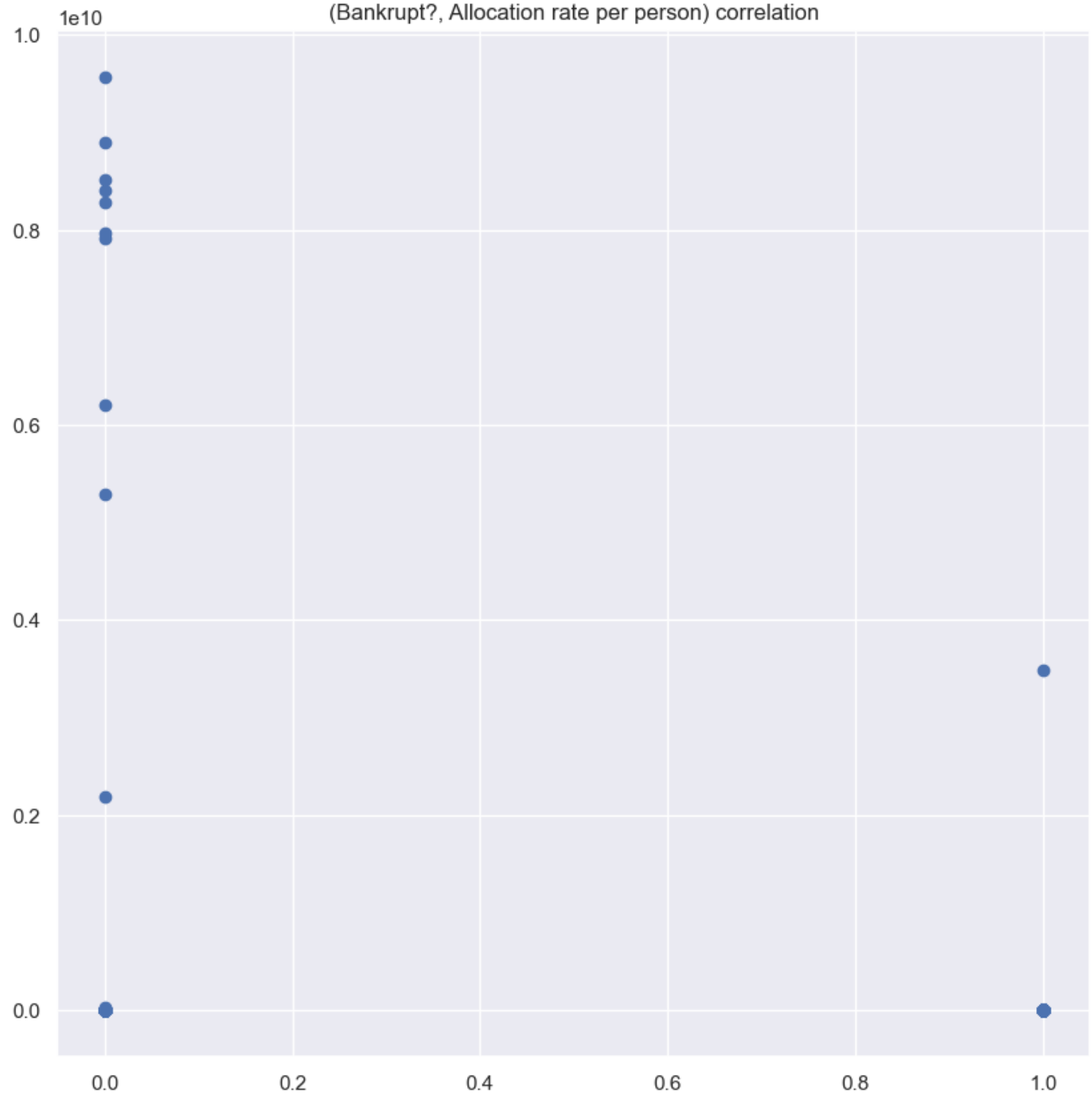

(Bankrupt?, Inventory Turnover Rate (times)) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

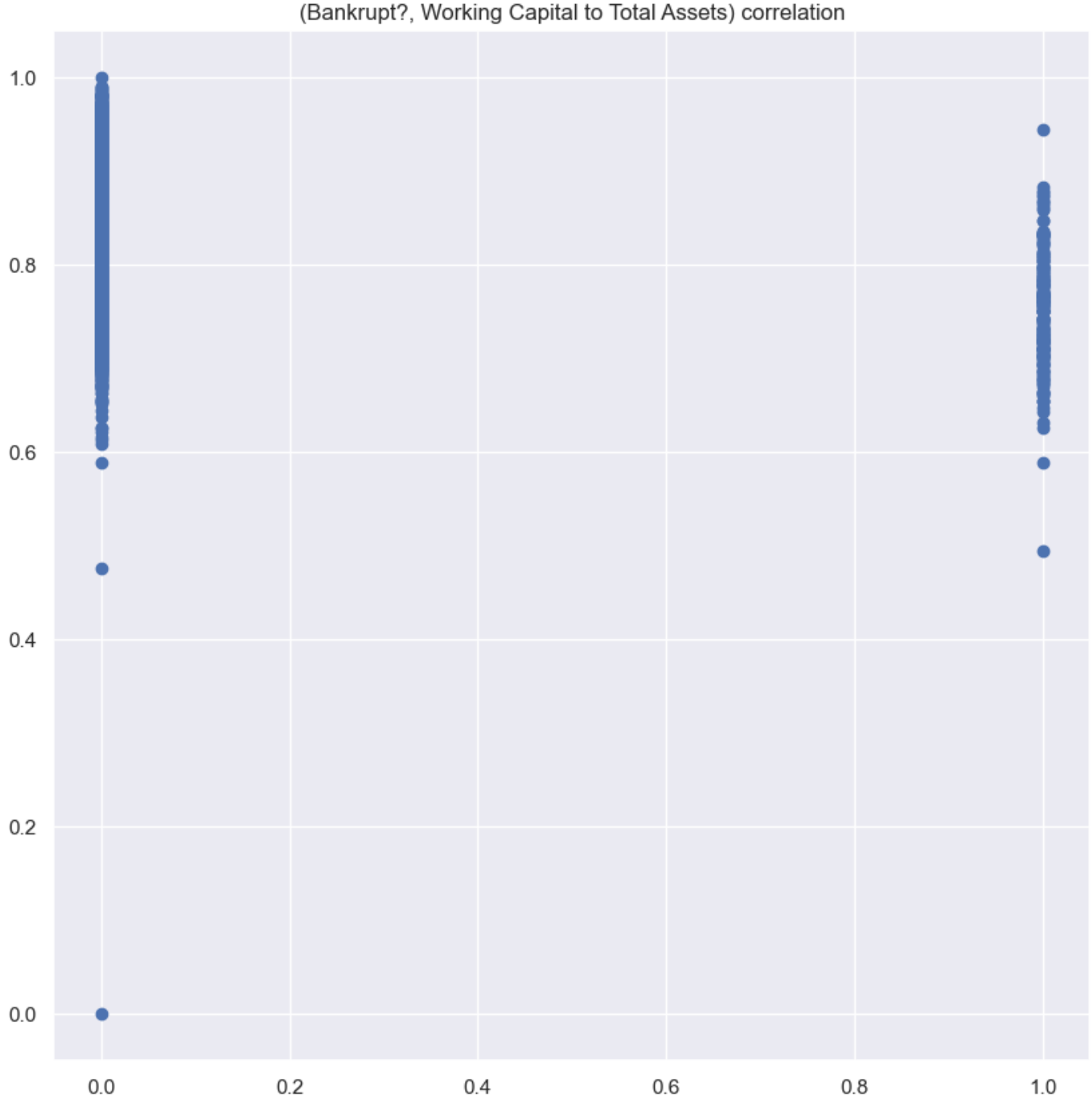

(Bankrupt?, Fixed Assets Turnover Frequency) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

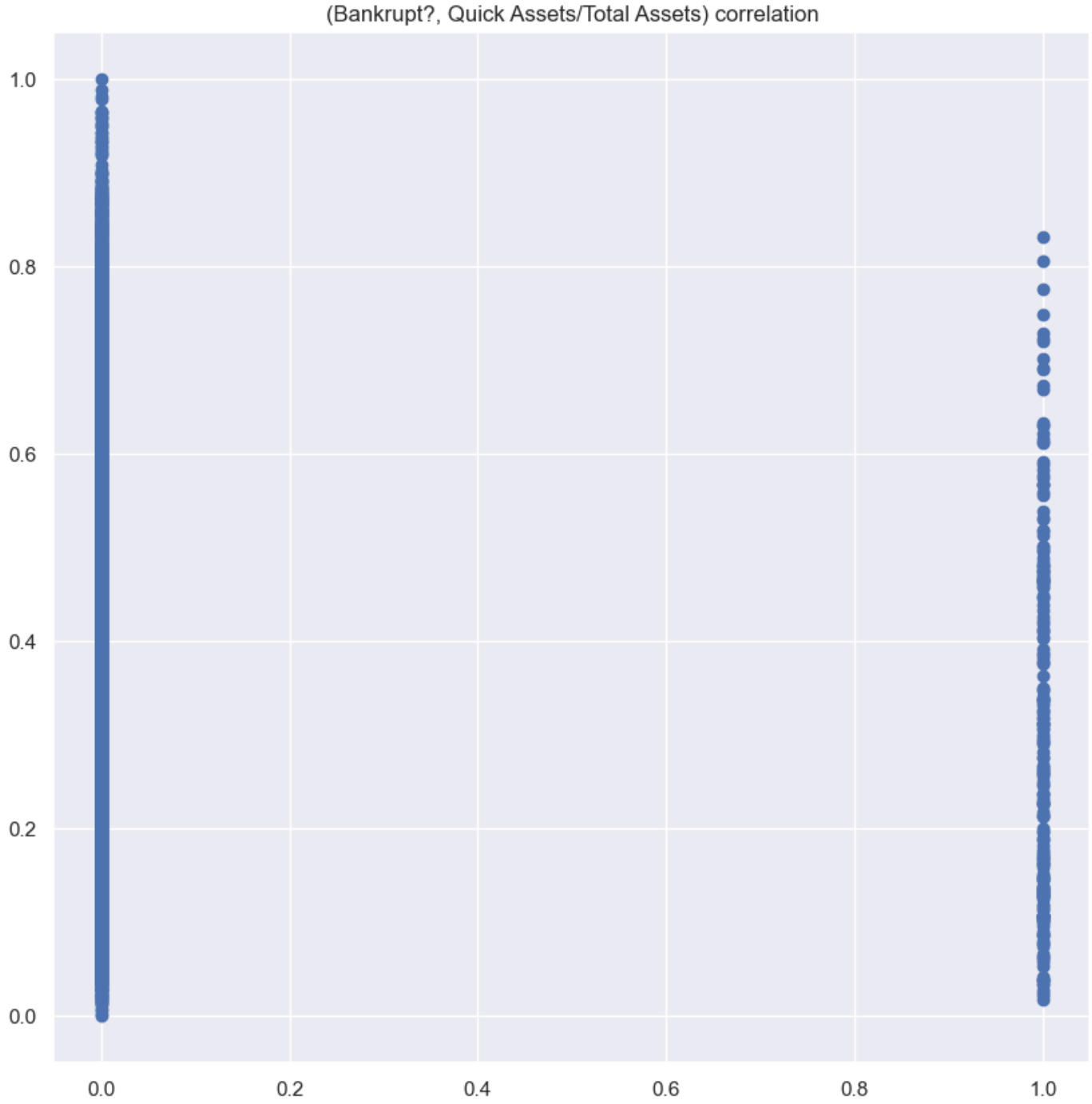

(Bankrupt?, Net Worth Turnover Rate (times)) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

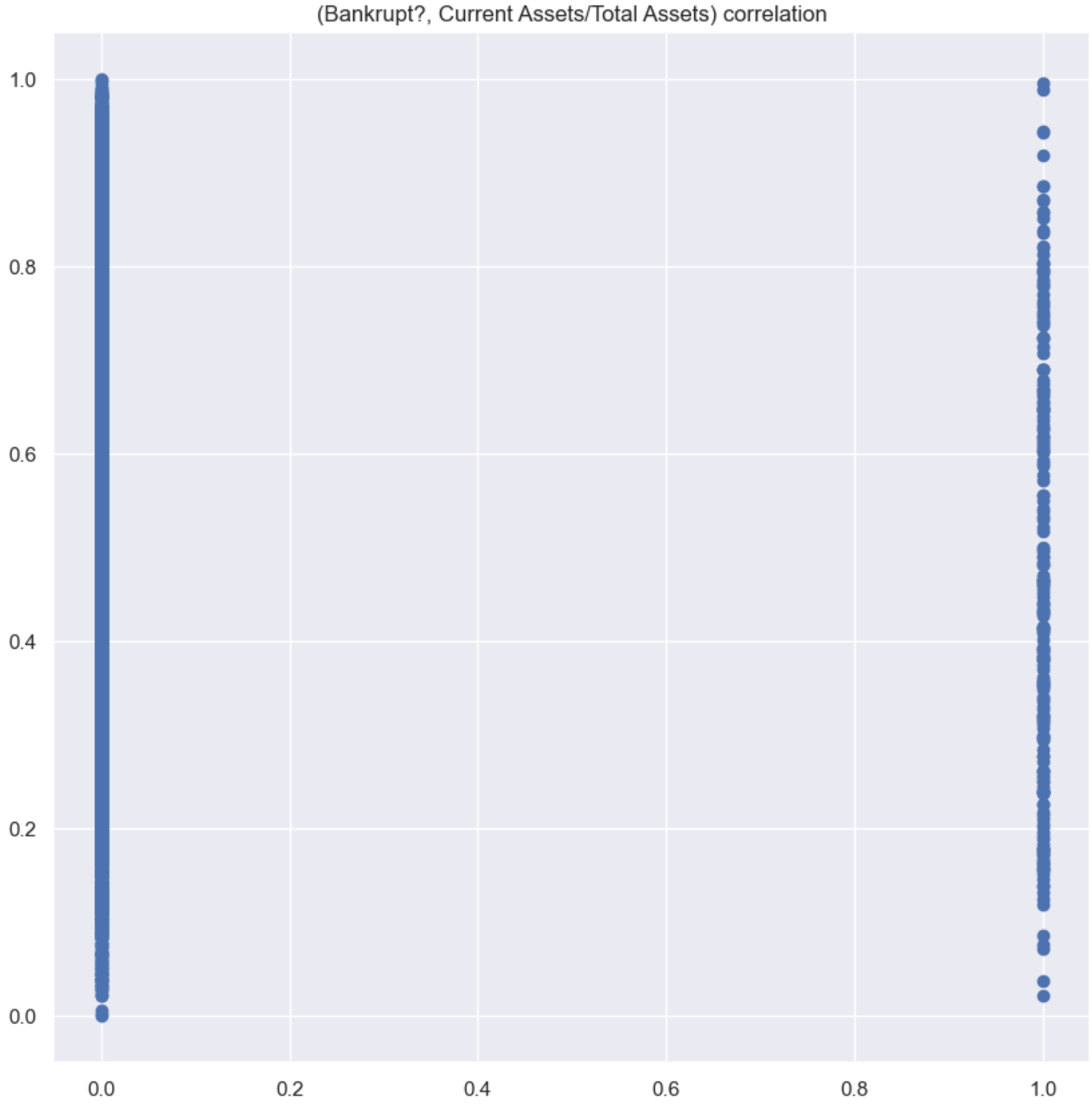

(Bankrupt?, Revenue per person) correlation
1e9
8
6
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0

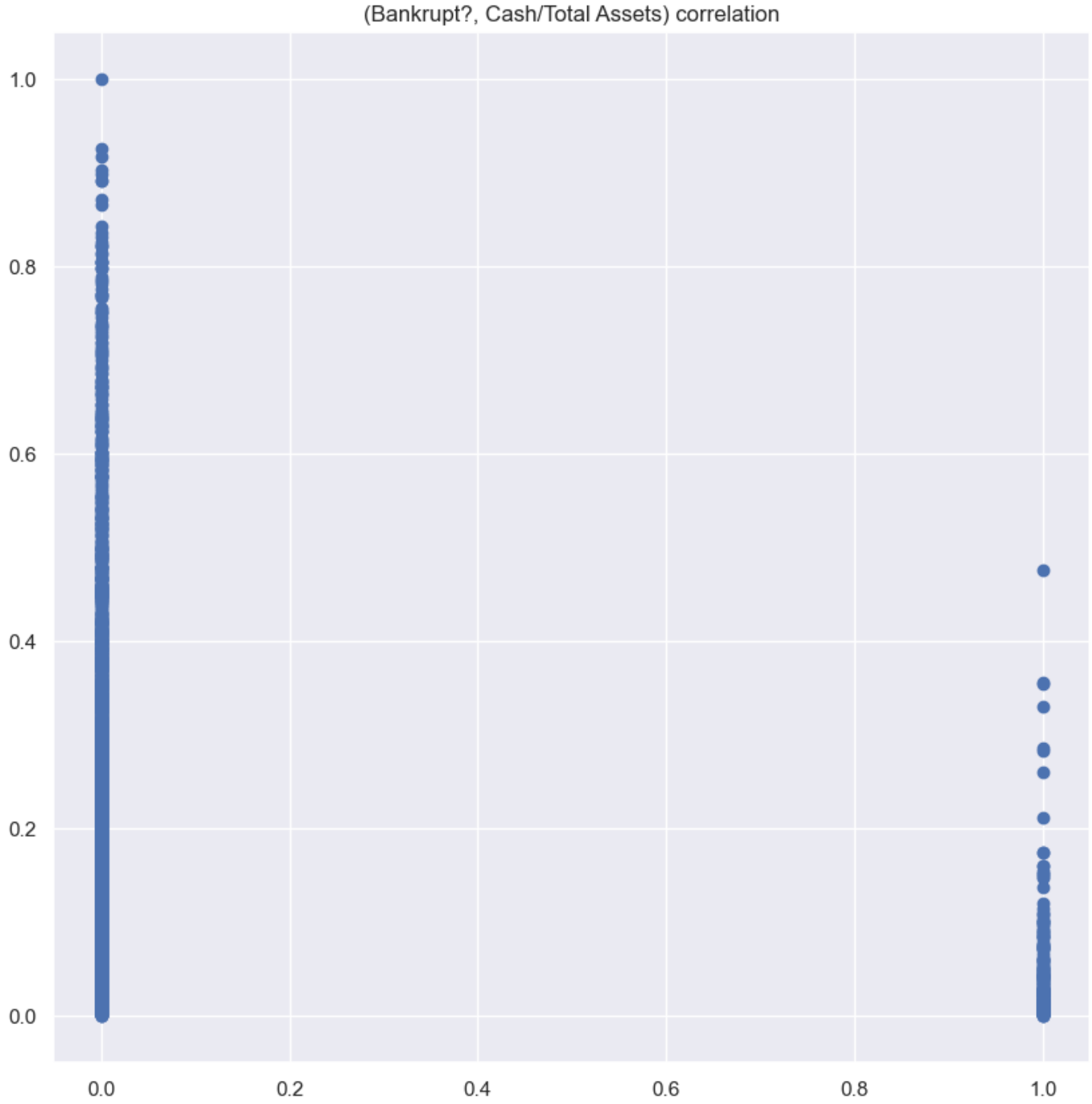

(Bankrupt?, Operating profit per person) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

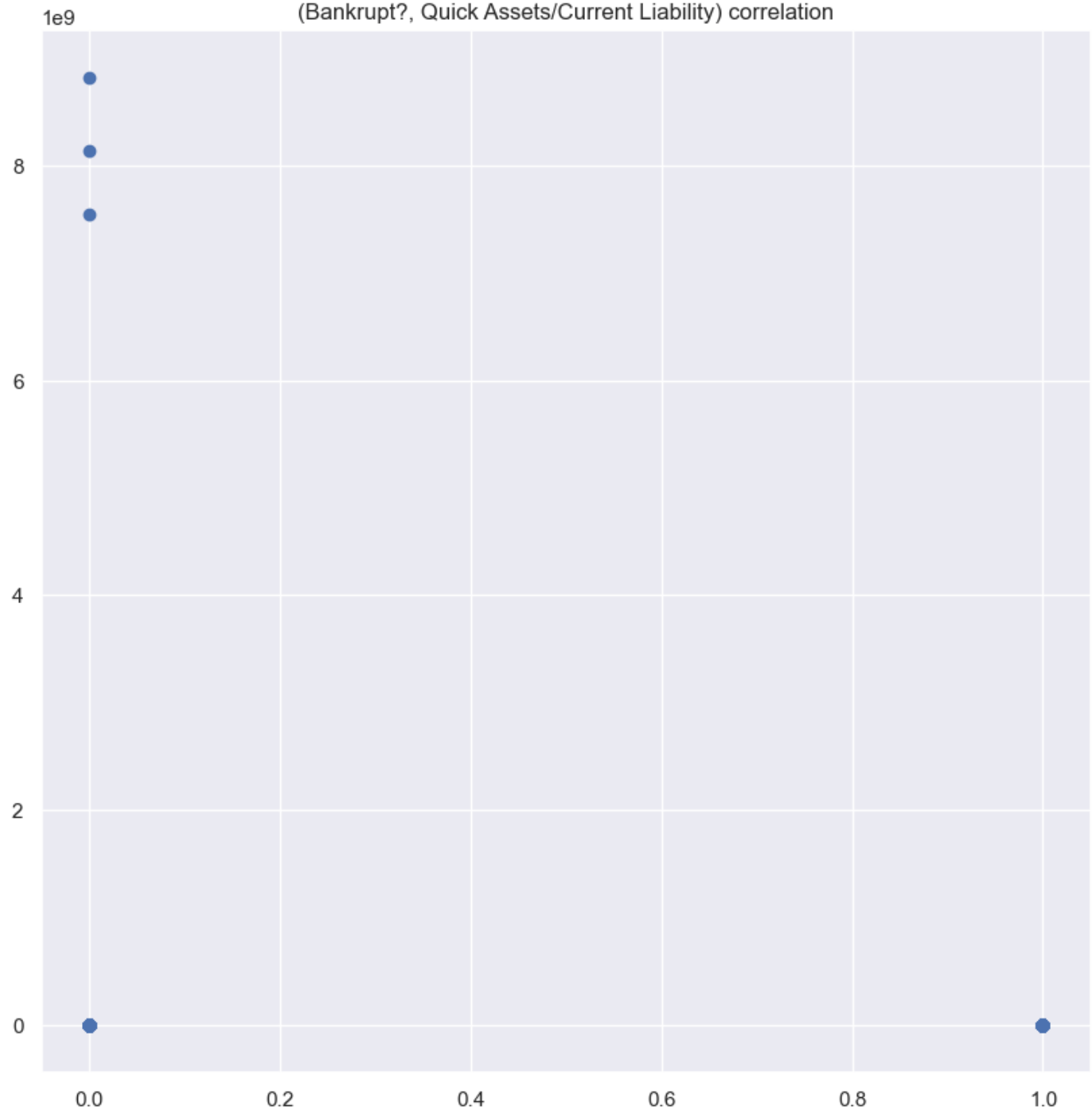

(Bankrupt?, Allocation rate per person) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

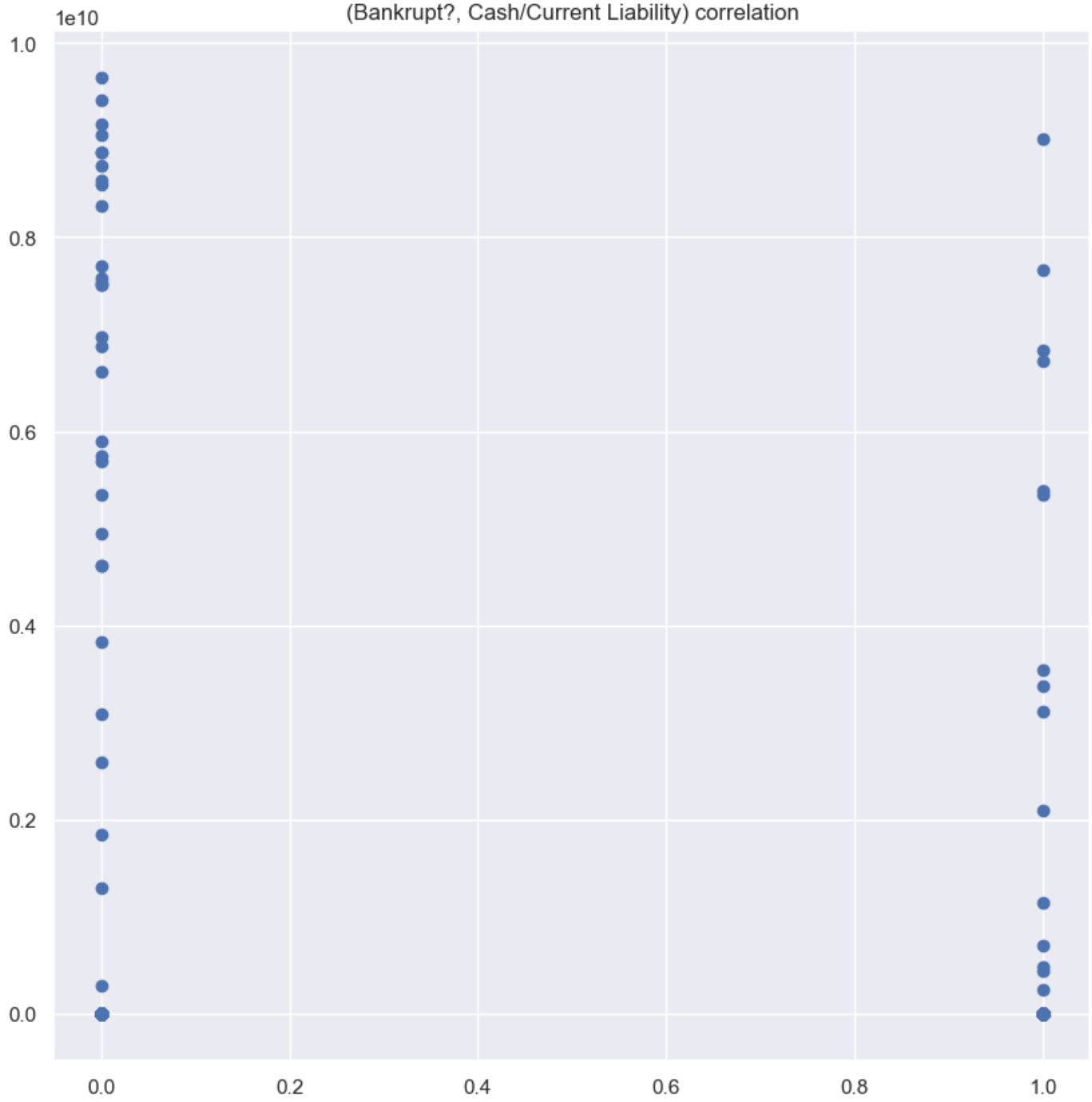

(Bankrupt?, Working Capital to Total Assets) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

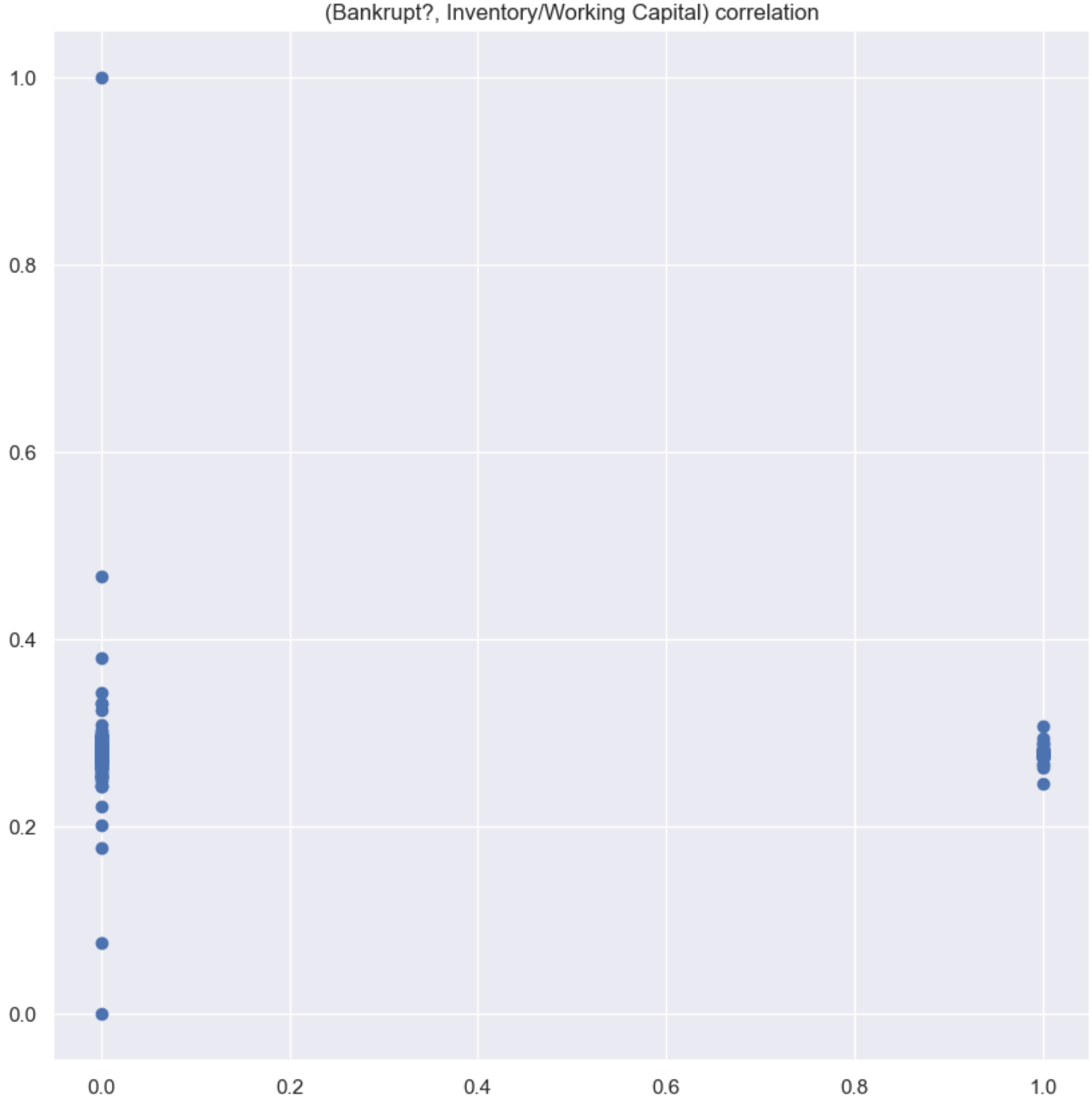

(Bankrupt?, Quick Assets/Total Assets) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

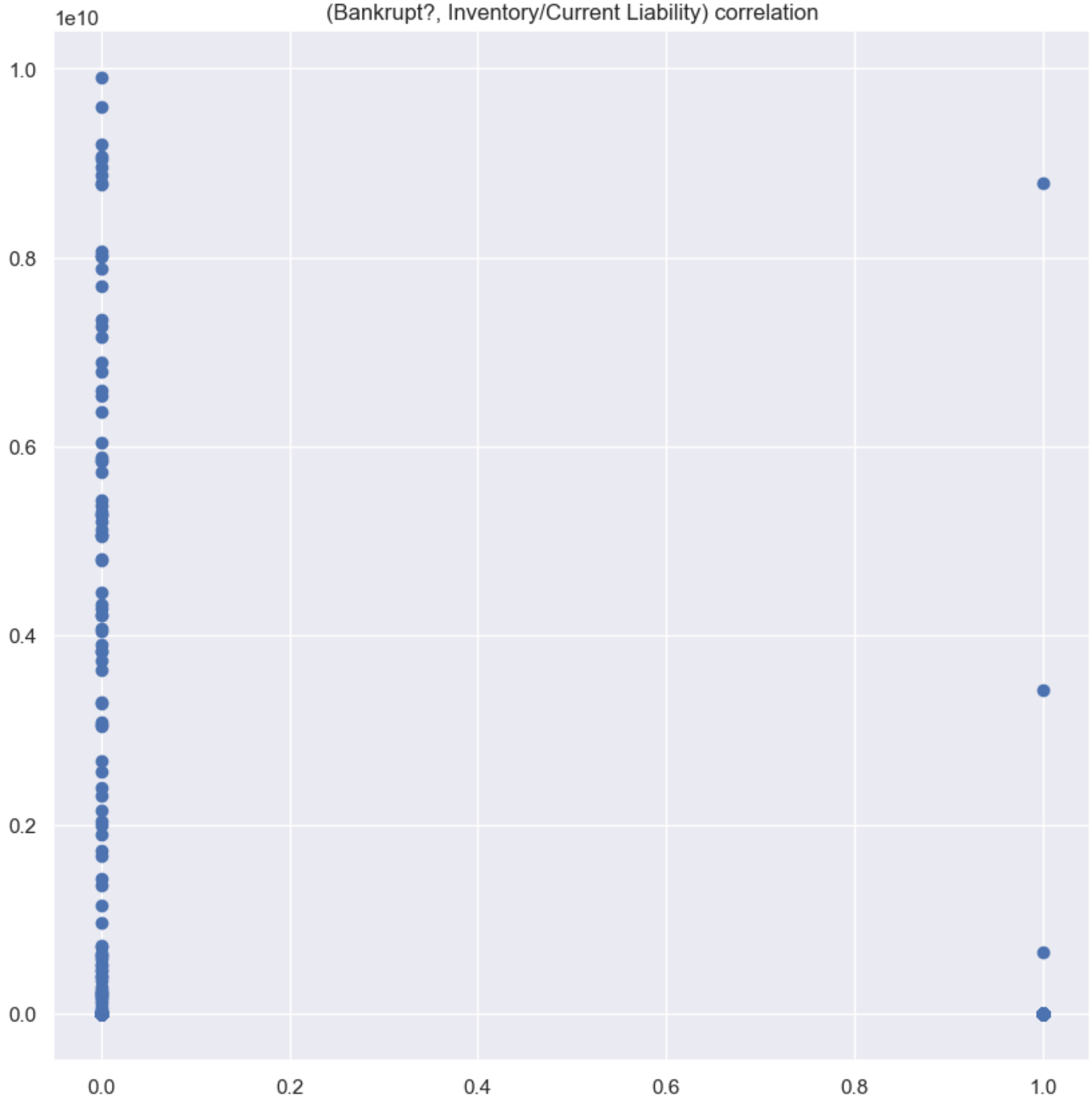

(Bankrupt?, Current Assets/Total Assets) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

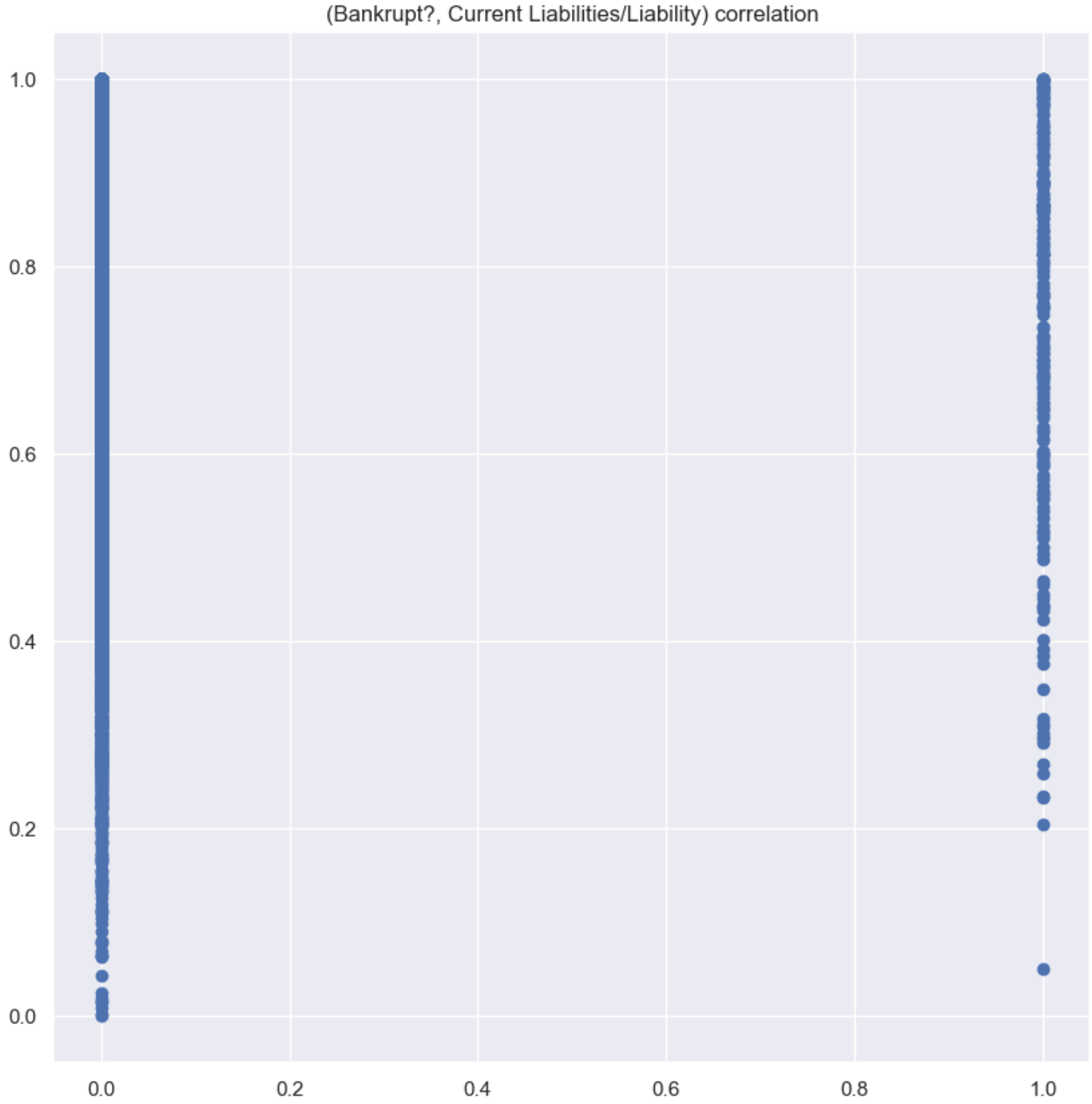

(Bankrupt?, Cash/Total Assets) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

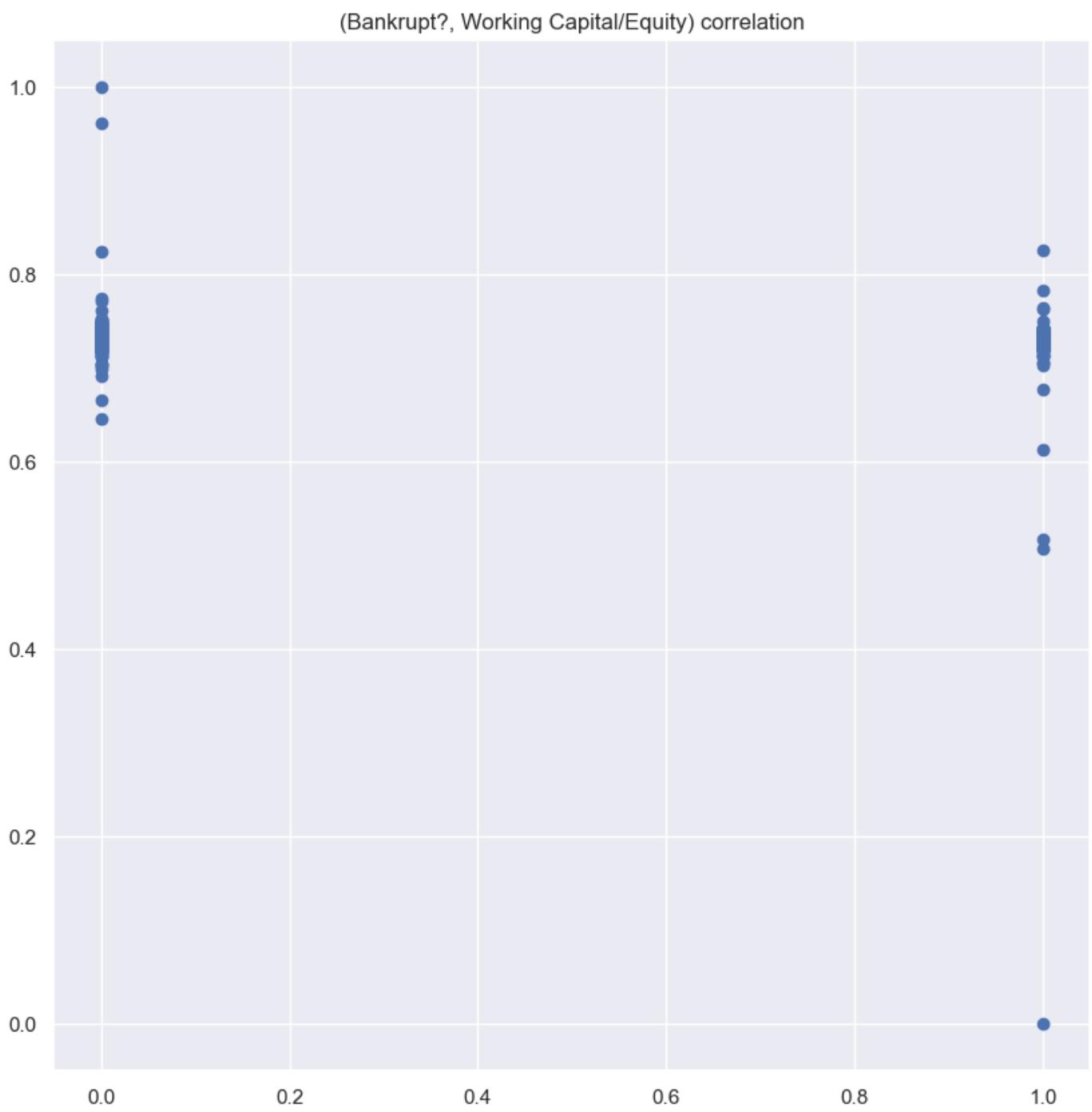

(Bankrupt?, Quick Assets/Current Liability) correlation
1e9
8
6
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0

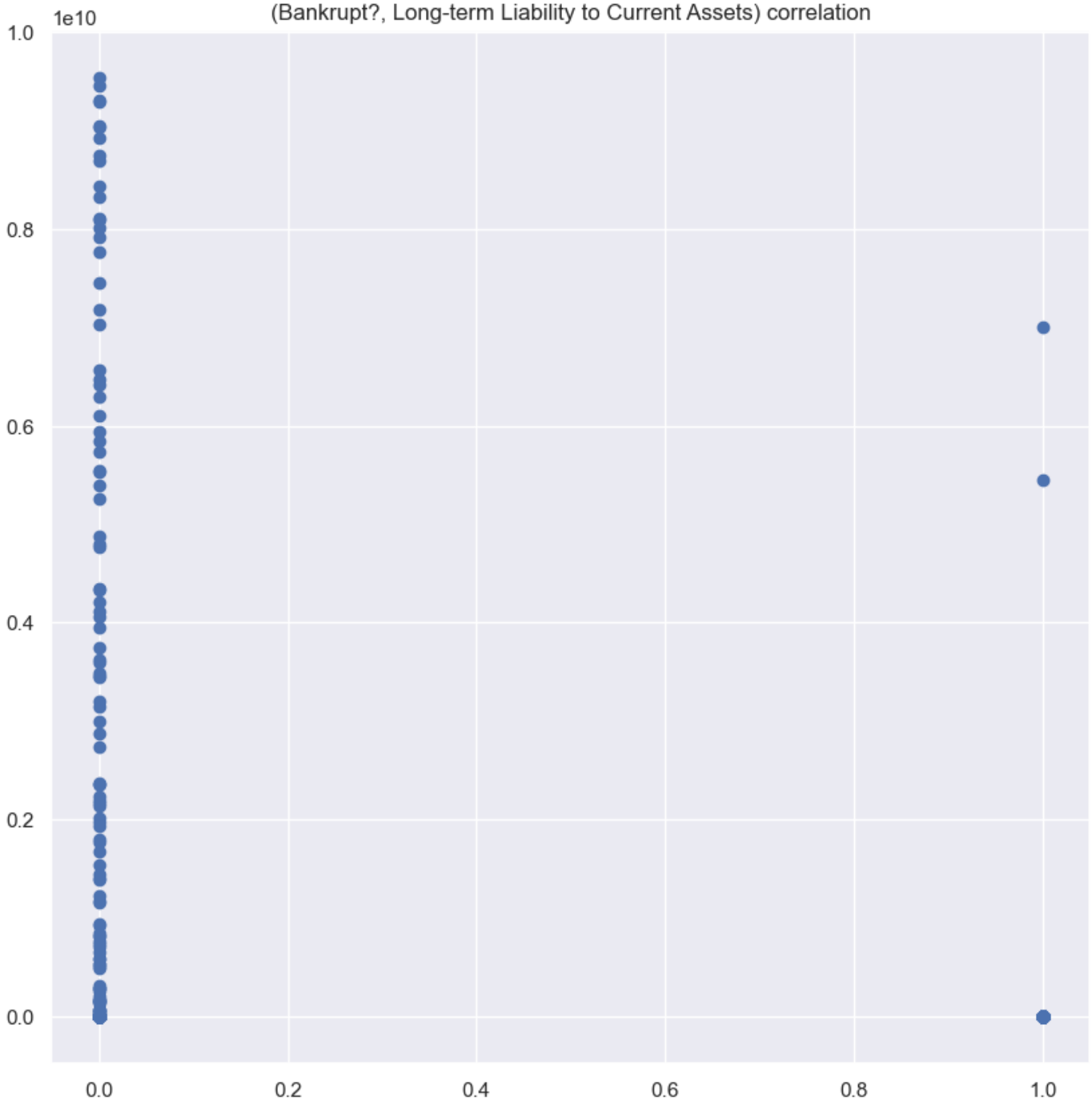

(Bankrupt?, Cash/Current Liability) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

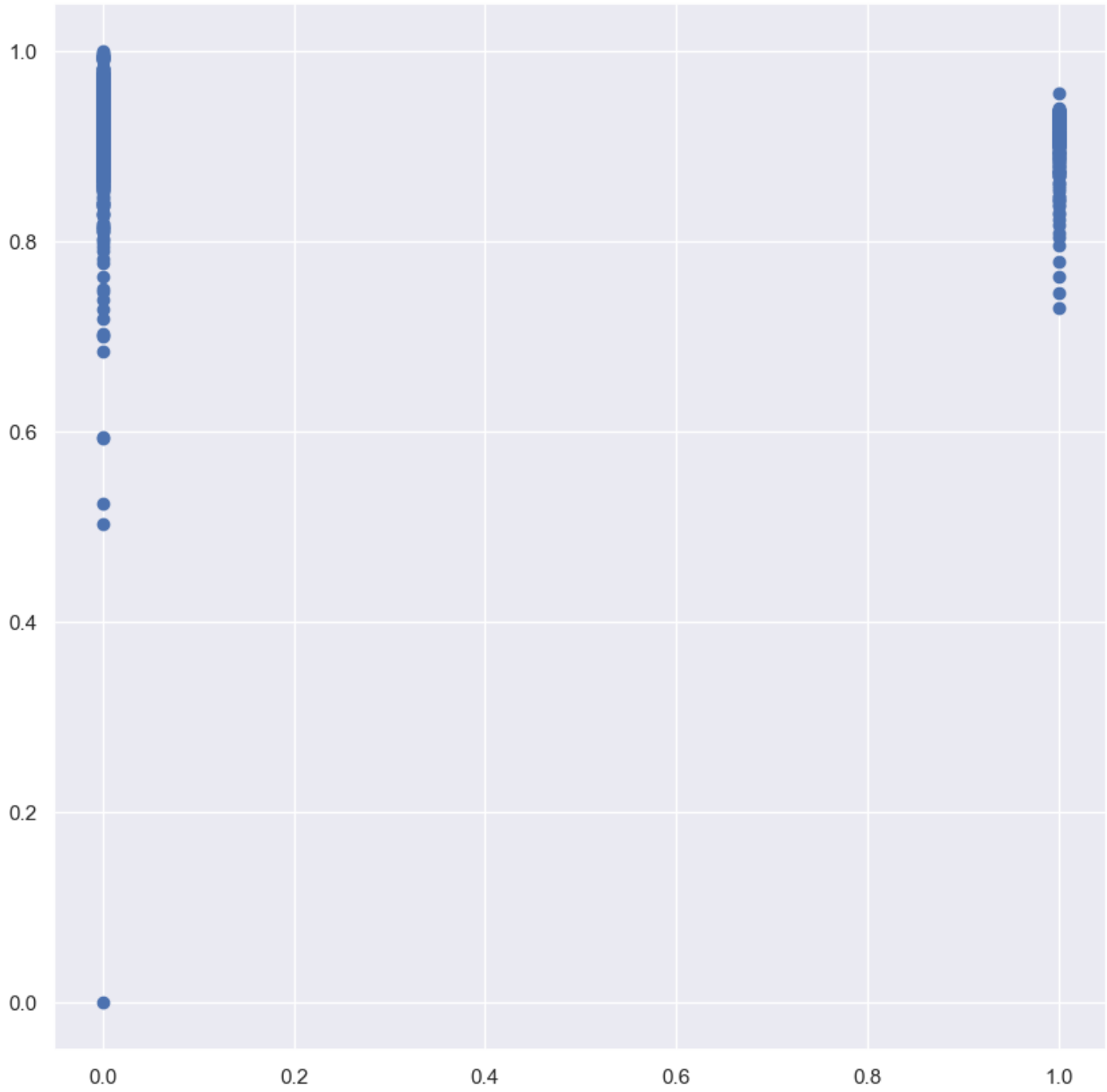

(Bankrupt?, Inventory/Working Capital) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

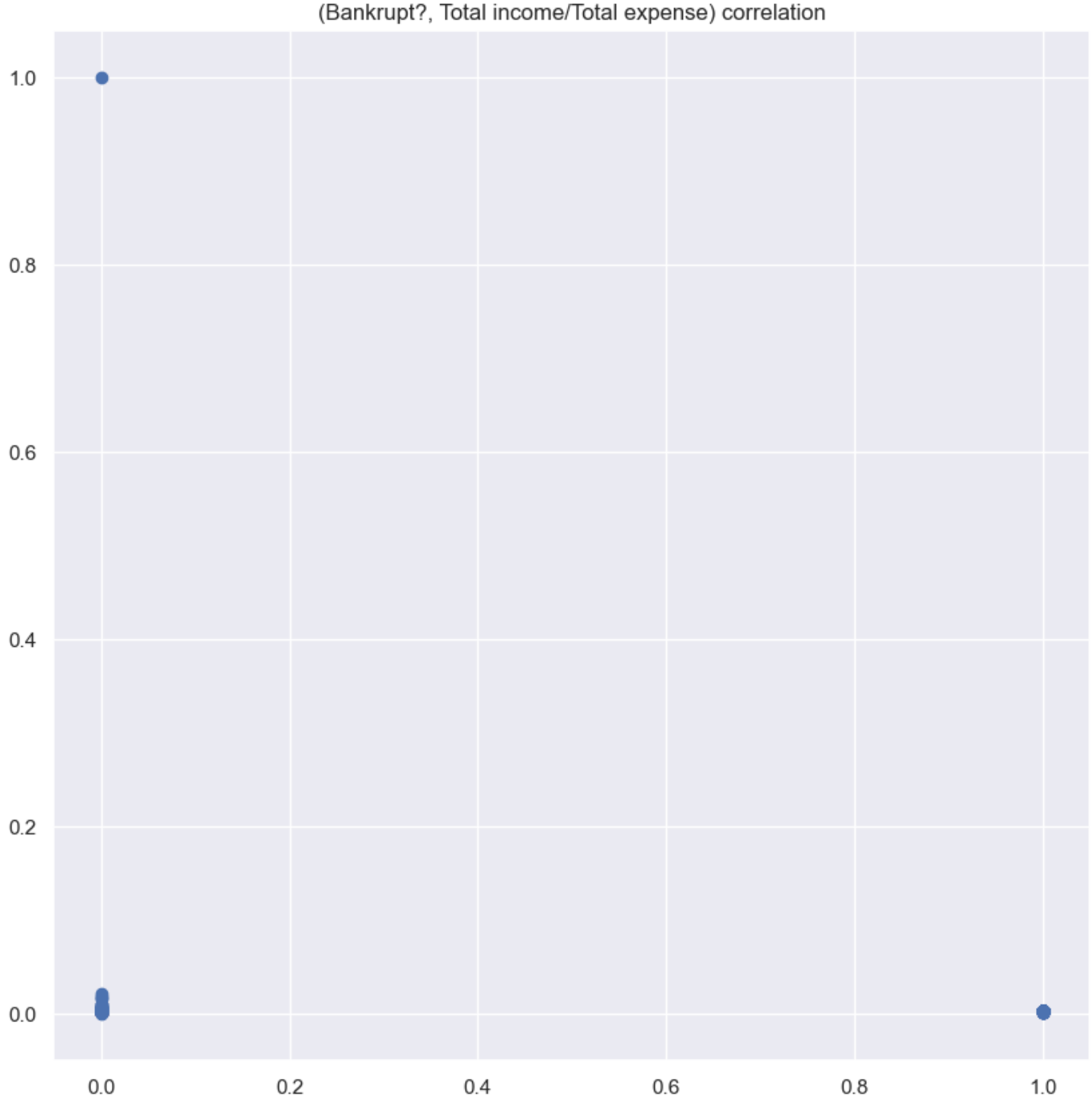

(Bankrupt?, Inventory/Current Liability) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

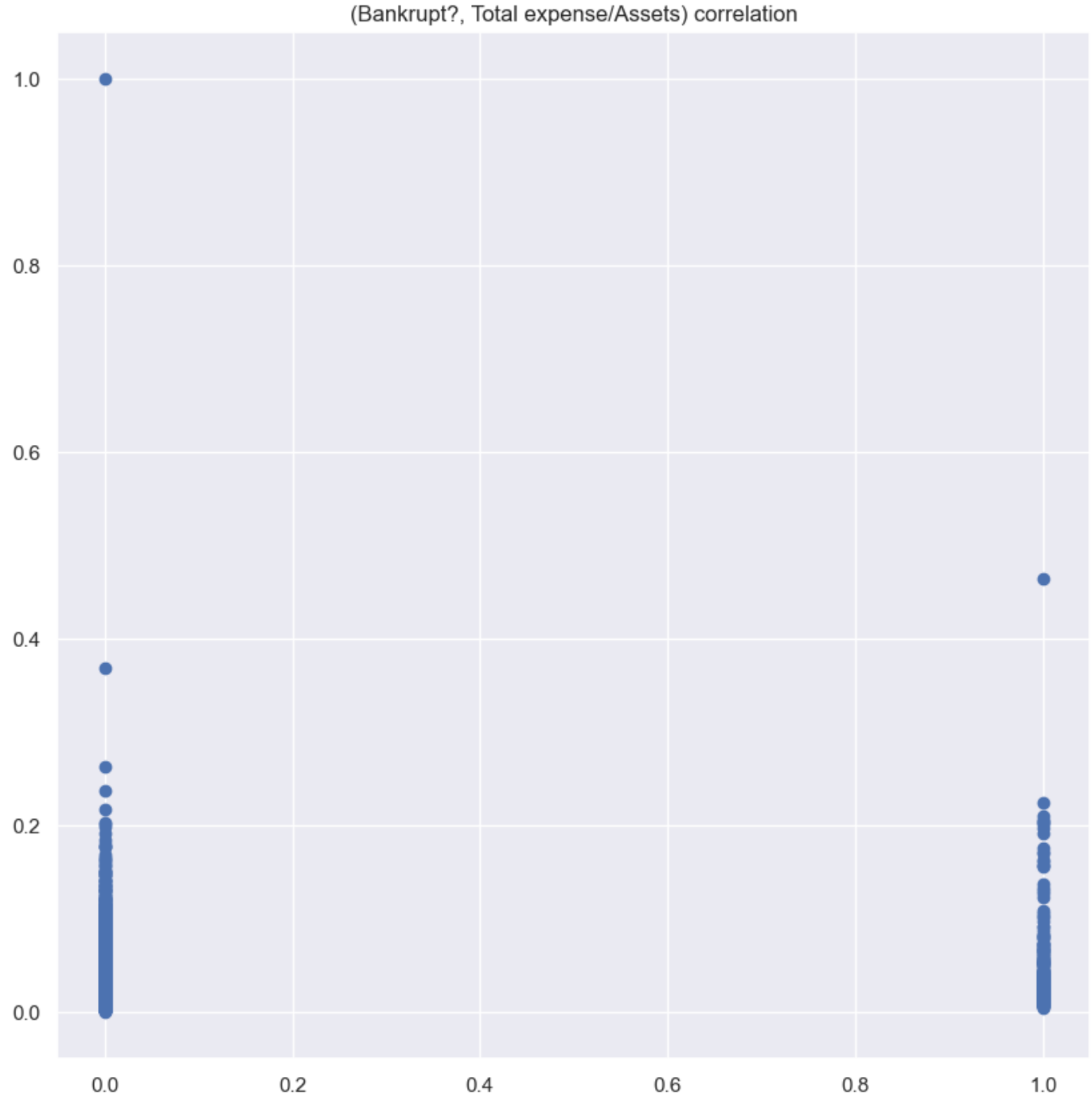

(Bankrupt?, Current Liabilities/Liability) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

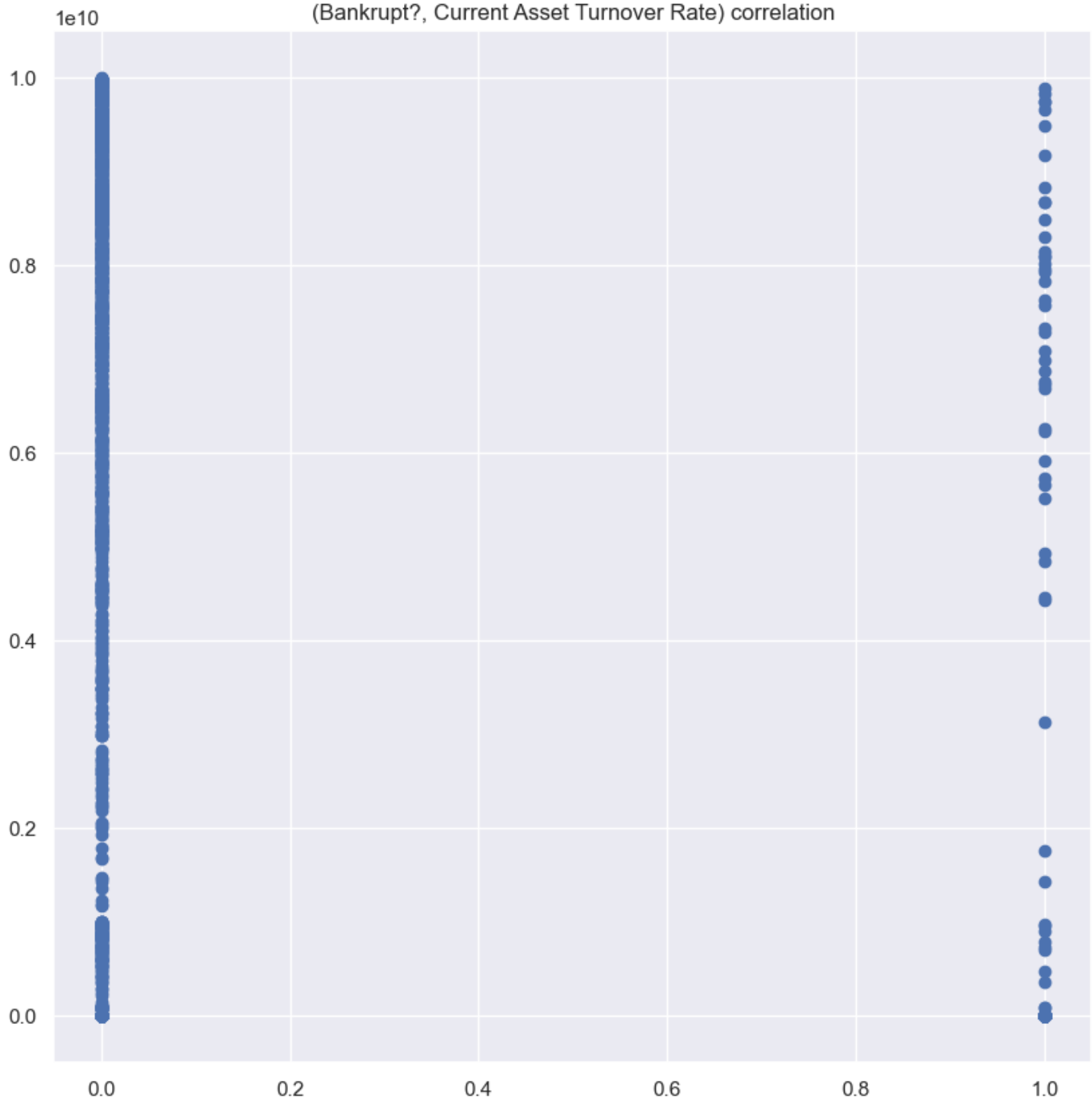

(Bankrupt?, Working Capital/Equity) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

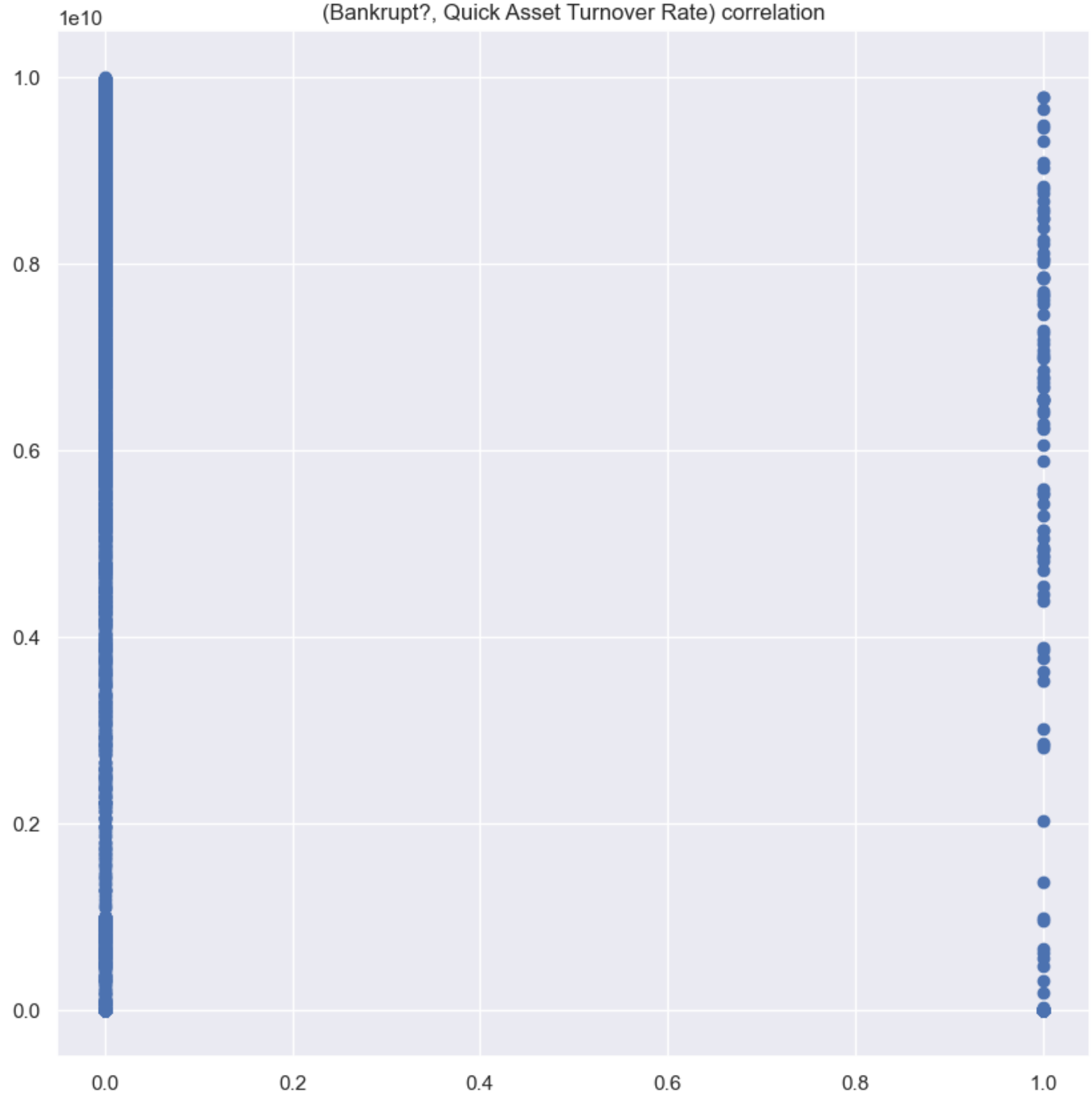

(Bankrupt?, Long-term Liability to Current Assets) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

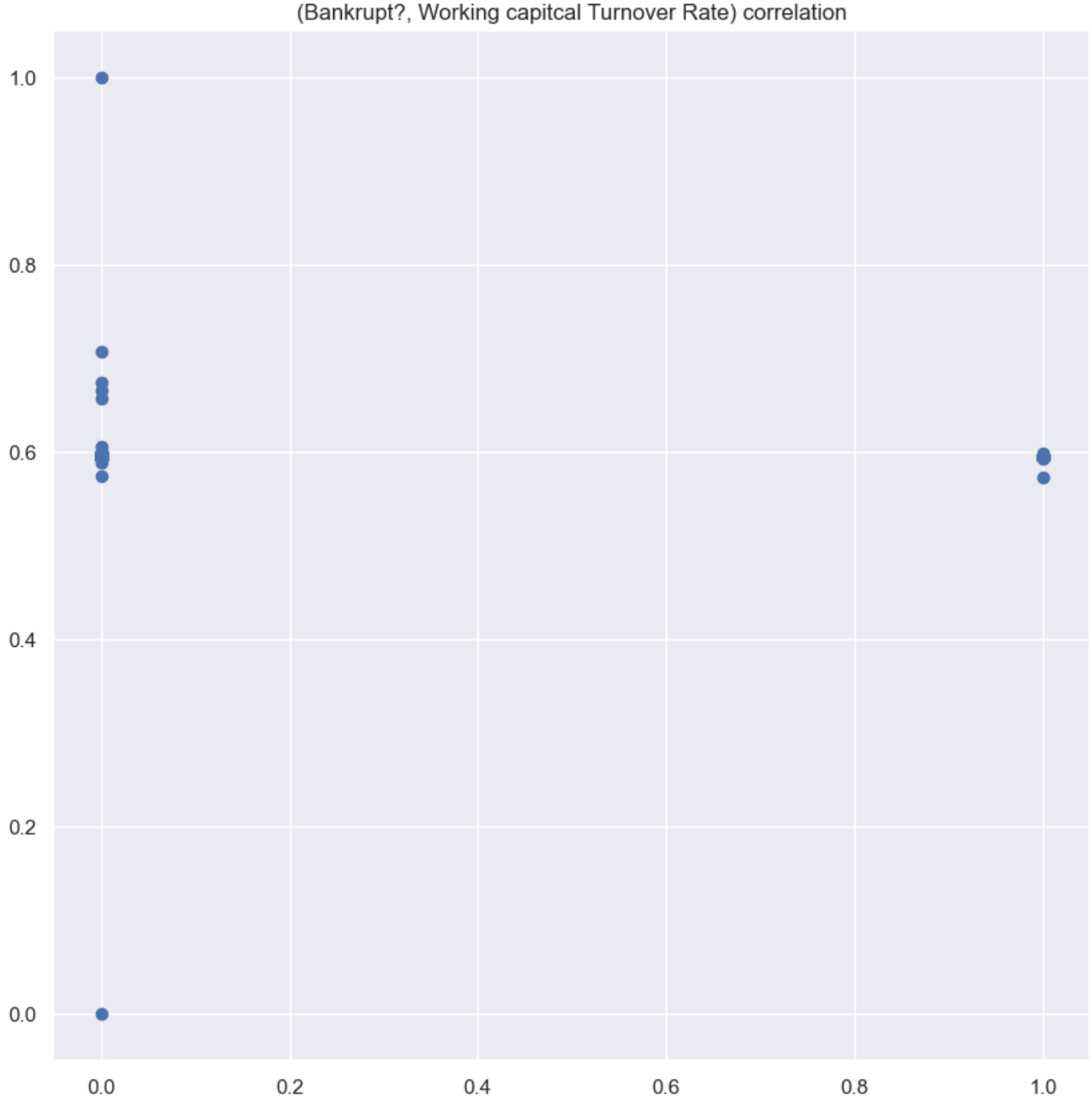

(Bankrupt?, Retained Earnings to Total Assets) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

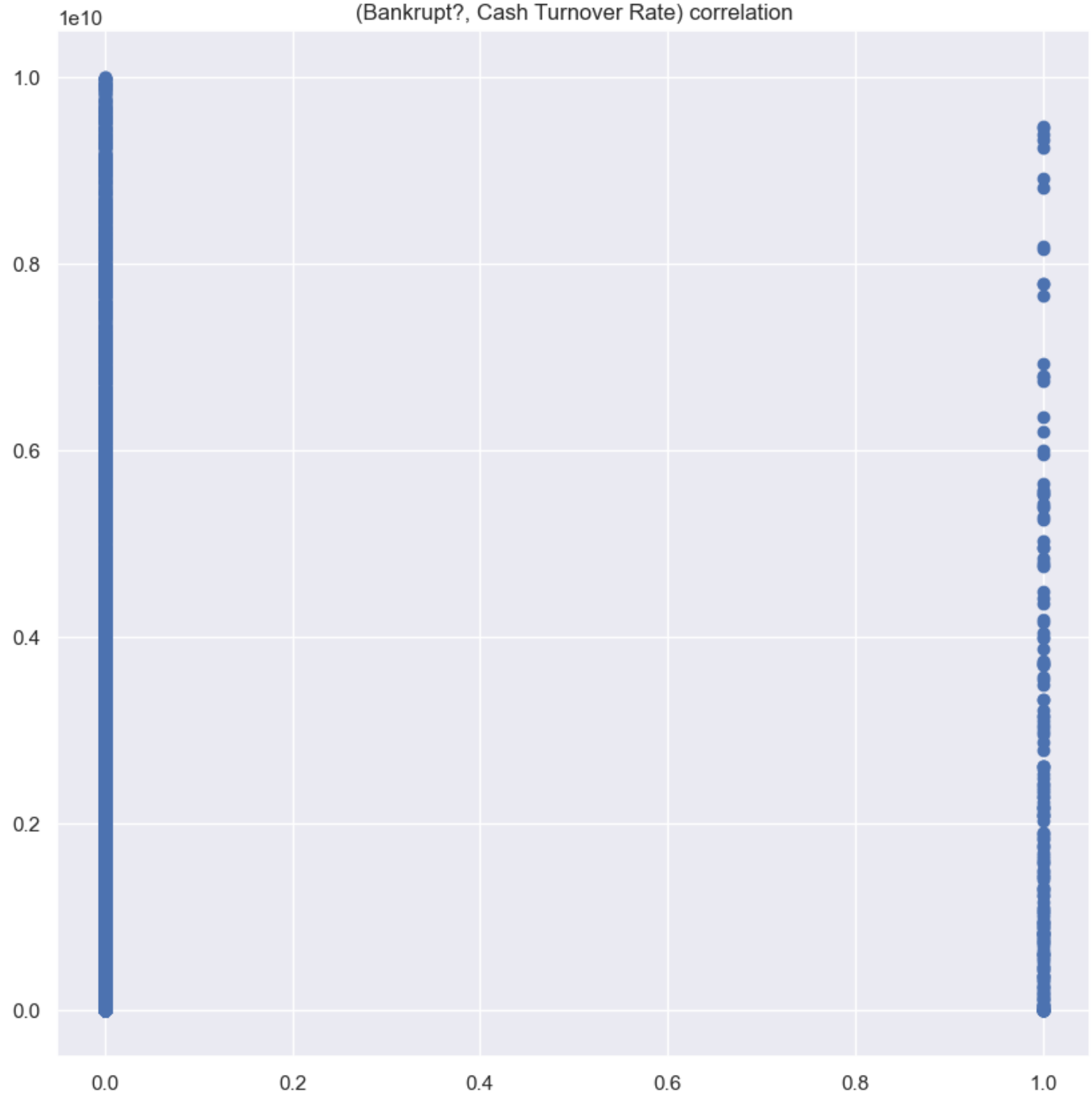

(Bankrupt?, Total income/Total expense) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

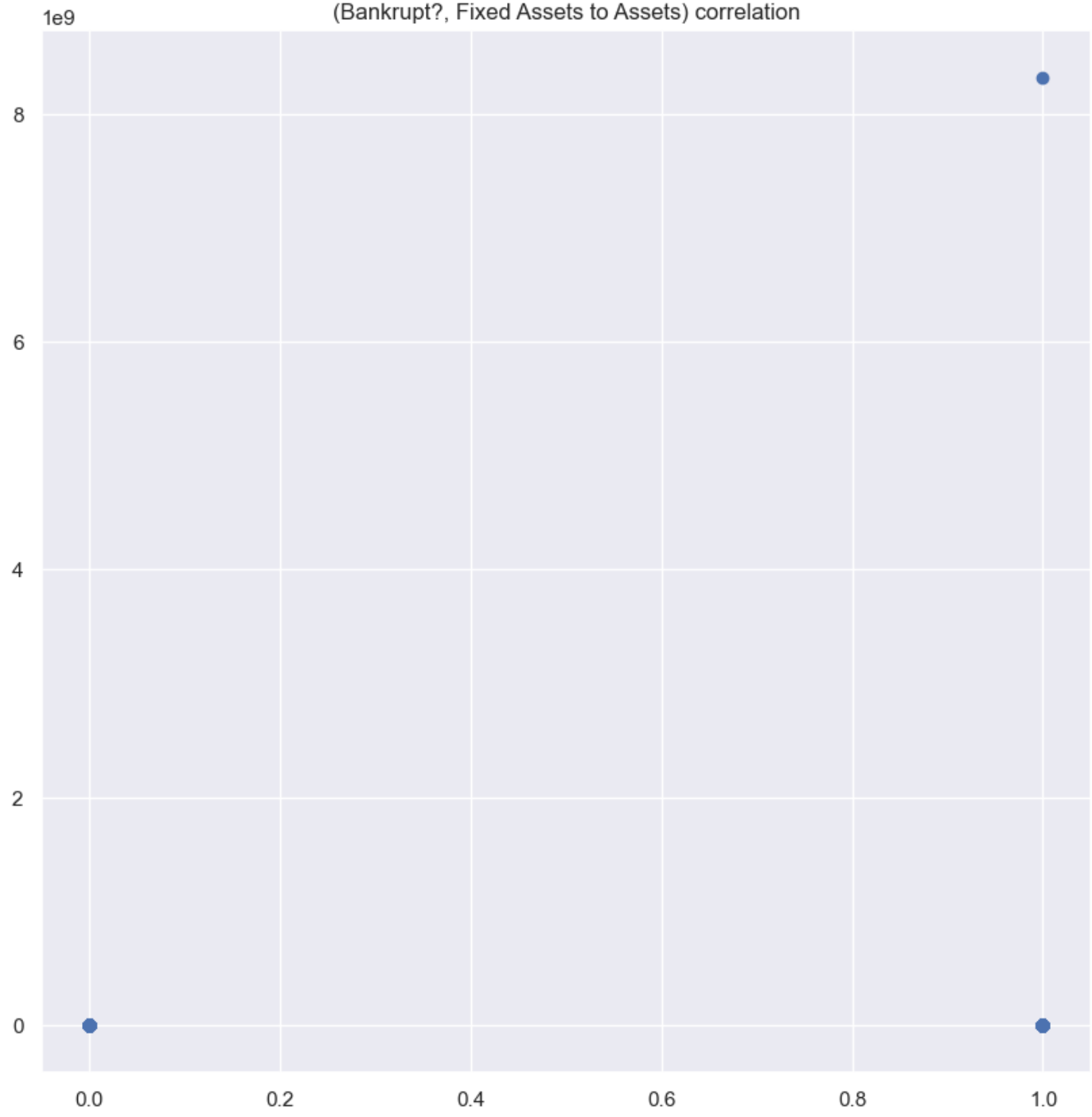

(Bankrupt?, Total expense/Assets) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

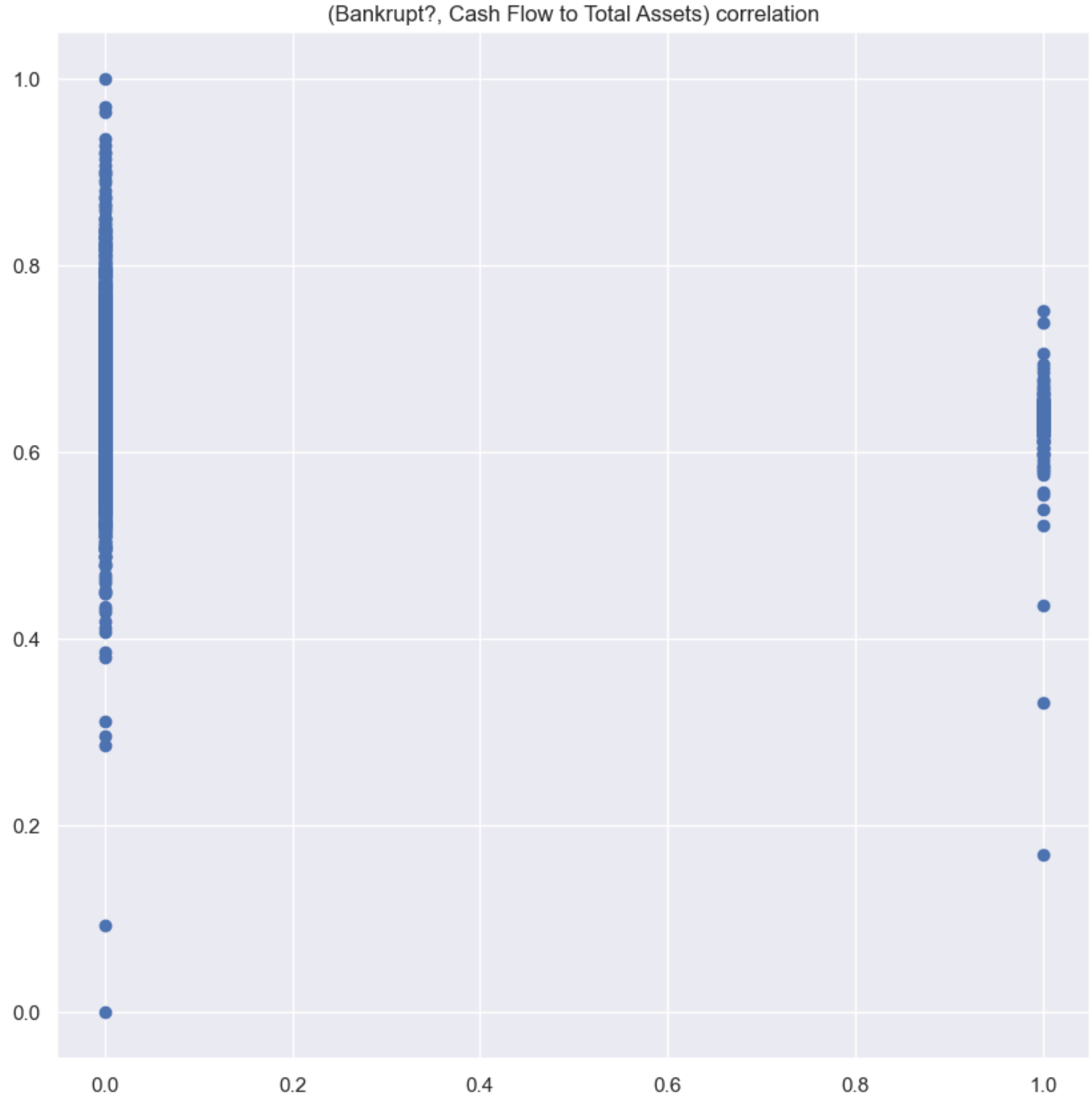

(Bankrupt?, Current Asset Turnover Rate) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

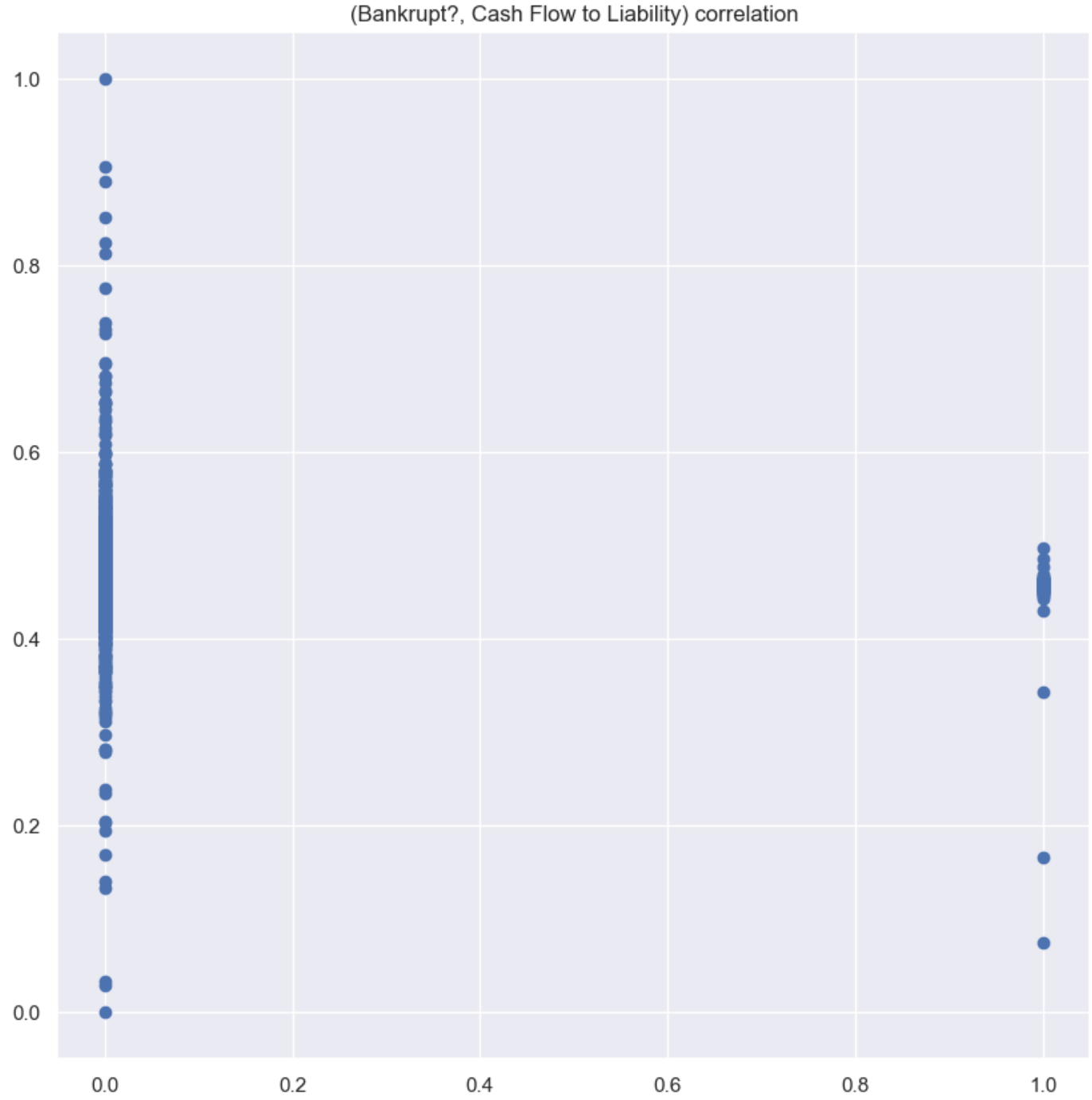

(Bankrupt?, Quick Asset Turnover Rate) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

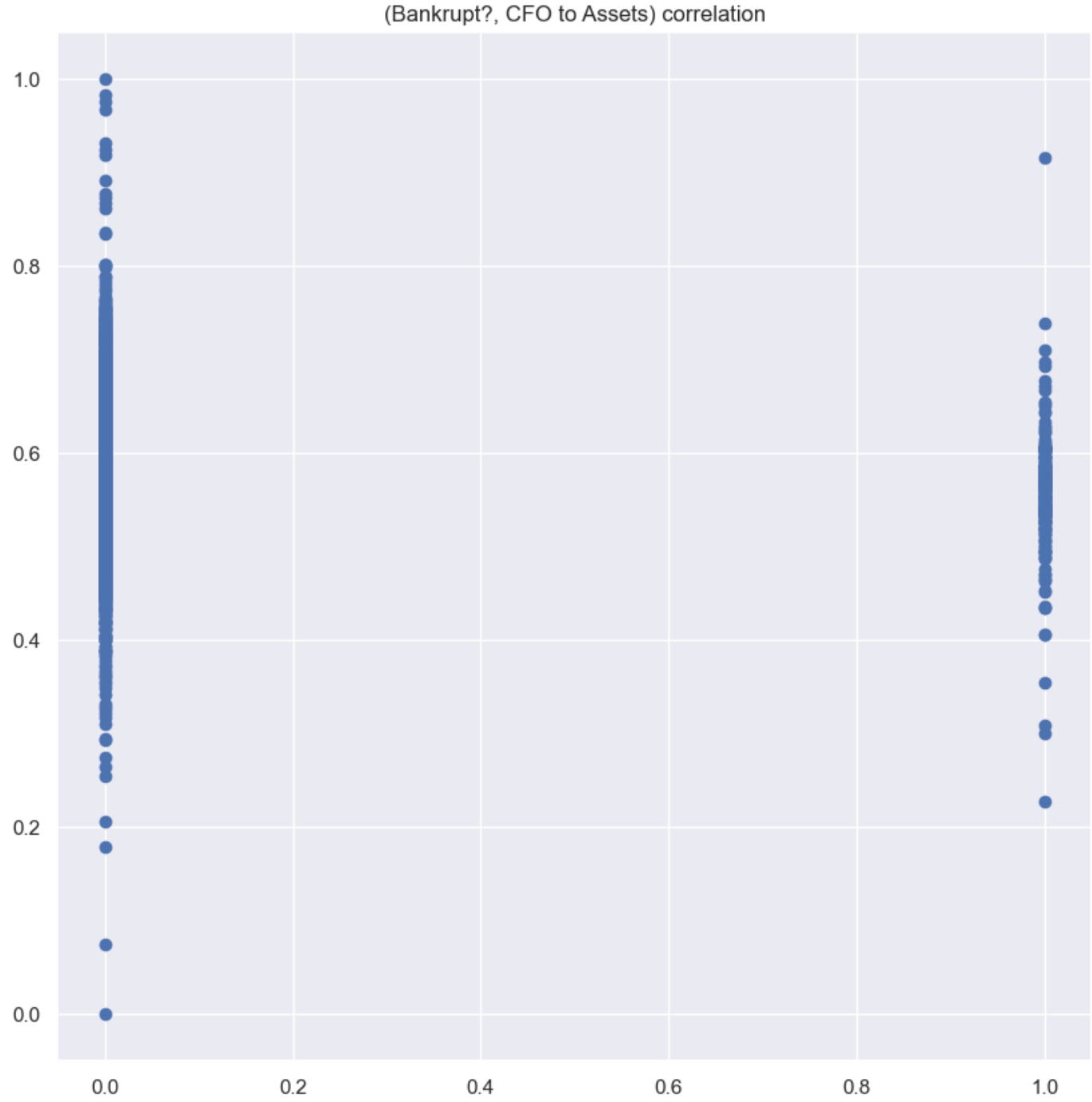

(Bankrupt?, Working capitcal Turnover Rate) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

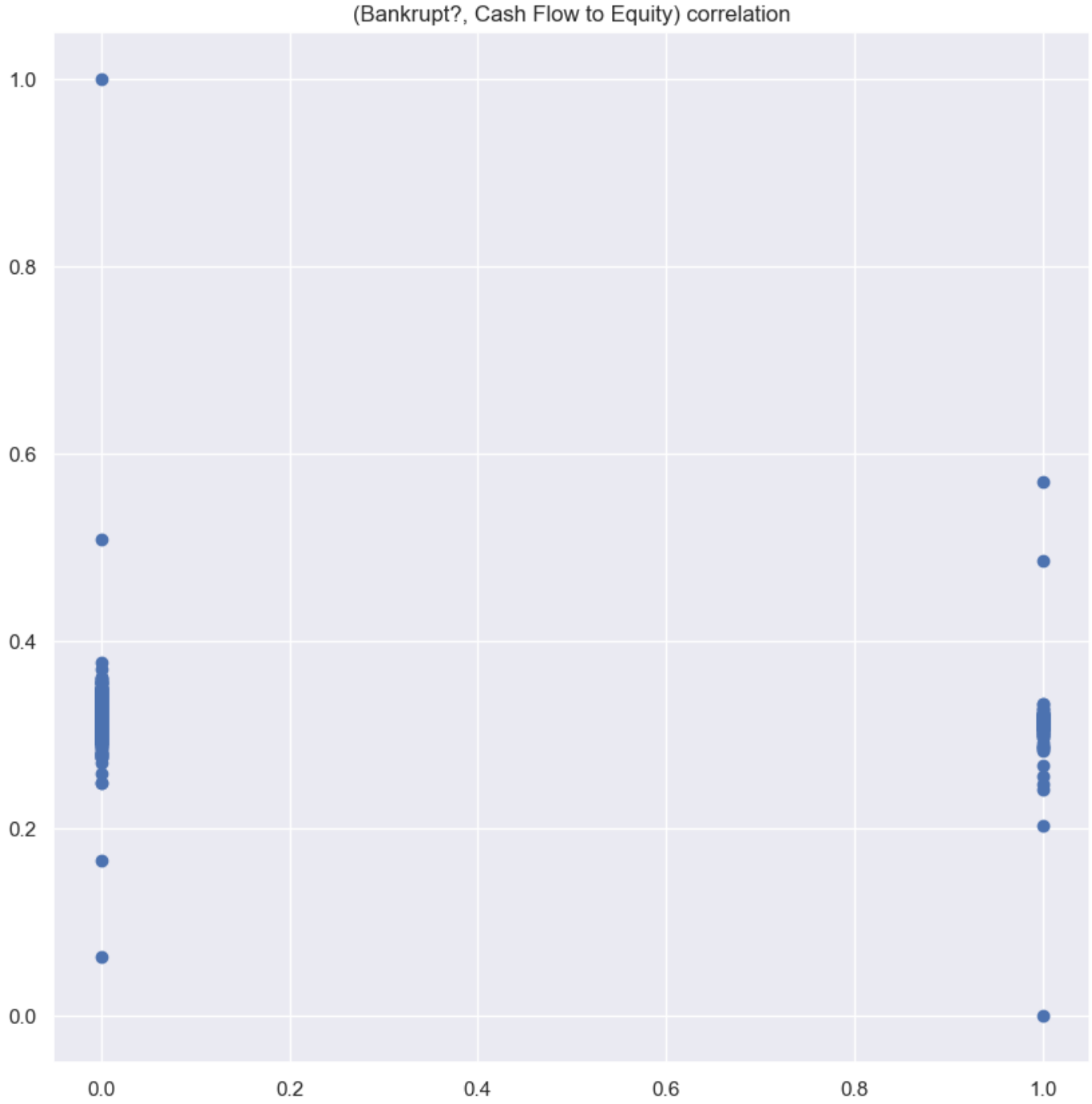

(Bankrupt?, Cash Turnover Rate) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

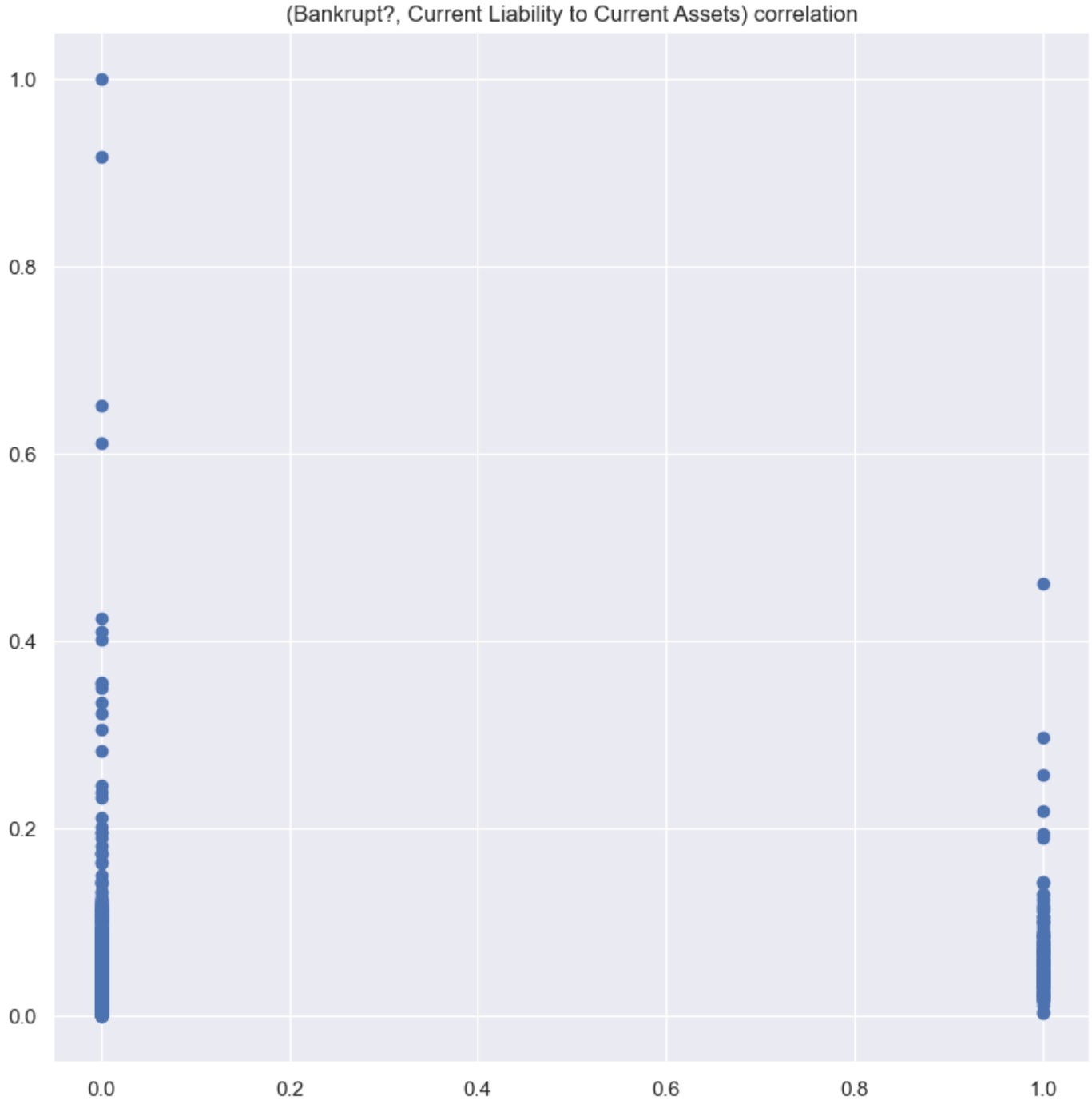

(Bankrupt?, Fixed Assets to Assets) correlation
1e9
8
6
4
2
0
0.0
0.2
0.4
0.6
0.8
1.0

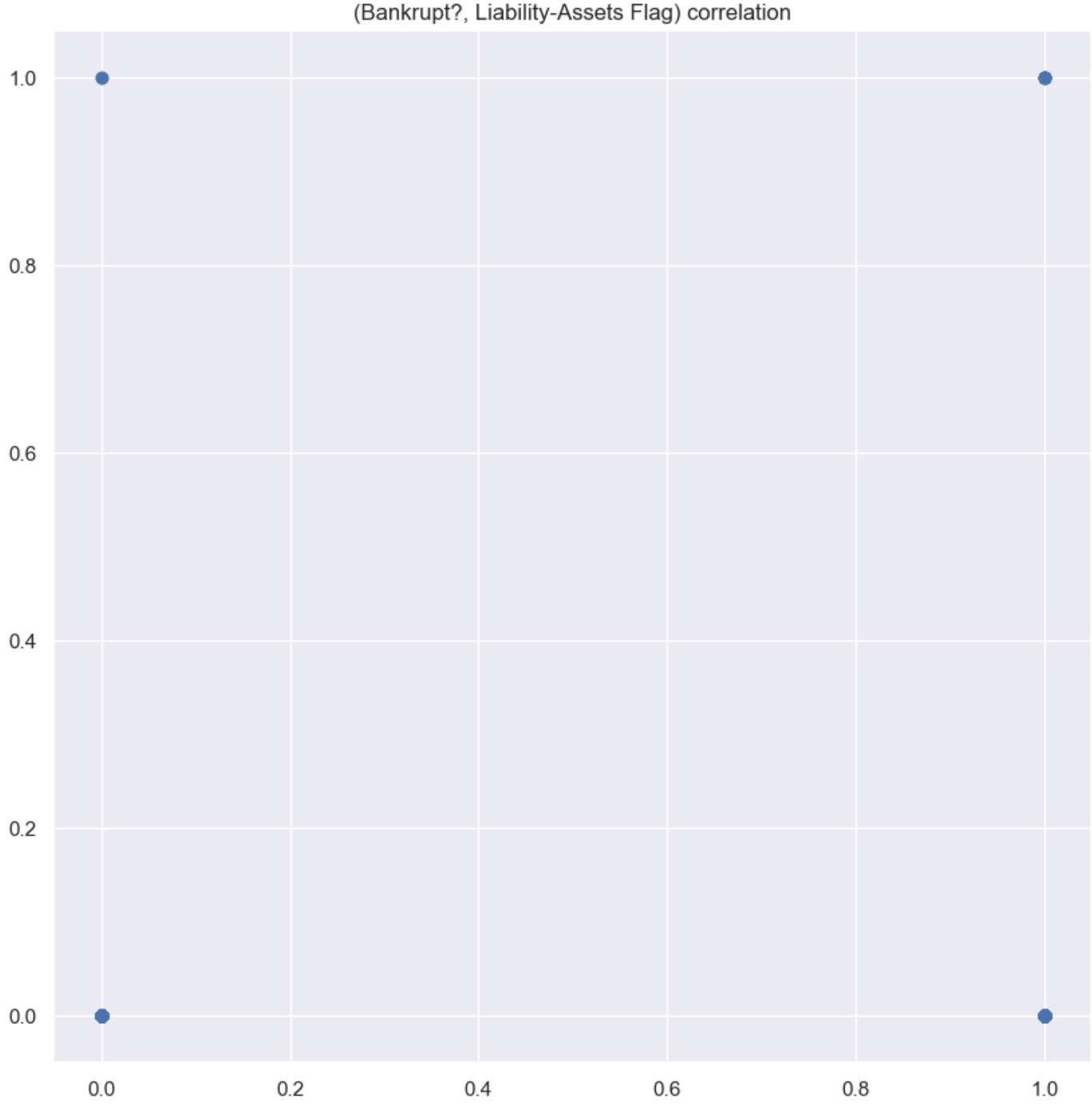

(Bankrupt?, Cash Flow to Total Assets) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

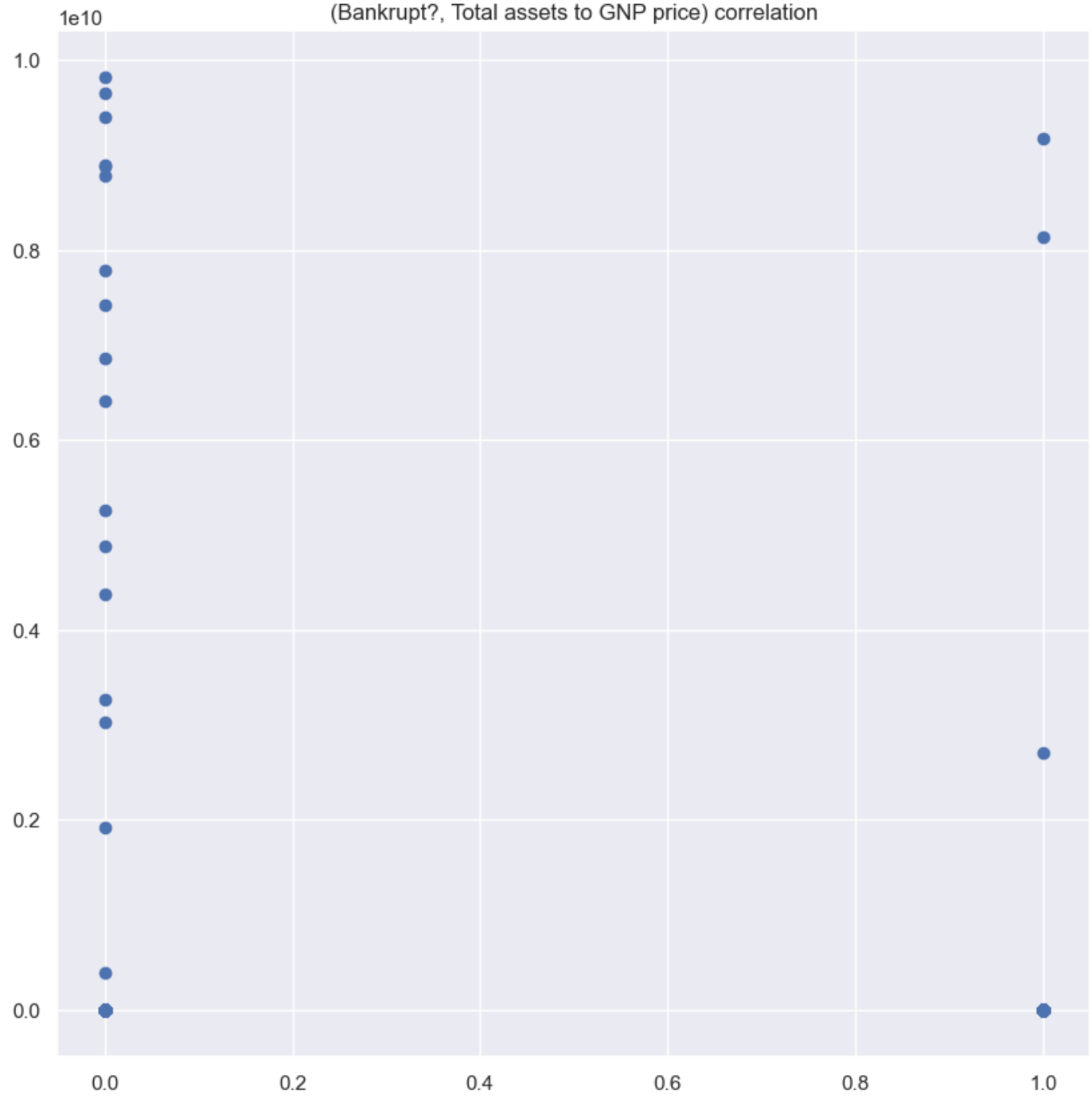

(Bankrupt?, Cash Flow to Liability) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

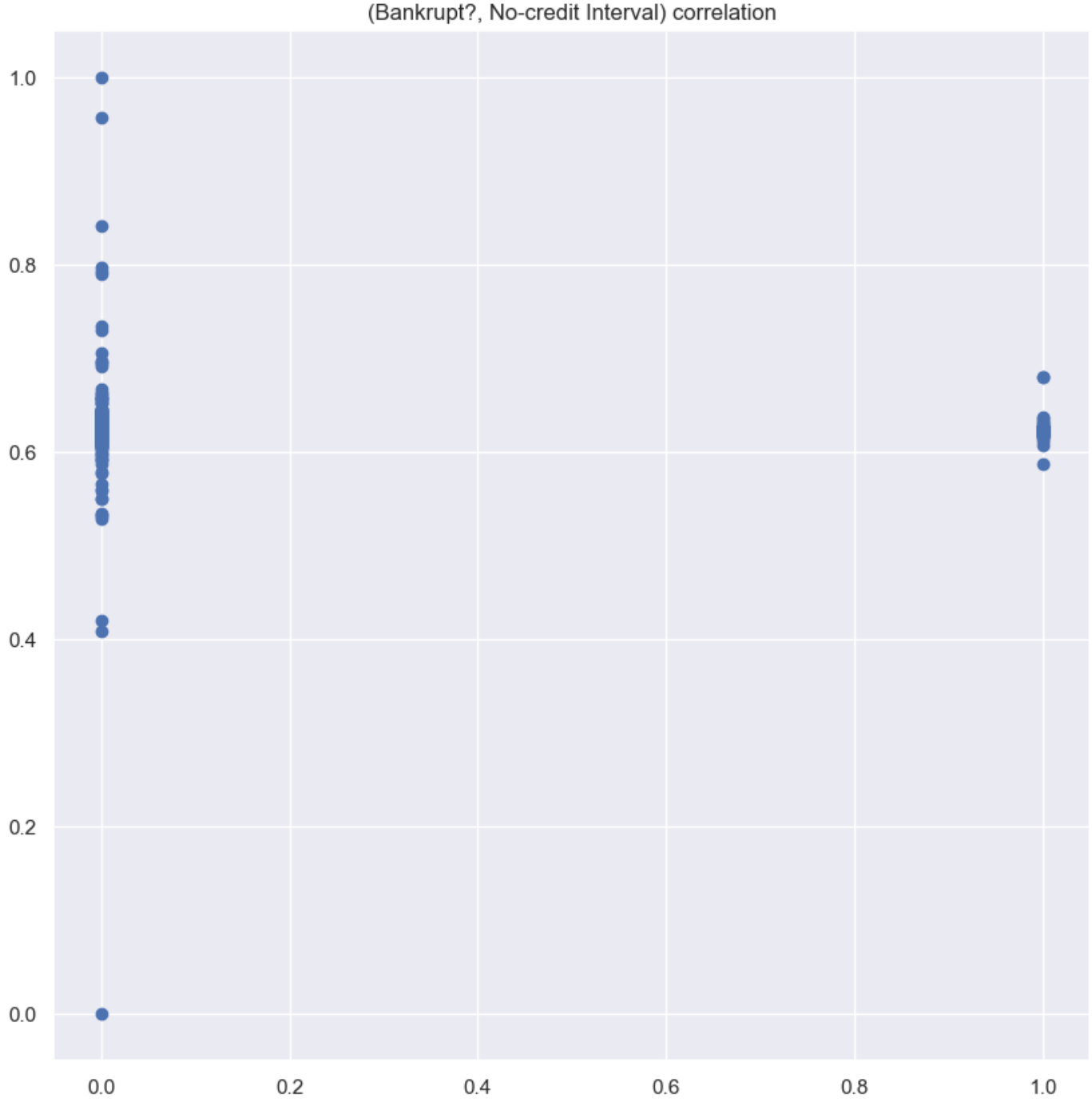

(Bankrupt?, CFO to Assets) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

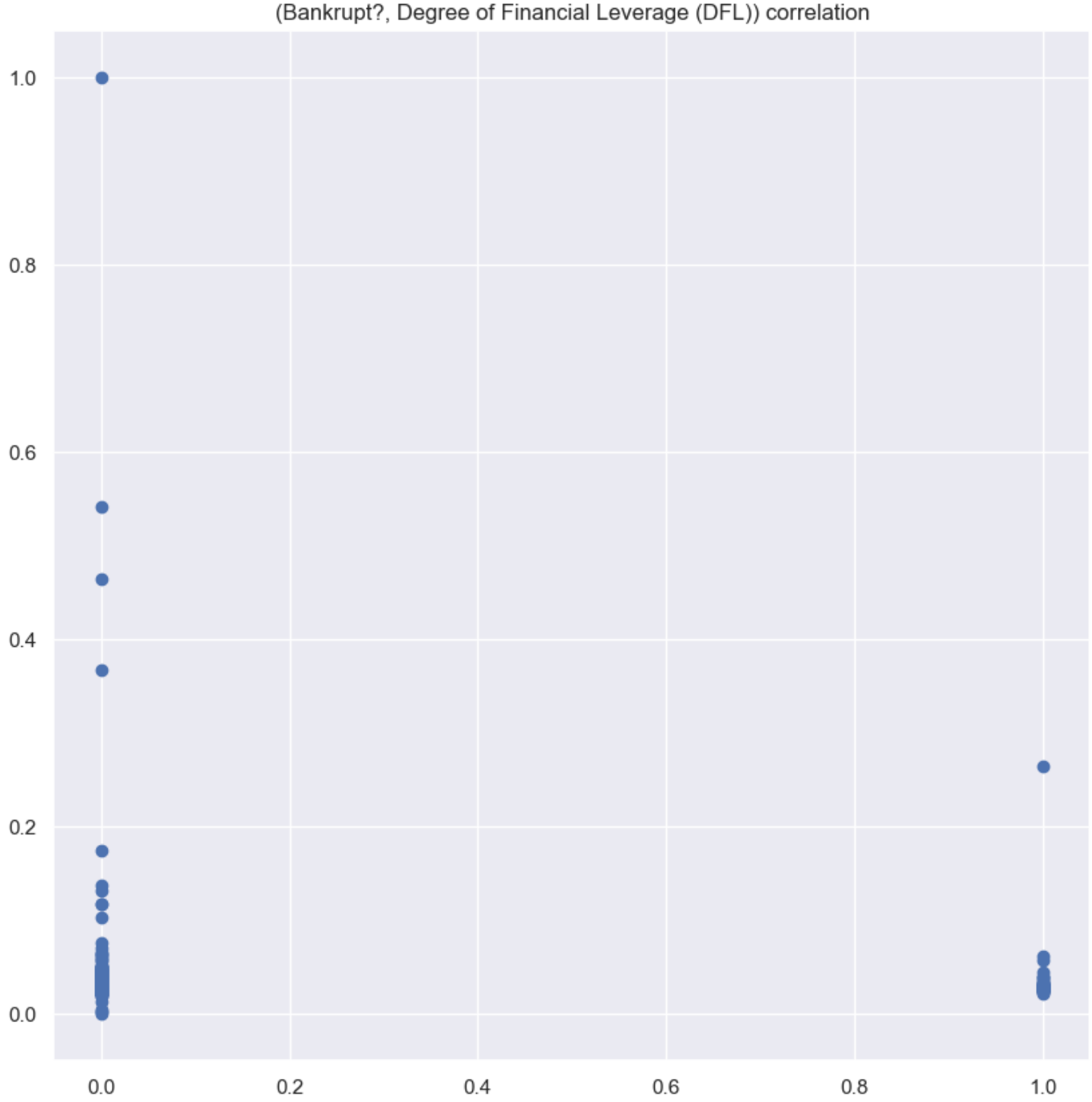

(Bankrupt?, Cash Flow to Equity) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

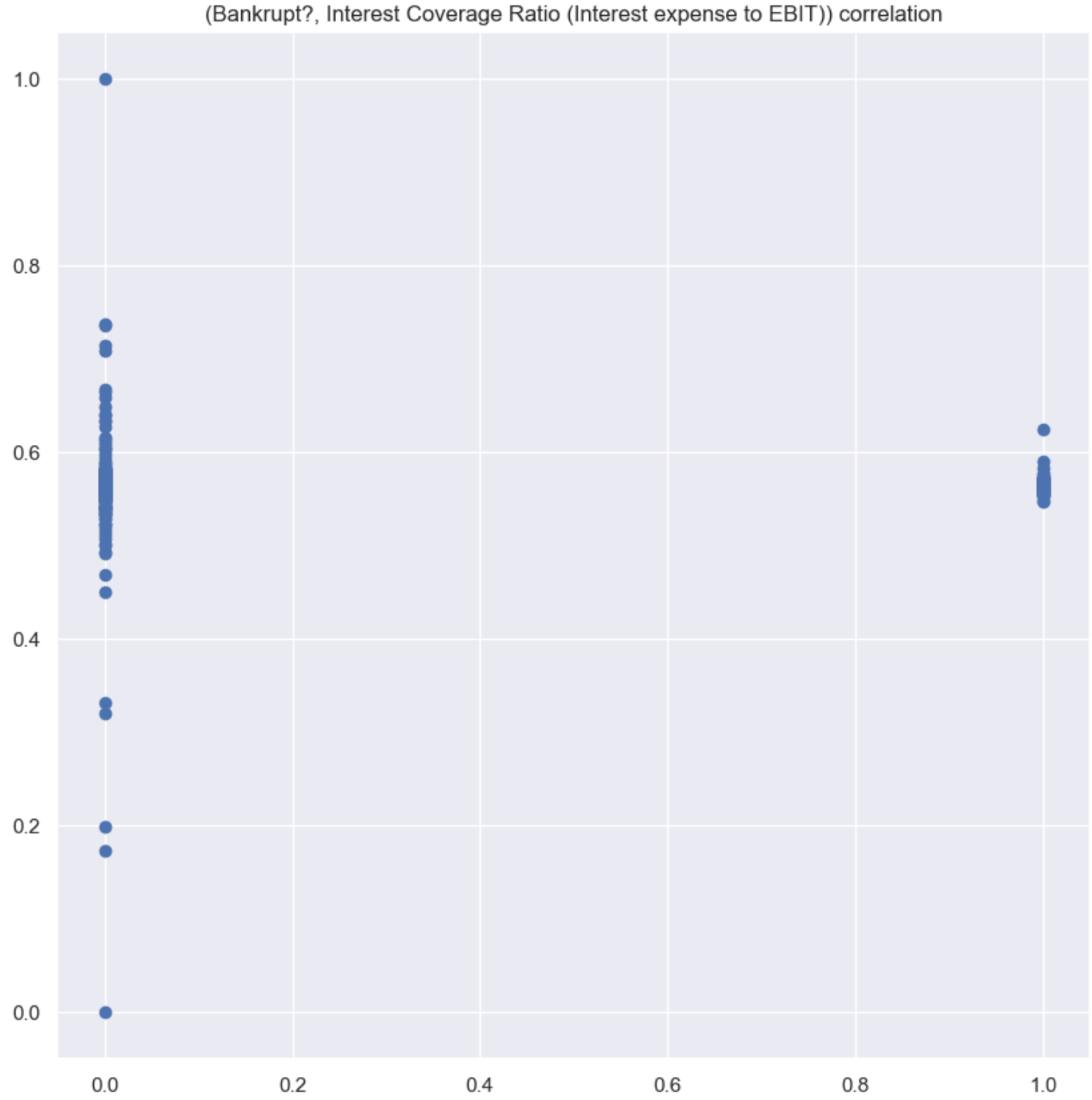

(Bankrupt?, Current Liability to Current Assets) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

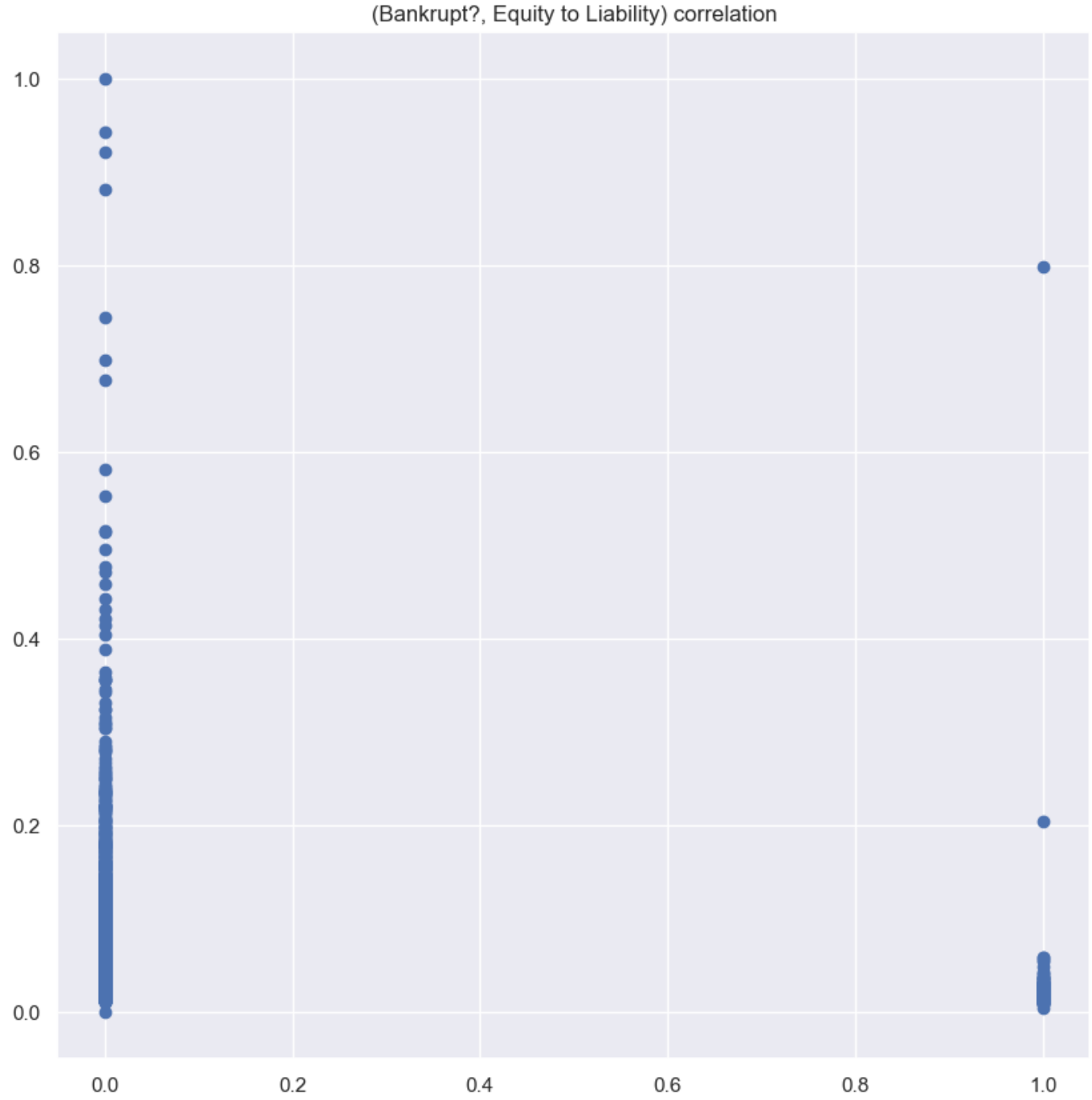

(Bankrupt?, Liability-Assets Flag) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


(Bankrupt?, Total assets to GNP price) correlation
1e10
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


(Bankrupt?, No-credit Interval) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


(Bankrupt?, Degree of Financial Leverage (DFL)) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


(Bankrupt?, Interest Coverage Ratio (Interest expense to EBIT)) correlation
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



{"('Bankrupt?', 'Bankrupt?')": None, "('Bankrupt?', ' ROA(C) before interest and depreciation before interest')": None, "('Bankrupt?', ' Operating Gross Margin')": None, "('Bankrupt?', ' Operating Profit Rate')": None, "('Bankrupt?', ' Non-industry income and expenditure/revenue')": None, "('Bankrupt?', ' Operating Expense Rate')": None, "('Bankrupt?', ' Research and development expense rate')": None, "('Bankrupt?', ' Cash flow rate')": None, "('Bankrupt?', ' Interest-bearing debt interest rate')": None, "('Bankrupt?', ' Tax rate (A)')": None, "('Bankrupt?', ' Net Value Per Share (B)')": None, "('Bankrupt?', ' Persistent EPS in the Last Four Seasons')": None, "('Bankrupt?', ' Cash Flow Per Share')": None, "('Bankrupt?', ' Revenue Per Share (Yuan ¥)')": None, "('Bankrupt?', ' Realized Sales Gross Profit Growth Rate')": None, "('Bankrupt?', ' Operating Profit Growth Rate')": None, "('Bankrupt?', ' After-tax Net Profit Growth Rate')": None, "('Bankrupt?', ' Continuous Net Profit Growth Rate')": None, "('Bankrupt?', ' Total Asset Growth Rate')": None, "('Bankrupt?', ' Net Value Growth Rate')": None, "('Bankrupt?', ' Total Asset Return Growth Rate Ratio')": None, "('Bankrupt?', ' Cash Reinvestment %')": None, "('Bankrupt?', ' Current Ratio')": None, "('Bankrupt?', ' Quick Ratio')": None, "('Bankrupt?', ' Interest Expense Ratio')": None, "('Bankrupt?', ' Total debt/Total net worth')": None, "('Bankrupt?', ' Debt ratio %')": None, "('Bankrupt?', ' Long-term fund suitability ratio (A)')": None, "('Bankrupt?', ' Borrowing dependency')": None, "('Bankrupt?', ' Contingent liabilities/Net worth')": None, "('Bankrupt?', ' Inventory and accounts receivable/Net value')": None, "('Bankrupt?', ' Total Asset Turnover')": None, "('Bankrupt?', ' Accounts Receivable Turnover')": None, "('Bankrupt?', ' Average Collection Days')": No

ne, "('Bankrupt?', ' Inventory Turnover Rate (times)')": None, "('Bankrupt?', ' Fixed As
sets Turnover Frequency')": None, "('Bankrupt?', ' Net Worth Turnover Rate (times)')": N
one, "('Bankrupt?', ' Revenue per person')": None, "('Bankrupt?', ' Operating profit per
person')": None, "('Bankrupt?', ' Allocation rate per person')": None, "('Bankrupt?', '
Working Capital to Total Assets')": None, "('Bankrupt?', ' Quick Assets/Total Assets')":
None, "('Bankrupt?', ' Current Assets/Total Assets')": None, "('Bankrupt?', ' Cash/Total
Assets')": None, "('Bankrupt?', ' Quick Assets/Current Liability')": None, "('Bankrup
t?', ' Cash/Current Liability')": None, "('Bankrupt?', ' Inventory/Working Capital')": N
one, "('Bankrupt?', ' Inventory/Current Liability')": None, "('Bankrupt?', ' Current Lia
bilities/Liability')": None, "('Bankrupt?', ' Working Capital/Equity')": None, "('Bankru
pt?', ' Long-term Liability to Current Assets')": None, "('Bankrupt?', ' Retained Earnin
gs to Total Assets')": None, "('Bankrupt?', ' Total income/Total expense')": None, "('Ba
nkrupt?', ' Total expense/Assets')": None, "('Bankrupt?', ' Current Asset Turnover Rat
e')": None, "('Bankrupt?', ' Quick Asset Turnover Rate')": None, "('Bankrupt?', ' Workin
g capitcal Turnover Rate')": None, "('Bankrupt?', ' Cash Turnover Rate')": None, "('Bank
rupt?', ' Fixed Assets to Assets')": None, "('Bankrupt?', ' Cash Flow to Total Asset
s')": None, "('Bankrupt?', ' Cash Flow to Liability')": None, "('Bankrupt?', ' CFO to As
sets')": None, "('Bankrupt?', ' Cash Flow to Equity')": None, "('Bankrupt?', ' Current L
iability to Current Assets')": None, "('Bankrupt?', ' Liability-Assets Flag')": None, "
('Bankrupt?', ' Total assets to GNP price')": None, "('Bankrupt?', ' No-credit Interva
l')": None, "('Bankrupt?', ' Degree of Financial Leverage (DFL)')": None, "('Bankrupt?',
' Interest Coverage Ratio (Interest expense to EBIT)')": None, "('Bankrupt?', ' Equity t
o Liability')": None}

As we can see in most case the indicator are uniformly distributed between the two possible results, but in other cases we have vary specific range of value in which a ratio falls when showing bankrupt.

Classification algorithm

Set up the pipeline

Having identified the feature we will bring to the final model we can now proceed in setting up the pieline preparing the input data of the model. We will compare the results between a minmax scaler and a z-score scaler, also we compare results between different number of feature in the PCA.
This is the right moment to identify the model to implement, we can follow several approach:

1. Support Vector Machine: work best with linear problem, it identify an hyperplane dividing the two category of firm. If the accuracy is high enough we can use alternativly the multivariate logistic regression.
2. Multivariate logistic regression: similarly to the SVM method place the firm bankrupt value on a logistic regression and round its value to find if it will dafault or not.
3. Decision tree: in order to define a process based on which we identify if a firm can go bankrupt. We will use this model when the problem result to be non-linear.

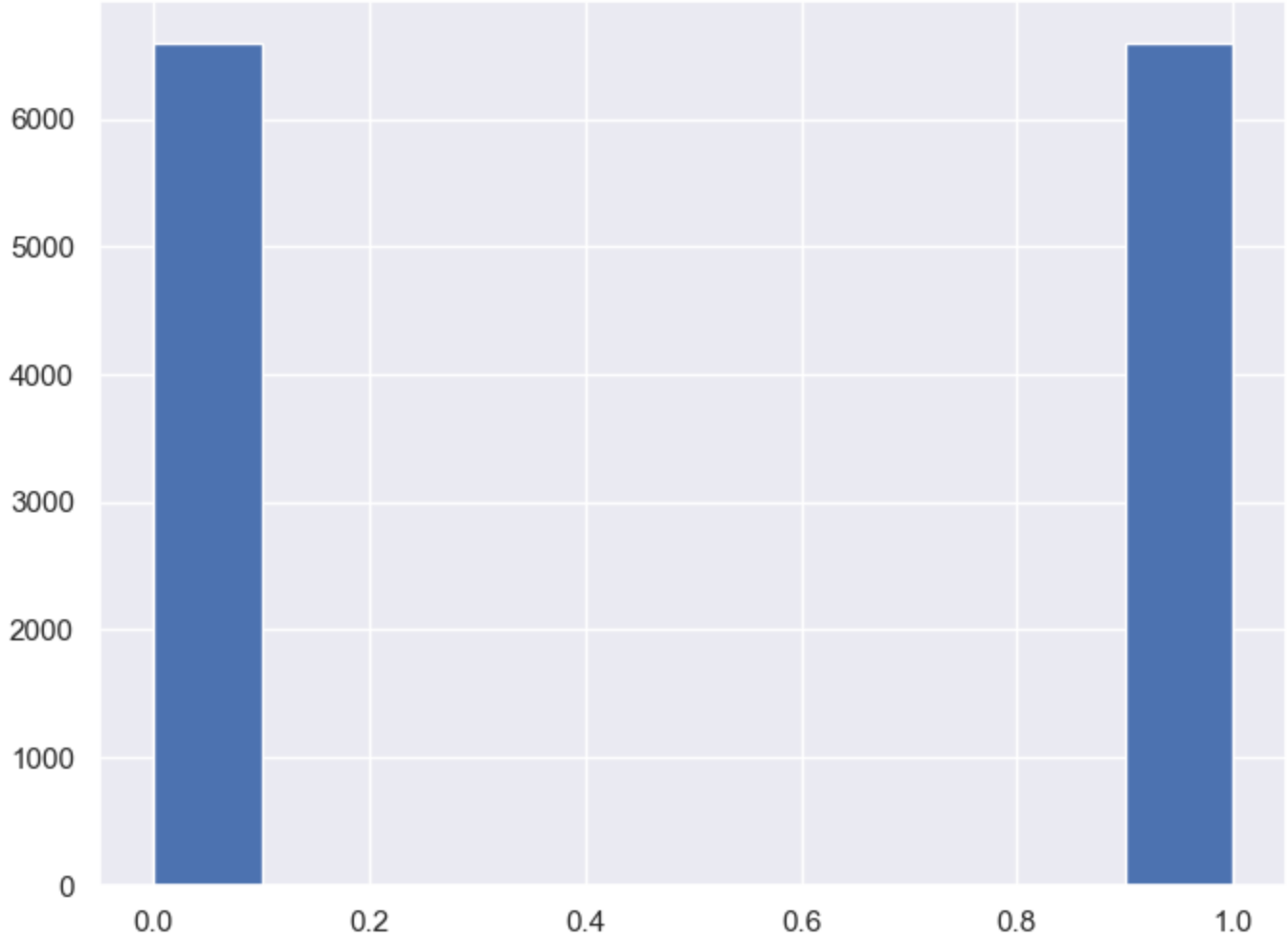
In [9]:
```
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler,StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression,LinearRegression
from sklearn.tree import DecisionTreeClassifier
from tempfile import mkdtemp

def steps(f,scl,model,d=5):
  s1 = ("Feature_reduction",PCA(f))
  if scl =="norm": s2 = ("Scale",StandardScaler())
  elif scl=="minmax": s2 = ("Scale",MinMaxScaler())
  else:
```

```
    print("Insert valid scaler")
    return 0
  if model =="svm": s3 = ("model",SVC())
  elif model =="logist": s3 = ("model",LogisticRegression())
  elif model =="lin": s3 = ("model",LinearRegression())
  elif model =="tree": s3 = ("model",DecisionTreeClassifier(max_depth=d))
  else:
    print("Insert valid scaler")
    return 0
  return [s1,s2,s3]
```

In order to handle the oversampling problem we will use SMOTE - Synthetic Minority Over-sampling TEchnique. Is a method used when a label we are trying to predict is rare. It can be summarised by the following steps:

1. Calculate the difference between a sample and its nearest neighbor.
2. Multiply this difference by a random number between 0 and 1.
3. Add the resulting value to the sample to create a new synthetic example.
4. Repeat this process with the next nearest neighbo
   (To go deeper on SMOTE method here is a nice article from Medium)

In [10]:
```
from imblearn.over_sampling import SMOTE
y,X = n_df[n_df.columns[0]].to_numpy(),n_df.drop(n_df.columns[0],axis=1).to_numpy()
X, y = SMOTE().fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=42,train_size=0.5)

plt.figure(figsize=(8,6))
plt.hist(y)
plt.show()
```

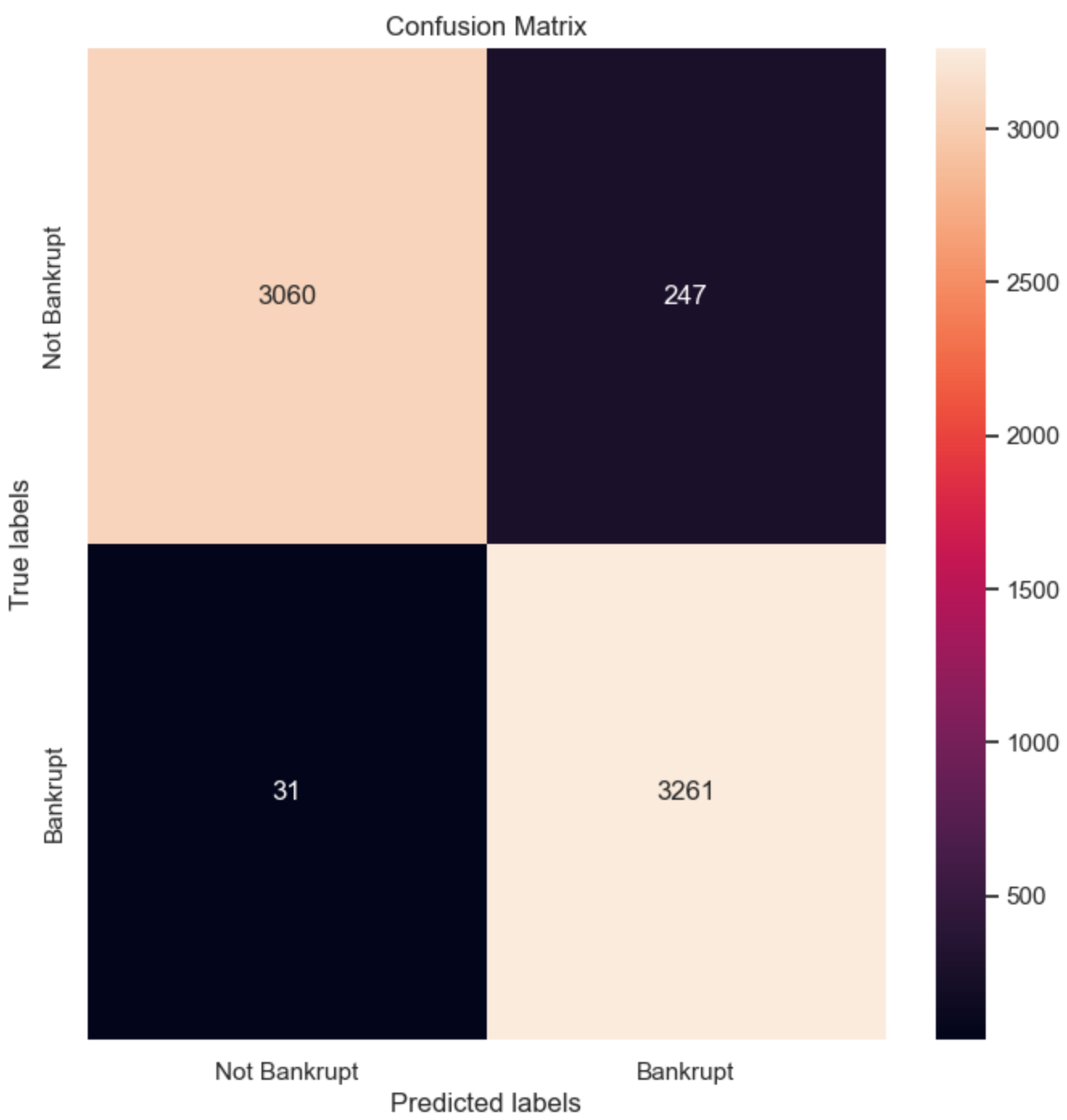


In [11]:
```
cachedir = mkdtemp() #used to avoid computing the fit transformers within a pipeline if
pipe = Pipeline(steps(40,"norm","svm"),memory=cachedir)
pipe
```

Out[11]:
```
Pipeline(memory='/var/folders/sr/r5xnmb6d4_dc9mgnjc301qr40000gn/T/tmpi_hgdohx',
         steps=[('Feature_reduction', PCA(n_components=40)),
                ('Scale', StandardScaler()), ('model', SVC())])
```

In [12]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
import matplotlib.pyplot as plt

model = pipe.fit(X_train,y_train)
y_pred = model.predict(X_test)
print("Model accuracy: ", accuracy_score(y_test,y_pred))

plt.figure(figsize=(8,8))
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True, fmt='g', ax=ax)
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['Not Bankrupt', 'Bankrupt'])
ax.yaxis.set_ticklabels(['Not Bankrupt', 'Bankrupt'])
```

Out[12]:
```
Model accuracy:  0.9578724049098348
[Text(0, 0.5, 'Not Bankrupt'), Text(0, 1.5, 'Bankrupt')]
```



In [13]:
```
cachedir = mkdtemp() #used to avoid computing the fit transformers within a pipeline if
log_reg = Pipeline(steps(40,"norm","logist"),memory=cachedir)
log_reg
```

Out[13]:
```
Pipeline(memory='/var/folders/sr/r5xnmb6d4_dc9mgnjc301qr40000gn/T/tmpzlwx4oyr',
         steps=[('Feature_reduction', PCA(n_components=40)),
                ('Scale', StandardScaler()), ('model', LogisticRegression())])
```

In [14]:
```
from sklearn.metrics import mean_absolute_error, max_error, mean_squared_error,mean_abso


model = log_reg.fit(X_train,y_train)
y_pred = model.predict(X_test)
print("Model mean absolute error: ", mean_absolute_error(y_test,y_pred))
print("Model accuracy: ", accuracy_score(y_test,y_pred))
```

```
Model mean absolute error:  0.10289437793605091
Model accuracy:  0.8971056220639491
```

The accuracy is great for both method (best for SVM), this allows to think that the problem is linear and

we do not need another model. But since in most case is useful to understand how a ML model make a decision, consider the decision tree classifier as a possible link between human logic and ML algorithm.

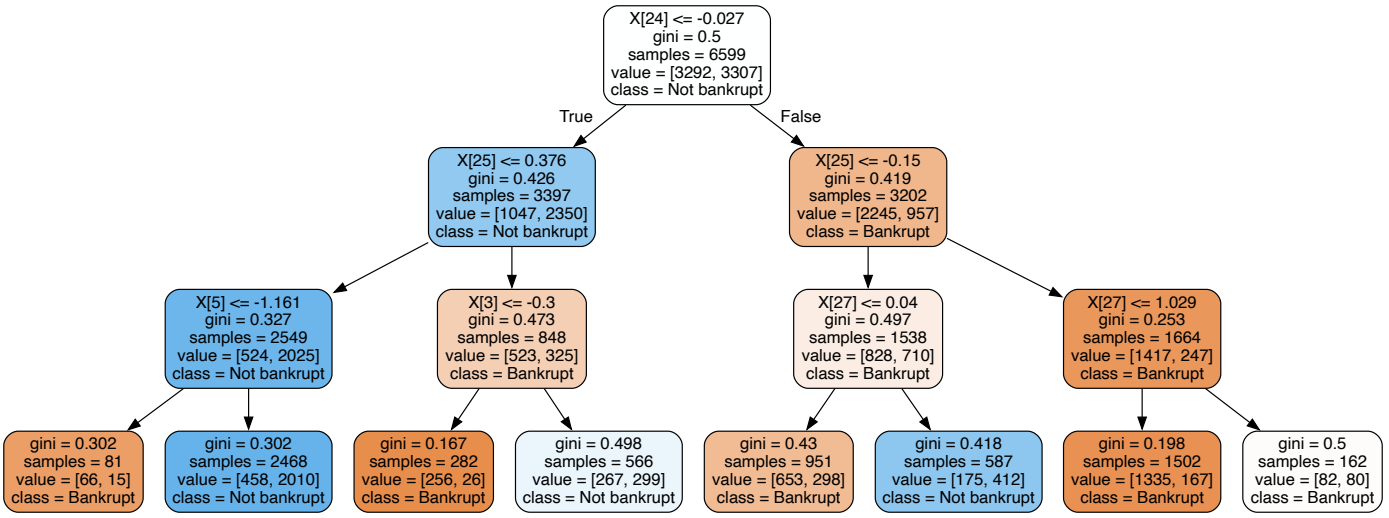
In [15]:
```
cachedir = mkdtemp() #used to avoid computing the fit transformers within a pipeline if
clf = Pipeline(steps(40,"norm","tree",3),memory=cachedir)
clf
```

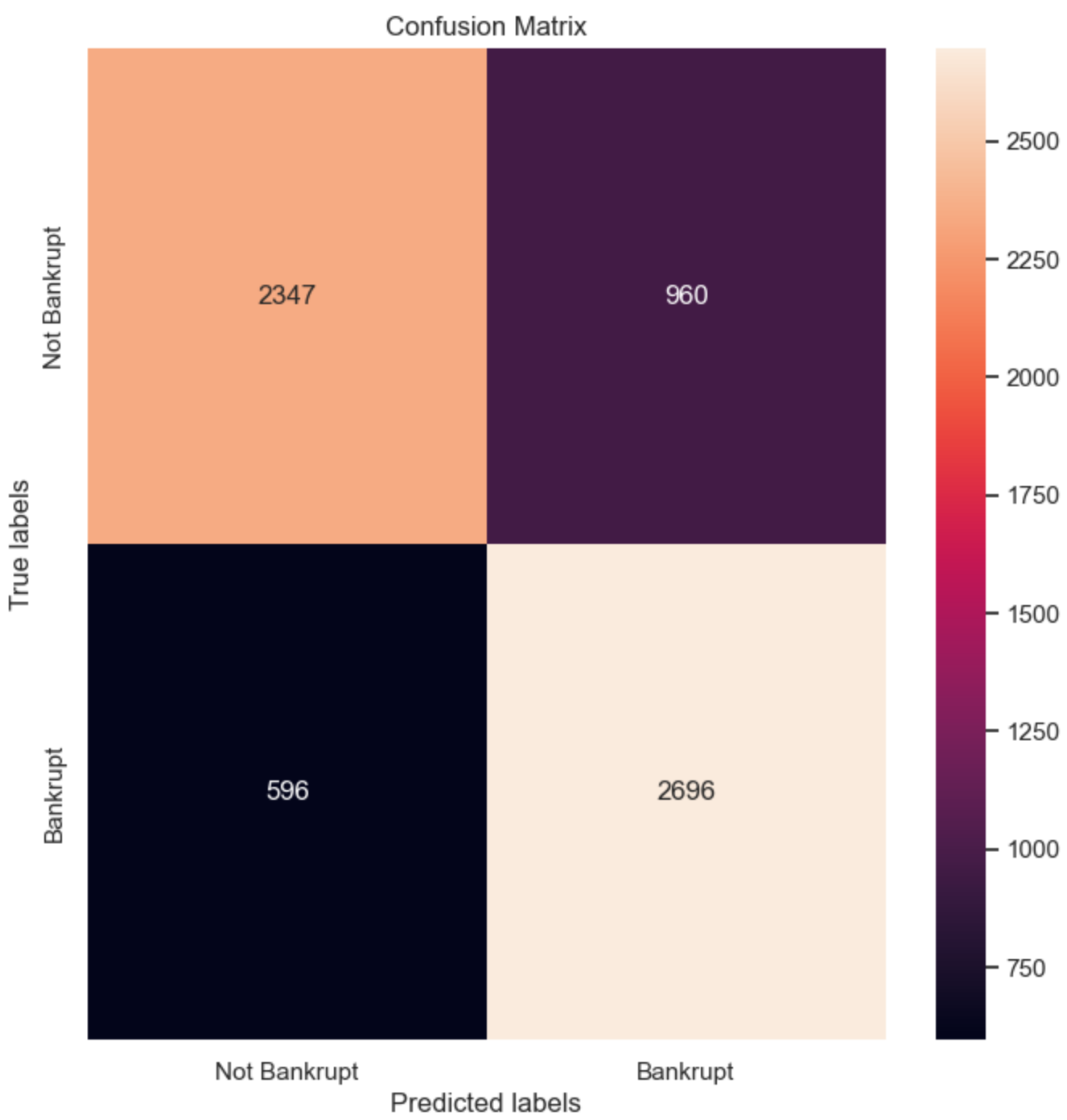
Out[15]:
```
Pipeline(memory='/var/folders/sr/r5xnmb6d4_dc9mgnjc301qr40000gn/T/tmpk7e_0xkx',
         steps=[('Feature_reduction', PCA(n_components=40)),
                ('Scale', StandardScaler()),
                ('model', DecisionTreeClassifier(max_depth=3))])
```

In [16]:
```
from sklearn import tree
import graphviz


model = clf.fit(X_train,y_train)
y_pred = model.predict(X_test)
#print("Model mean absolute error: ", mean_absolute_error(y_test,y_pred))
print("Model accuracy: ", accuracy_score(y_test,y_pred))


plt.figure(figsize=(8,8))
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True, fmt='g', ax=ax)
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['Not Bankrupt', 'Bankrupt'])
ax.yaxis.set_ticklabels(['Not Bankrupt', 'Bankrupt'])

dot_data = tree.export_graphviz(model[-1],class_names=["Bankrupt",'Not bankrupt'],filled

graph = graphviz.Source(dot_data)
graph
```

```
Model accuracy:  0.7642066979845431
```

Out[16]:





Hyperparameter tuning

The best model results to be the SVM, now we try to optimise further the accuracy by modifying the original pipeline and train size.

In [17]:
```
cachedir = mkdtemp() #used to avoid computing the fit transformers within a pipeline if
pipe = Pipeline(steps(40,"norm","svm"),memory=cachedir)
X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=42,train_size=0.5)

data_size = [0.15,0.20,0.33,0.4,0.5,0.55]
pca_size = [5,10,20,30,35,40,45,50,60,65]
scal = ["minmax","norm"]

acc_mx_norm = np.ndarray(shape=(len(data_size),len(pca_size)))
acc_mx_mnmx= np.ndarray(shape=(len(data_size),len(pca_size)))

for d in range(len(data_size)):
  for p in range(len(pca_size)):
```

```
    for s in scal:
      pipe = Pipeline(steps(pca_size[p],s,"svm"),memory=cachedir)
      X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=42,train_siz
      model = pipe.fit(X_train,y_train)
      acc = accuracy_score(y_test,model.predict(X_test))
      #nrm = {"train_size":d ,"feature": p,"scaler":s,"accuracy": acc}
      #print(f"Accuracy for {d} training size, {p} total feature reduced and {s} scaler:
      if s == "norm":
        acc_mx_norm[d,p] = acc
      else:
        acc_mx_mnmx[d,p] = acc

norm_acc = pd.DataFrame(data = acc_mx_norm, columns = pca_size,index =data_size)
minmax_acc = pd.DataFrame(data = acc_mx_mnmx, columns = pca_size,index =data_size)
```

We have created a dataframe representing the accuracy for different sample size of the training set and number of feature used by the model. We can now identify the best combination for our model.

In [18]:
```
print("Standard scaler")
display(norm_acc.style.applymap(lambda x: 'background-color : yellow' if x>0.95 else '')
print("MinMax scaler")
display(minmax_acc.style.applymap(lambda x: 'background-color : yellow' if x>0.92 else '
```

Standard scaler

	5	10	20	30	35	40	45	50	60
0.150000	0.676442	0.780194	0.765933	0.915946	0.932882	0.935556	0.935288	0.936804	0.936269
0.200000	0.681030	0.793446	0.765982	0.918837	0.934463	0.939767	0.938725	0.939388	0.938915
0.330000	0.679294	0.824720	0.784688	0.930793	0.943006	0.946172	0.946511	0.949565	0.948208
0.400000	0.691249	0.831671	0.790251	0.939260	0.951130	0.953908	0.956181	0.957949	0.955297
0.500000	0.703743	0.841037	0.799060	0.944082	0.955145	0.957872	0.959691	0.963176	0.959691
0.550000	0.701684	0.842761	0.803199	0.939899	0.956734	0.959764	0.959428	0.964815	0.962626

MinMax scaler

	5	10	20	30	35	40	45	50	60
0.150000	0.676174	0.771994	0.749710	0.903022	0.910331	0.911311	0.912203	0.910955	0.914431
0.200000	0.678852	0.768160	0.751681	0.901790	0.913628	0.914859	0.913818	0.915238	0.916659
0.330000	0.671378	0.790343	0.765577	0.913039	0.920841	0.919937	0.918127	0.917449	0.918693
0.400000	0.683293	0.787852	0.772446	0.919939	0.926380	0.926632	0.926380	0.926758	0.928147
0.500000	0.686468	0.795120	0.775269	0.919685	0.928929	0.927413	0.930596	0.929989	0.930444
0.550000	0.701684	0.797811	0.775758	0.913131	0.925758	0.926094	0.927104	0.928283	0.928283

We can highlight how the standard scaler result in better accuracy, while for the number of feature after 30 we reach a platou where the increase of feature increase the accuracy slightly.
Since a large training set could lead to overfitting the model we coould chose as hyperparameter a standard scaler with 40% trainig set and 45 feature in total.

Conclusion

After proper trasnformation of the dataset, we have built three possible ML model to predict the default of a firm. First we have explored the dataset analysing the kind of datatype and the correlation among the features. To address the class imbalance we introduced the SMOTE method. We have done some

preliminary feature selection through the mean of correlation and then a proper PCA built inside the pipeline itself. Finally, we have built three different ML model focusing at the end on the SVM method which shows bets accuracy performance.

Further work

The dataset imbalance is a great limit for the model, so talking abount future steps we surely need to enlarge the dataset in that direction. We could also think about a more deep feature selection considering the tradeoff between overfitting and accuracy.