```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from google.colab import files
```

Uploading and setting up the Polish Dataset

uploading and renaming the columns for further review of the polish dataset.

```
#pulling data from the website.
polishData=pd.read_csv("https://archive.ics.uci.edu/static/public/365/data.csv")
#adding the data into a dataframe
polish_Dataset = pd.DataFrame(polishData)

# Set display options to show all rows and columns
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)

# Set display width to avoid wrapping of wide DataFrames
pd.set_option('display.width', None)

# Set option to display full content of each column
pd.set_option('display.max_colwidth', None)
polish_Dataset.head(10)
```

	year	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A1
0	1	0.200550	0.379510	0.396410	2.0472	32.3510	0.38825	0.249760	1.33050	1.13890	0.50494	0.249760	0.659800	0.166600	0.24976
1	1	0.209120	0.499880	0.472250	1.9447	14.7860	0.00000	0.258340	0.99601	1.69960	0.49788	0.261140	0.516800	0.158350	0.25834
2	1	0.248660	0.695920	0.267130	1.5548	-1.1523	0.00000	0.309060	0.43695	1.30900	0.30408	0.312580	0.641840	0.244350	0.30906
3	1	0.081483	0.307340	0.458790	2.4928	51.9520	0.14988	0.092704	1.86610	1.05710	0.57353	0.092704	0.301630	0.094257	0.09270
4	1	0.187320	0.613230	0.229600	1.4063	-7.3128	0.18732	0.187320	0.63070	1.15590	0.38677	0.187320	0.331470	0.121820	0.18732
5	1	0.228220	0.497940	0.359690	1.7502	-47.7170	0.00000	0.281390	1.00830	1.97860	0.50206	0.286450	0.586910	0.148120	0.28139
6	1	0.111090	0.647440	0.289710	1.4705	2.5349	0.00000	0.111090	0.54454	1.73480	0.35256	0.125750	0.180410	0.309630	0.11109
7	1	0.532320	0.027059	0.705540	53.9540	299.5800	0.00000	0.652400	35.95700	0.65273	0.97294	0.693940	48.966000	1.060200	0.65240
8	1	0.009020	0.632020	0.053735	1.1263	-37.8420	0.00000	0.014434	0.58223	1.33320	0.36798	0.043162	0.033921	0.038938	0.01443
9	1	0.124080	0.838370	0.142040	1.1694	-91.8830	0.00000	0.153280	0.19279	2.11560	0.16163	0.184540	0.182840	0.075411	0.15328

```
#The orginal dataset did not have named columns. It provided an explantion to each attribute in their reserach paper. The code below is to ad
#replacing the column name with actual names.

new_Column_names={
    'A1':'net_profit/total assets',
    "A2":'Total_Liabilities/total_assets',
    "A3":'Working_Capital_to_Total_Assets_Ratio',
    "A4":'Current_Ratio',
    "A5":"Defensive_Interval_Ratio",
    "A6":'Retained_Earnings_to_Total_Assets_Ratio',
    "A7":"Operating_Income_to_Total_Asset_ratio",
    "A8":"book_value_of_equity/total_liabilities",
    "A9":'Asset_Turnover_Ratio',
    "A10":'Equity_to_Total_Assets_Ratio',
    "A11":'gross_profit+extraordinary_items+financial_expenses)/total',
    "A12":'gross_profit/short-term_liabilities',
    "A13":'(gross_profit+depreciation)/sales',
    "A14":'(gross_profit+interest)/total_assets',
    "A15":'(total_liabilities*365)/(gross_profit+depreciation)',
    "A16":'(gross profit+depreciation)/total_liabilities',
    "A17":'total_assets/total_liabilities',
    "A18":'gross_profit/total_assets',
    "A19":'gross_profit/sales',
    "A20":'(inventory*365)/sales',
    "A21":'sales(n)/sales(n-1)',
    "A22":'profit_on_operating_activities/total_assets',
```

```
    "A23":'net_profit/sales',
    "A24":'gross_profit(in3years)/total_assets',
    "A25":'(equity-share_capital)/total_assets',
    "A26":'(net_profit+depreciation)/total_liabilities',
    "A27":'profit_on_operating_activities/financial_expenses',
    "A28":'working_capital/fixed_assets',
    "A29":'logarithm_of_total_assets',
    "A30":'(total_liabilities-cash)/sales',
    "A31":'(gross_profit+interest)/sales',
    "A32":'(current_liabilities*365)/cost_of_products_sold',
    "A33":'operating_expenses/short-term_liabilities',
    "A34":'operating_expenses_/_total_liabilities',
    "A35":'profit_on_sales/total_assets',
    "A36":'total_sales/total_assets',
    "A37":'(current_assets-inventories)/long-term_liabilities',
    "A38":'constant_capital/total_assets',
    "A39":'profit_on_sales/sales',
    "A40":'(current_assets-inventory-receivables)/short-term liabilities',
    "A41":'total_liabilities/((profit_on_operating_activities+depreciation)*(12/365))',
    "A42":'profit_on_operating_activities/sales',
    "A43":'rotation_receivables+inventory_turnover_in_days',
    "A44":'(receivables*365)/sales',
    "A45": 'net_profit/inventory',
    "A46":'(current_assets-inventory)/short-term_liabilities',
    "A47":'(inventory*365)/cost_of_products_sold',
    "A48":'EBITDA_(profit_on_operating_activities-depreciation)/total_assets',
    "A49":'EBITDA_(profit_on_operating_activities-depreciation)/sales',
    "A50":'current_assets/total_liabilities',
    "A51":'short-term_liabilities/total_assets',
    "A52":'(short-term_liabilities*365)/cost_of_products_sold)',
    "A53":'equity/fixed_assets',
    "A54":'constant_capital/fixed_assets',
    "A55":'working_capital',
    "A56":'(sales-cost_of_products_sold)/sales',
    "A57":'(current_assets-inventory-short-term_liabilities)/(sales-gross_profit-depreciation)',
    "A58":'total_costs/total_sales',
    "A59":'long-term_liabilities/equity',
    "A60":'sales/inventory',
    "A61":'sales/receivables',
    "A62":'(short-term_liabilities*365)/sales',
    "A63":'sales/short-term liabilities',
    "A64":'sales/fixed assets',
                        }
```

```
polish_Dataset =polish_Dataset.rename(columns=new_Column_names)
```

```
#based on the intial review of all the attributes type, it was noted that the class ratio was listed as int.
#making the class into a categorial data type from int
polish_Dataset['class']=polish_Dataset['class'].astype('category')
```

```
#snapshot of the features after change in data type 'class'
polish_Dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43405 entries, 0 to 43404
Data columns (total 66 columns):
 #   Column                                                                  Non-Null Count  Dtype
---  ------                                                                  --------------  -----
 0   year                                                                    43405 non-null  int64
 1   net_profit/total assets                                                 43397 non-null  float64
 2   Total_Liabilities/total_assets                                          43397 non-null  float64
 3   Working_Capital_to_Total_Assets_Ratio                                   43397 non-null  float64
 4   Current_Ratio                                                           43271 non-null  float64
 5   Defensive_Interval_Ratio                                                43316 non-null  float64
 6   Retained_Earnings_to_Total_Assets_Ratio                                 43397 non-null  float64
 7   Operating_Income_to_Total_Asset_ratio                                   43397 non-null  float64
 8   book_value_of_equity/total_liabilities                                  43311 non-null  float64
 9   Asset_Turnover_Ratio                                                    43396 non-null  float64
 10  Equity_to_Total_Assets_Ratio                                            43397 non-null  float64
 11  gross_profit+extraordinary_items+financial_expenses)/total              43361 non-null  float64
 12  gross_profit/short-term_liabilities                                     43271 non-null  float64
```

```
13  (gross_profit+depreciation)/sales                                    43278 non-null  float64
14  (gross_profit+interest)/total_assets                                 43397 non-null  float64
15  (total_liabilities*365)/(gross_profit+depreciation)                  43369 non-null  float64
16  (gross profit+depreciation)/total_liabilities                        43310 non-null  float64
17  total_assets/total_liabilities                                       43311 non-null  float64
18  gross_profit/total_assets                                            43397 non-null  float64
19  gross_profit/sales                                                   43277 non-null  float64
20  (inventory*365)/sales                                                43278 non-null  float64
21  sales(n)/sales(n-1)                                                  37551 non-null  float64
22  profit_on_operating_activities/total_assets                          43397 non-null  float64
23  net_profit/sales                                                     43278 non-null  float64
24  gross_profit(in3years)/total_assets                                  42483 non-null  float64
25  (equity-share_capital)/total_assets                                  43397 non-null  float64
26  (net_profit+depreciation)/total_liabilities                          43310 non-null  float64
27  profit_on_operating_activities/financial_expenses                    40641 non-null  float64
28  working_capital/fixed_assets                                         42593 non-null  float64
29  logarithm_of_total_assets                                            43397 non-null  float64
30  (total_liabilities-cash)/sales                                       43278 non-null  float64
31  (gross_profit+interest)/sales                                        43278 non-null  float64
32  (current_liabilities*365)/cost_of_products_sold                      43037 non-null  float64
33  operating_expenses/short-term_liabilities                            43271 non-null  float64
34  operating_expenses_/_total_liabilities                               43311 non-null  float64
35  profit_on_sales/total_assets                                         43397 non-null  float64
36  total_sales/total_assets                                             43397 non-null  float64
37  (current_assets-inventories)/long-term_liabilities                   24421 non-null  float64
38  constant_capital/total_assets                                        43397 non-null  float64
39  profit_on_sales/sales                                                43278 non-null  float64
40  (current_assets-inventory-receivables)/short-term liabilities        43271 non-null  float64
41  total_liabilities/((profit_on_operating_activities+depreciation)*(12/365))  42651 non-null  float64
42  profit_on_operating_activities/sales                                 43278 non-null  float64
43  rotation_receivables+inventory_turnover_in_days                      43278 non-null  float64
44  (receivables*365)/sales                                              43278 non-null  float64
45  net_profit/inventory                                                 41258 non-null  float64
46  (current_assets-inventory)/short-term_liabilities                    43270 non-null  float64
47  (inventory*365)/cost_of_products_sold                                43108 non-null  float64
48  EBITDA_(profit_on_operating_activities-depreciation)/total_assets    43396 non-null  float64
49  EBITDA_(profit_on_operating_activities-depreciation)/sales           43278 non-null  float64
50  current_assets/total_liabilities                                     43311 non-null  float64
51  short-term_liabilities/total_assets                                  43397 non-null  float64
```

Variable Additions

This research paper was to introduce financial ratios that can help identify financial stress. This analysis was done by reviewing all attributes and seeing what can be added.

The following additions are based on using exisitng variables to create these additions. This is based on existing and industry knowledge.

```
#The introduction of this variable is to get an understanding of how much of interest paid is to sales.
#A high proportion of interest to sales will indicate that a good portion of their sales earned will go towards paying the interest and othe
#Variable 1
Interest_to_Sales = (polish_Dataset['(gross_profit+interest)/sales']-polish_Dataset['gross_profit/sales'])
Interest_to_Sales.replace([np.inf, -np.inf], np.NaN, inplace=True )
Interest_to_Sales
polish_Dataset['Interest_to_Sales'] = Interest_to_Sales



#To gain an understanding on liquidity compared to all assets. Cash is the most liquid asset and can be
#used to cover debt/interest in the short term.
#variable 2
Cash_to_assets  = (polish_Dataset['Total_Liabilities/total_assets']-(polish_Dataset['(total_liabilities-cash)/sales'] * polish_Dataset['tota
Cash_to_assets.replace([np.inf, -np.inf], np.NaN, inplace=True )
Cash_to_assets
polish_Dataset['Cash_to_assets'] = Cash_to_assets



#This is to gain an understanding on how much cash can cover total liabilities. Too high can tell us that the company may have a hard time c
#variable 3
Total_liabilities_to_Cash = (polish_Dataset['Total_Liabilities/total_assets']/ polish_Dataset['Cash_to_assets'] )
Total_liabilities_to_Cash.replace([np.inf, -np.inf], np.NaN, inplace=True )
Total_liabilities_to_Cash
polish_Dataset['Total_liabilities_to_Cash'] = Total_liabilities_to_Cash
```

```
#Tells us how much interest can be covered by cash. A high number gives comfort that the company can cover their interest obligations
#variable 4
cash_to_interest = (polish_Dataset['Cash_to_assets']/ polish_Dataset['total_sales/total_assets'] * (1/polish_Dataset['Interest_to_Sales']))
cash_to_interest.replace([np.inf, -np.inf], np.NaN, inplace=True )
cash_to_interest
polish_Dataset['cash_to_interest'] = cash_to_interest


#After initial analysis, I realized that not all companies have debt/paid interest during their fiscal year.
#This is ratio is taking the inverse.
#variable 5
Interest_to_cash = 1 /(polish_Dataset['cash_to_interest'])
Interest_to_cash.replace([np.inf, -np.inf], np.NaN, inplace=True )
Interest_to_cash
polish_Dataset['Interest_to_cash'] = Interest_to_cash


#This ratio to review how much financial expense is covered by total assets. a high number indicates to me that the company has taken on toc
#may not be enough relative to debt. this ratio can be used with asset turnover to provide more direction.
#variable 6
Financial_Expense_To_Total_Asset = (polish_Dataset['profit_on_operating_activities/total_assets'] / polish_Dataset['profit_on_operating_acti
Financial_Expense_To_Total_Asset.replace([np.inf, -np.inf], np.NaN, inplace=True )
Financial_Expense_To_Total_Asset
polish_Dataset['Financial_Expense_To_Total_Asset'] = Financial_Expense_To_Total_Asset


#variable 6
Financial_expense_to_operating = (polish_Dataset['profit_on_operating_activities/total_assets'] / (polish_Dataset["Operating_Income_to_Total
Financial_expense_to_operating.replace([np.inf, -np.inf], np.NaN, inplace=True )
Financial_expense_to_operating
polish_Dataset['Financial_expense_to_operating']=Financial_expense_to_operating


#this ratio gives me an idea on how much operating income can cover the interst for the fiscal period. a high ratio is a good indicator.
#variable 7
Debt_Coverage_Ratio = ((polish_Dataset["Operating_Income_to_Total_Asset_ratio"]*polish_Dataset['profit_on_operating_activities/financial_exp
Debt_Coverage_Ratio.replace([np.inf, -np.inf], np.NaN, inplace=True )
Debt_Coverage_Ratio
polish_Dataset['Debt_Coverage_Ratio'] = Debt_Coverage_Ratio


#operating income can be viewed as cash flow from operating. it it not however, but in its place it can give the reader an idea.
#THis give us the percentage that is avaiable to firm after all expenses are paid from sales for the fiscal
#variable 8
Operating_income_Margin = (polish_Dataset['Operating_Income_to_Total_Asset_ratio'] / polish_Dataset['total_sales/total_assets'])
Operating_income_Margin.replace([np.inf, -np.inf], np.NaN, inplace=True )
Operating_income_Margin
polish_Dataset['Operating_income_Margin'] = Operating_income_Margin


#this ratio tells us how many days  cash is recieved from the momement it is used to pay for inventory and accounts payble, as well as cash
#Variable 9 Cash Conversion Cycle Formula = DIO + DSO – DPO.
Cash_conversion_cycle = ((polish_Dataset['(inventory*365)/cost_of_products_sold']) + (polish_Dataset['(receivables*365)/sales']) - (polish_D
Cash_conversion_cycle.replace([np.inf, -np.inf], np.NaN, inplace=True )
Cash_conversion_cycle
polish_Dataset['Cash_conversion_cycle'] = Cash_conversion_cycle


#This ratio is used to determine how much cash can be used to cover financial expense. since cash is the most liquid asset, a high number in
#the ability to cover financial expenses
#variable 10
Cash_to_Financial_expense = (polish_Dataset['Cash_to_assets'] / polish_Dataset['Financial_Expense_To_Total_Asset'])
Cash_to_Financial_expense.replace([np.inf, -np.inf], np.NaN, inplace=True )
polish_Dataset['Cash_to_Financial_Expense']= Cash_to_Financial_expense


#Used as part of the dupont formula (i.e Return on equity), gives us indication how much leverage is used.

#variable 11
Equity_multipler= 1/ polish_Dataset['Equity_to_Total_Assets_Ratio']
Equity_multipler.replace([np.inf, -np.inf], np.NaN, inplace=True )
polish_Dataset['Financial_Leverage'] = Equity_multipler


#not addition but transforming an existing attribute to 1,0. if working capital is postive, it will be given a 1, if negative it will given
#making working capital into a indicator. 1 = postive values, 0 = negative
polish_Dataset['working_capital_indicator'] =np.where(polish_Dataset['working_capital']<=0,0,1)
polish_Dataset['working_capital_indicator']=polish_Dataset['working_capital_indicator'].astype('category')
```

```
#final ratio is return on equity. this is to gauge on management perfromance on how much they retun to shareholders.
#since this ratio is the result of the dupoint formula, it helps us determine what goes into the results for shareholders. such as
#asset turnover and financial leverage. a high financial leverage tell us that the results are coming from debt.
#Variable12
Return_On_equity =polish_Dataset['net_profit/sales'] * polish_Dataset['total_sales/total_assets'] *polish_Dataset['Financial_Leverage']
Return_On_equity.replace([np.inf, -np.inf], np.NaN, inplace=True )
polish_Dataset['Return_On_equity'] = Return_On_equity
```

Start coding or generate with AI.

```
#Review of all financial ratios in the polish dataset
polish_Dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43405 entries, 0 to 43404
Data columns (total 80 columns):
 #   Column                                                                Non-Null Count  Dtype
---  ------                                                                --------------  -----
 0   year                                                                  43405 non-null  int64
 1   net_profit/total assets                                               43397 non-null  float64
 2   Total_Liabilities/total_assets                                        43397 non-null  float64
 3   Working_Capital_to_Total_Assets_Ratio                                 43397 non-null  float64
 4   Current_Ratio                                                         43271 non-null  float64
 5   Defensive_Interval_Ratio                                              43316 non-null  float64
 6   Retained_Earnings_to_Total_Assets_Ratio                               43397 non-null  float64
 7   Operating_Income_to_Total_Asset_ratio                                 43397 non-null  float64
 8   book_value_of_equity/total_liabilities                                43311 non-null  float64
 9   Asset_Turnover_Ratio                                                  43396 non-null  float64
 10  Equity_to_Total_Assets_Ratio                                          43397 non-null  float64
 11  gross_profit+extraordinary_items+financial_expenses)/total            43361 non-null  float64
 12  gross_profit/short-term_liabilities                                   43271 non-null  float64
 13  (gross_profit+depreciation)/sales                                     43278 non-null  float64
 14  (gross_profit+interest)/total_assets                                  43397 non-null  float64
 15  (total_liabilities*365)/(gross_profit+depreciation)                   43369 non-null  float64
 16  (gross profit+depreciation)/total_liabilities                         43310 non-null  float64
 17  total_assets/total_liabilities                                        43311 non-null  float64
 18  gross_profit/total_assets                                             43397 non-null  float64
 19  gross_profit/sales                                                    43277 non-null  float64
 20  (inventory*365)/sales                                                 43278 non-null  float64
 21  sales(n)/sales(n-1)                                                   37551 non-null  float64
 22  profit_on_operating_activities/total_assets                           43397 non-null  float64
 23  net_profit/sales                                                      43278 non-null  float64
 24  gross_profit(in3years)/total_assets                                   42483 non-null  float64
 25  (equity-share_capital)/total_assets                                   43397 non-null  float64
 26  (net_profit+depreciation)/total_liabilities                           43310 non-null  float64
 27  profit_on_operating_activities/financial_expenses                     40641 non-null  float64
 28  working_capital/fixed_assets                                          42593 non-null  float64
 29  logarithm_of_total_assets                                             43397 non-null  float64
 30  (total_liabilities-cash)/sales                                        43278 non-null  float64
 31  (gross_profit+interest)/sales                                         43278 non-null  float64
 32  (current_liabilities*365)/cost_of_products_sold                       43037 non-null  float64
 33  operating_expenses/short-term_liabilities                             43271 non-null  float64
 34  operating_expenses_/_total_liabilities                                43311 non-null  float64
 35  profit_on_sales/total_assets                                          43397 non-null  float64
 36  total_sales/total_assets                                              43397 non-null  float64
 37  (current_assets-inventories)/long-term_liabilities                    24421 non-null  float64
 38  constant_capital/total_assets                                         43397 non-null  float64
 39  profit_on_sales/sales                                                 43278 non-null  float64
 40  (current_assets-inventory-receivables)/short-term liabilities         43271 non-null  float64
 41  total_liabilities/((profit_on_operating_activities+depreciation)*(12/365))  42651 non-null  float64
 42  profit_on_operating_activities/sales                                  43278 non-null  float64
 43  rotation_receivables+inventory_turnover_in_days                       43278 non-null  float64
 44  (receivables*365)/sales                                               43278 non-null  float64
 45  net_profit/inventory                                                  41258 non-null  float64
 46  (current_assets-inventory)/short-term_liabilities                     43270 non-null  float64
 47  (inventory*365)/cost_of_products_sold                                 43108 non-null  float64
 48  EBITDA_(profit_on_operating_activities-depreciation)/total_assets     43396 non-null  float64
 49  EBITDA_(profit_on_operating_activities-depreciation)/sales            43278 non-null  float64
 50  current_assets/total_liabilities                                      43311 non-null  float64
 51  short-term_liabilities/total_assets                                   43397 non-null  float64
 52  (short-term_liabilities*365)/cost_of_products_sold)                   43104 non-null  float64
```

```
polish_Dataset.head()
```

	year	net_profit/total assets	Total_Liabilities/total_assets	Working_Capital_to_Total_Asse
0	1	0.200550	0.37951	
1	1	0.209120	0.49988	
2	1	0.248660	0.69592	
3	1	0.081483	0.30734	
4	1	0.187320	0.61323	

```
polish_Dataset.tail()
```

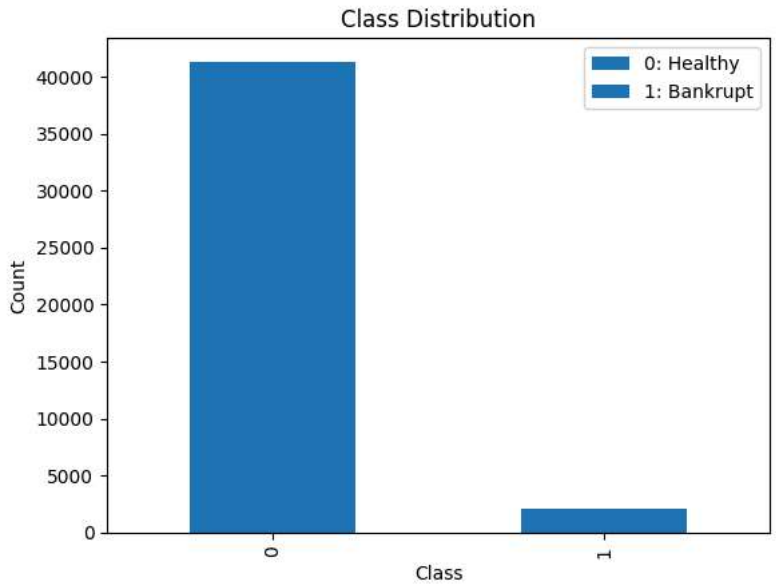
	year	net_profit/total assets	Total_Liabilities/total_assets	Working_Capital_to_Total_
43400	5	0.012898	0.70621	
43401	5	-0.578050	0.96702	
43402	5	-0.179050	1.25530	
43403	5	-0.108860	0.74394	
43404	5	-0.105370	0.53629	

```
filename='alldata.csv'
polish_Dataset.to_csv(filename,index=False)
```

```
files.download('alldata.csv')
```

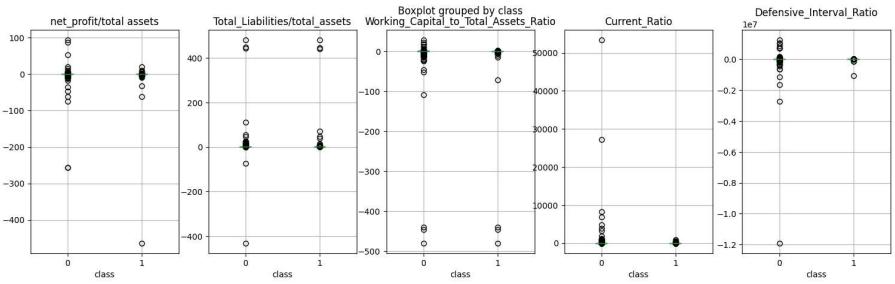
Initial Description Analysis

```
category_counts = polish_Dataset['class'].value_counts()
ax = category_counts.plot(kind='bar')
category_counts.plot(kind='bar')
plt.title('Class Distribution')
plt.xlabel('Class')
plt.ylabel('Count')
plt.legend(['0: Healthy', '1: Bankrupt'])
plt.show()
```

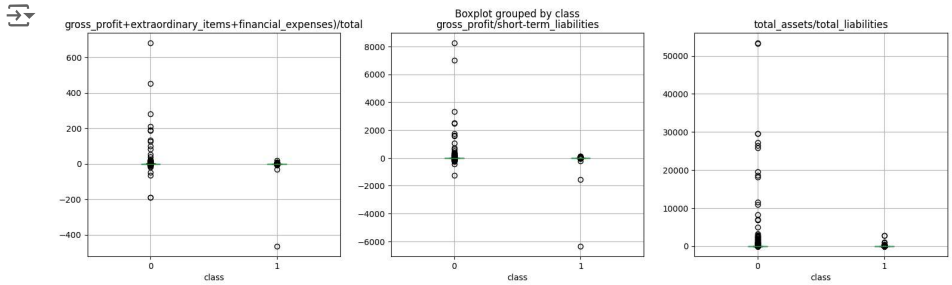



```
def PlotBarCharts(inpData, colsToPlot):
    %matplotlib inline

    import matplotlib.pyplot as plt

    # Generating multiple subplots
    fig, subPlot = plt.subplots(nrows=1, ncols=len(colsToPlot), figsize=(20, 5))
    fig.suptitle('Bar charts of: ' + str(colsToPlot))

    for colName, plotNumber in zip(colsToPlot, range(len(colsToPlot))):
        inpData.groupby(colName).size().plot(kind='bar', ax=subPlot[plotNumber])
```

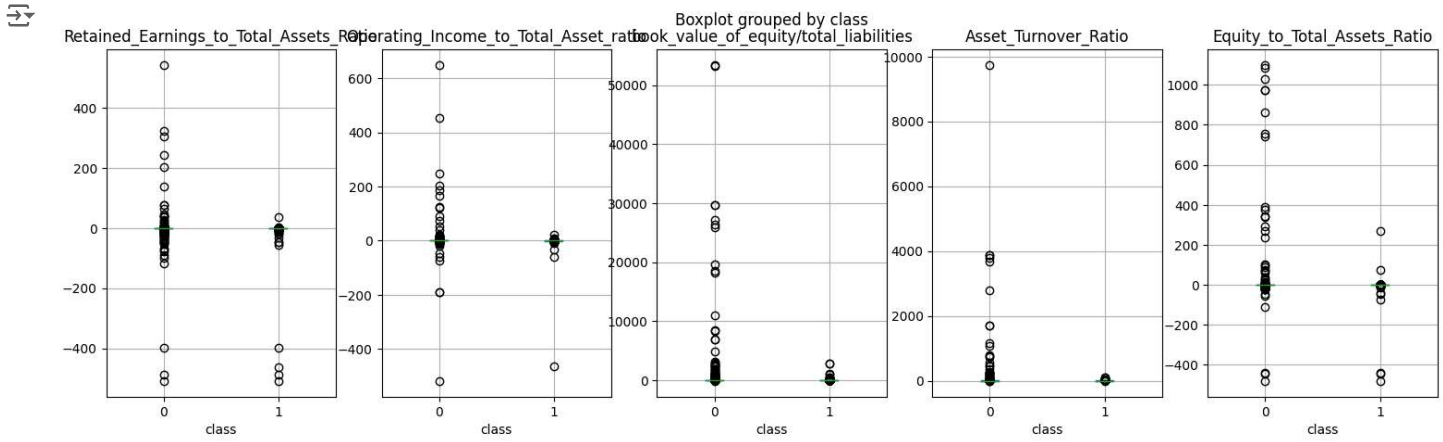
```
import matplotlib.pyplot as plt

continuous_cols = ['net_profit/total assets','Total_Liabilities/total_assets','Working_Capital_to_Total_Assets_Ratio','Current_Ratio','Defer

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```



```
continuous_cols = ['gross_profit+extraordinary_items+financial_expenses)/total','gross_profit/short-term_liabilities','total_assets/total_li

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```

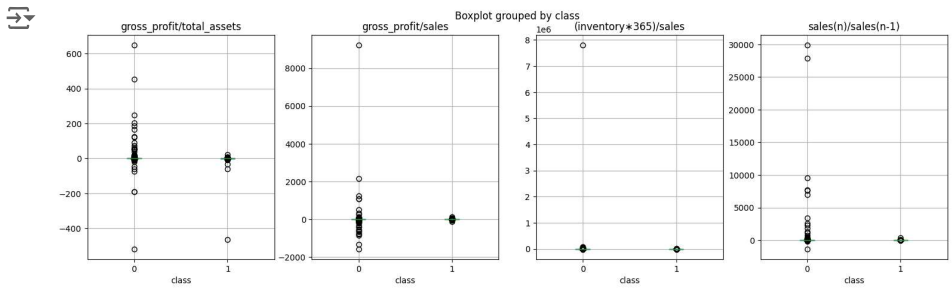



```
continuous_cols = ['Retained_Earnings_to_Total_Assets_Ratio','Operating_Income_to_Total_Asset_ratio','book_value_of_equity/total_liabilities'

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```

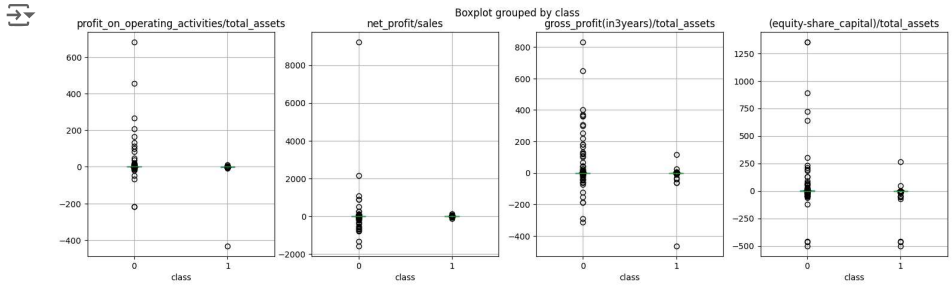



```
continuous_cols = ['gross_profit/total_assets','gross_profit/sales','(inventory*365)/sales','sales(n)/sales(n-1)']

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))
```

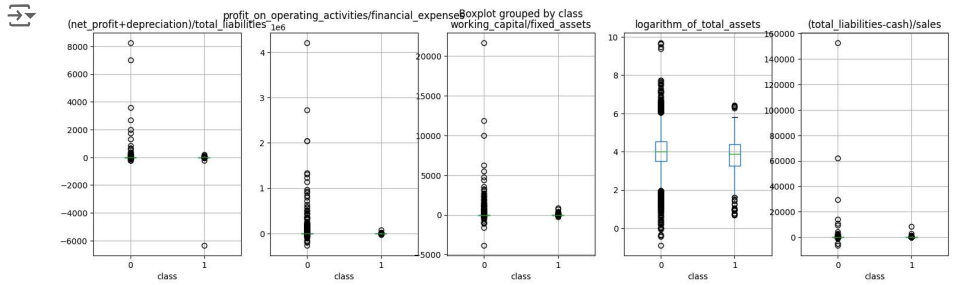
```
# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```



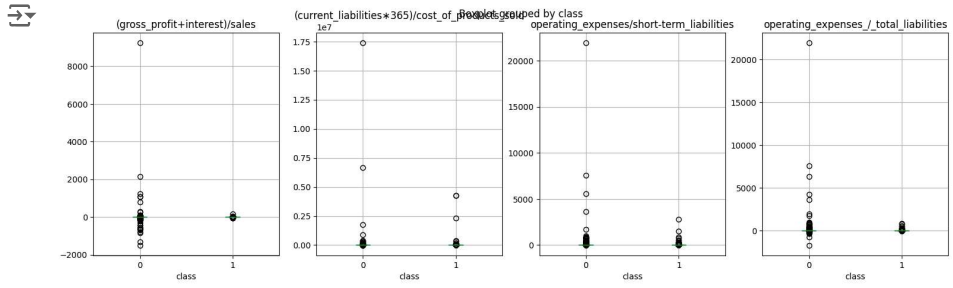

```
continuous_cols = ['profit_on_operating_activities/total_assets','net_profit/sales','gross_profit(in3years)/total_assets','(equity-share_cap

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```



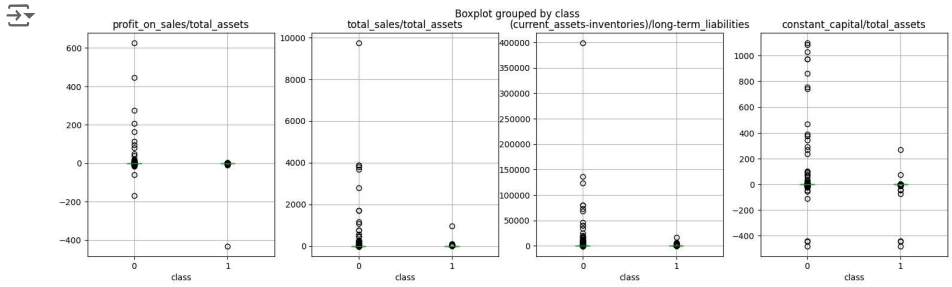
```
continuous_cols = ['(net_profit+depreciation)/total_liabilities','profit_on_operating_activities/financial_expenses','working_capital/fixed_

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```



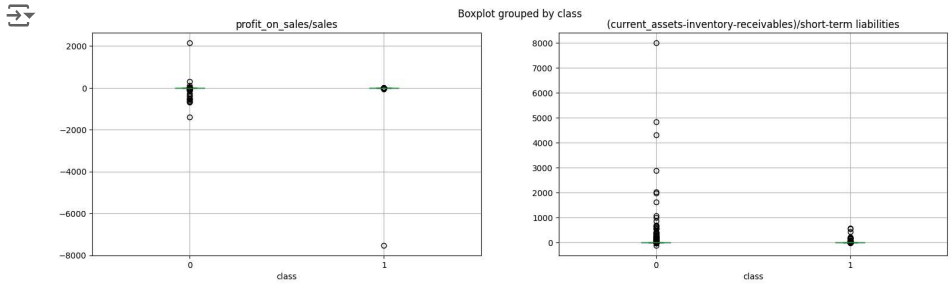

```
continuous_cols = ['(gross_profit+interest)/sales','(current_liabilities*365)/cost_of_products_sold','operating_expenses/short-term_liabilit

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```



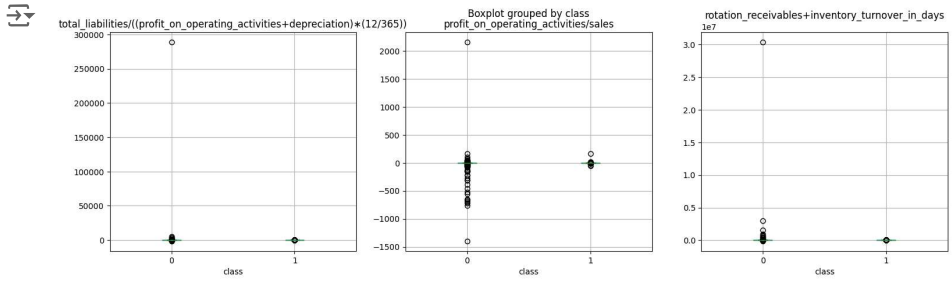
```
continuous_cols = ['profit_on_sales/total_assets','total_sales/total_assets','(current_assets-inventories)/long-term_liabilities','constant_

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```



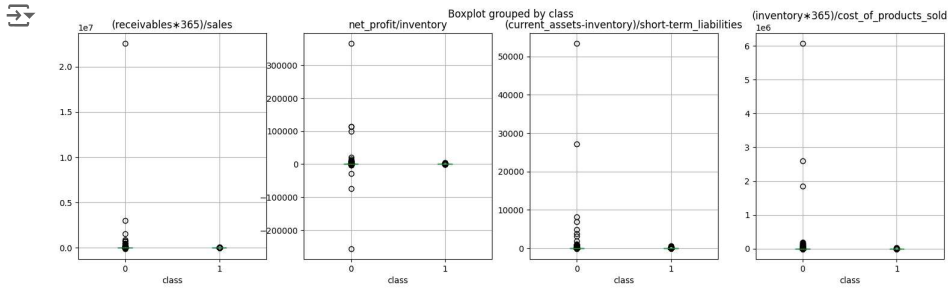

```
continuous_cols = ['profit_on_sales/sales','(current_assets-inventory-receivables)/short-term liabilities']

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```



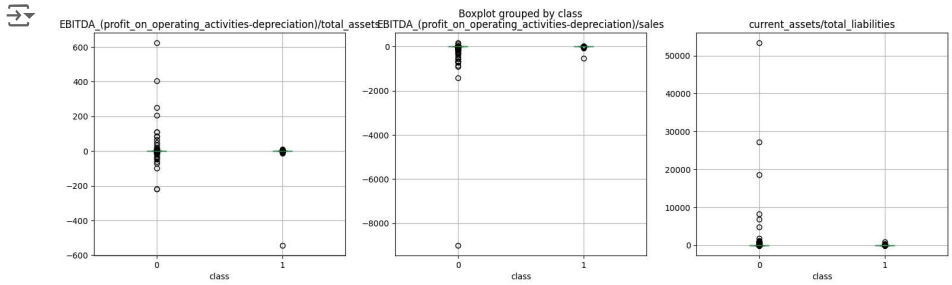
```
continuous_cols = ['total_liabilities/((profit_on_operating_activities+depreciation)*(12/365))','profit_on_operating_activities/sales','rota

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```



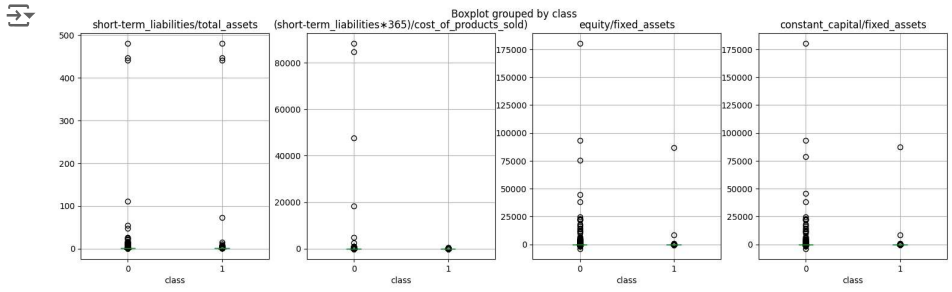

```
continuous_cols = ['(receivables*365)/sales','net_profit/inventory','(current_assets-inventory)/short-term_liabilities','(inventory*365)/cos

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```



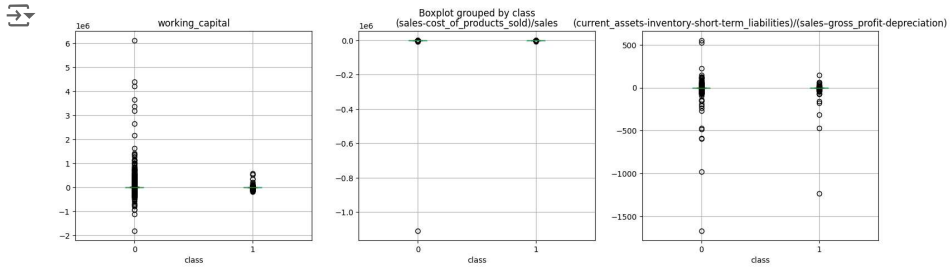
```
continuous_cols = ['EBITDA_(profit_on_operating_activities-depreciation)/total_assets','EBITDA_(profit_on_operating_activities-depreciation)

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```



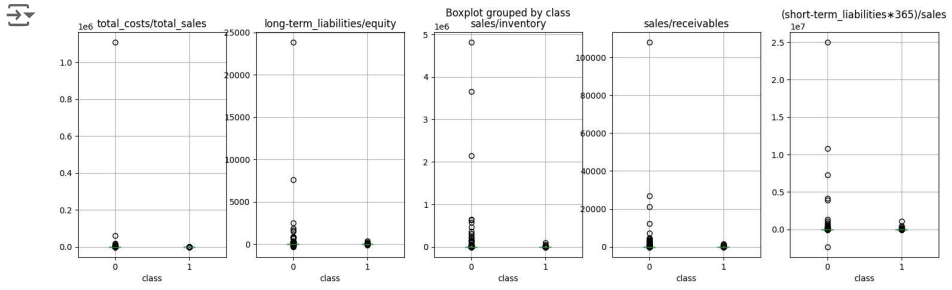

```
continuous_cols = ['short-term_liabilities/total_assets','(short-term_liabilities*365)/cost_of_products_sold)','equity/fixed_assets','consta

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```



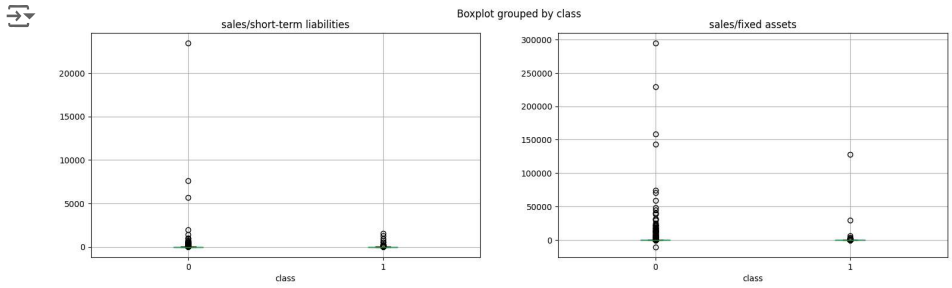
```
continuous_cols = ['working_capital','(sales-cost_of_products_sold)/sales','(current_assets-inventory-short-term_liabilities)/(sales-gross_p

fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```




```
continuous_cols = ['total_costs/total_sales','long-term_liabilities/equity','sales/inventory','sales/receivables','(short-term_liabilities*3
fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```



```
continuous_cols = ['sales/short-term liabilities','sales/fixed assets']
fig, axes = plt.subplots(nrows=1, ncols=len(continuous_cols), figsize=(18, 5))

# Creating box plots for each continuous predictor against the Target Variable "class"
for col, ax in zip(continuous_cols, axes):
    polish_Dataset.boxplot(column=col, by='class', figsize=(5,5), vert=True, ax=ax)
    ax.set_title(col)
plt.show()
```




```
polish_Dataset.isnull().any()
```

```
year                                                                  False
net_profit/total assets                                                True
Total_Liabilities/total_assets                                         True
Working_Capital_to_Total_Assets_Ratio                                  True
Current_Ratio                                                          True
Defensive_Interval_Ratio                                               True
Retained_Earnings_to_Total_Assets_Ratio                                True
Operating_Income_to_Total_Asset_ratio                                  True
book_value_of_equity/total_liabilities                                 True
Asset_Turnover_Ratio                                                   True
Equity_to_Total_Assets_Ratio                                           True
gross_profit+extraordinary_items+financial_expenses)/total             True
gross_profit/short-term_liabilities                                    True
(gross_profit+depreciation)/sales                                      True
(gross_profit+interest)/total_assets                                   True
(total_liabilities*365)/(gross_profit+depreciation)                    True
(gross profit+depreciation)/total_liabilities                          True
total_assets/total_liabilities                                         True
gross_profit/total_assets                                              True
gross_profit/sales                                                     True
(inventory*365)/sales                                                  True
sales(n)/sales(n-1)                                                    True
profit_on_operating_activities/total_assets                            True
net_profit/sales                                                       True
gross_profit(in3years)/total_assets                                    True
(equity-share_capital)/total_assets                                    True
(net_profit+depreciation)/total_liabilities                            True
profit_on_operating_activities/financial_expenses                      True
working_capital/fixed_assets                                           True
logarithm_of_total_assets                                              True
(total_liabilities-cash)/sales                                         True
(gross_profit+interest)/sales                                          True
(current_liabilities*365)/cost_of_products_sold                        True
operating_expenses/short-term_liabilities                              True
operating_expenses_/_total_liabilities                                 True
profit_on_sales/total_assets                                           True
total_sales/total_assets                                               True
(current_assets-inventories)/long-term_liabilities                     True
constant_capital/total_assets                                          True
profit_on_sales/sales                                                  True
(current_assets-inventory-receivables)/short-term liabilities          True
total_liabilities/((profit_on_operating_activities+depreciation)*(12/365))  True
profit_on_operating_activities/sales                                   True
```

```
rotation_receivables+inventory_turnover_in_days                                  True
(receivables*365)/sales                                                          True
net_profit/inventory                                                             True
(current_assets-inventory)/short-term_liabilities                                True
(inventory*365)/cost_of_products_sold                                            True
EBITDA_(profit_on_operating_activities-depreciation)/total_assets                True
EBITDA_(profit_on_operating_activities-depreciation)/sales                       True
current_assets/total_liabilities                                                 True
short-term_liabilities/total_assets                                              True
(short-term_liabilities*365)/cost_of_products_sold)                              True
equity/fixed_assets                                                              True
constant_capital/fixed_assets                                                    True
working_capital                                                                  True
(sales-cost_of_products_sold)/sales                                              True
(current_assets-inventory-short-term_liabilities)/(sales-gross_profit-depreciation)  True
```

```
#number of null
Features_Null=(polish_Dataset.isnull().sum())
Features_Null
```

```
year                                                                              0
net_profit/total assets                                                           8
Total_Liabilities/total_assets                                                    8
Working_Capital_to_Total_Assets_Ratio                                             8
Current_Ratio                                                                   134
Defensive_Interval_Ratio                                                         89
Retained_Earnings_to_Total_Assets_Ratio                                           8
Operating_Income_to_Total_Asset_ratio                                             8
book_value_of_equity/total_liabilities                                           94
Asset_Turnover_Ratio                                                              9
Equity_to_Total_Assets_Ratio                                                      8
gross_profit+extraordinary_items+financial_expenses)/total                       44
gross_profit/short-term_liabilities                                             134
(gross_profit+depreciation)/sales                                               127
(gross_profit+interest)/total_assets                                              8
(total_liabilities*365)/(gross_profit+depreciation)                              36
(gross profit+depreciation)/total_liabilities                                    95
total_assets/total_liabilities                                                   94
gross_profit/total_assets                                                         8
gross_profit/sales                                                              128
(inventory*365)/sales                                                           127
sales(n)/sales(n-1)                                                            5854
profit_on_operating_activities/total_assets                                       8
net_profit/sales                                                                127
gross_profit(in3years)/total_assets                                             922
(equity-share_capital)/total_assets                                               8
(net_profit+depreciation)/total_liabilities                                      95
profit_on_operating_activities/financial_expenses                              2764
working_capital/fixed_assets                                                    812
logarithm_of_total_assets                                                         8
(total_liabilities-cash)/sales                                                  127
(gross_profit+interest)/sales                                                   127
(current_liabilities*365)/cost_of_products_sold                                 368
operating_expenses/short-term_liabilities                                       134
operating_expenses_/_total_liabilities                                           94
profit_on_sales/total_assets                                                      8
total_sales/total_assets                                                          8
(current_assets-inventories)/long-term_liabilities                            18984
constant_capital/total_assets                                                     8
profit_on_sales/sales                                                           127
(current_assets-inventory-receivables)/short-term liabilities                   134
total_liabilities/((profit_on_operating_activities+depreciation)*(12/365))      754
profit_on_operating_activities/sales                                            127
rotation_receivables+inventory_turnover_in_days                                 127
(receivables*365)/sales                                                         127
net_profit/inventory                                                           2147
(current_assets-inventory)/short-term_liabilities                               135
(inventory*365)/cost_of_products_sold                                           297
EBITDA_(profit_on_operating_activities-depreciation)/total_assets                 9
EBITDA_(profit_on_operating_activities-depreciation)/sales                      127
current_assets/total_liabilities                                                 94
short-term_liabilities/total_assets                                               8
(short-term_liabilities*365)/cost_of_products_sold)                             301
equity/fixed_assets                                                             812
constant_capital/fixed_assets                                                   812
working_capital                                                                   1
(sales-cost_of_products_sold)/sales                                             127
(current_assets-inventory-short-term_liabilities)/(sales-gross_profit-depreciation)   7
```

```
polish_Dataset.corr()
```

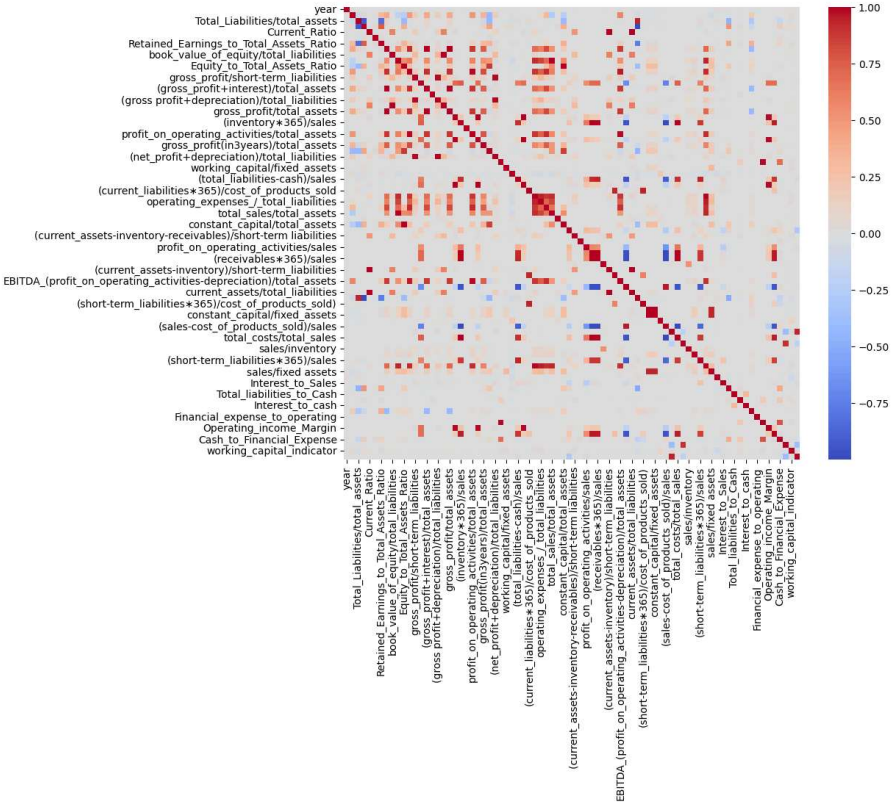
	year	net_profit/tot asse
year	1.000000	-0.004(
net_profit/total assets	-0.004052	1.000(
Total_Liabilities/total_assets	-0.004834	-0.1041
Working_Capital_to_Total_Assets_Ratio	0.004653	0.058(
Current_Ratio	0.004548	0.000{
Defensive_Interval_Ratio	0.001471	0.002{
Retained_Earnings_to_Total_Assets_Ratio	-0.000409	0.3151
Operating_Income_to_Total_Asset_ratio	-0.019313	0.4084
book_value_of_equity/total_liabilities	0.005614	0.0001
Asset_Turnover_Ratio	-0.017984	-0.225{
Equity_to_Total_Assets_Ratio	-0.021114	-0.240{
gross_profit+extraordinary_items+financial_expenses)/total	-0.019025	0.4921
gross_profit/short-term_liabilities	-0.000910	0.168{
(gross_profit+depreciation)/sales	-0.006130	0.001{
(gross_profit+interest)/total_assets	-0.019312	0.4084
(total_liabilities∗365)/(gross_profit+depreciation)	0.002806	0.000{
(gross profit+depreciation)/total_liabilities	-0.000490	0.163{
total_assets/total_liabilities	0.005595	0.000{
gross_profit/total_assets	-0.018880	0.406{
gross_profit/sales	-0.004937	0.003{
(inventory∗365)/sales	-0.007237	-0.000(
sales(n)/sales(n-1)	-0.009455	-0.001{
profit_on_operating_activities/total_assets	-0.016220	0.5013
net_profit/sales	-0.005043	0.003{
gross_profit(in3years)/total_assets	-0.011841	0.334{
(equity-share_capital)/total_assets	-0.016473	-0.413{
(net_profit+depreciation)/total_liabilities	-0.000569	0.169(
profit_on_operating_activities/financial_expenses	-0.006109	0.007{
working_capital/fixed_assets	0.013252	0.002{
logarithm_of_total_assets	-0.015942	0.039{
(total_liabilities-cash)/sales	-0.007532	0.000(
(gross_profit+interest)/sales	-0.004699	0.003{
(current_liabilities∗365)/cost_of_products_sold	0.001381	-0.002{
operating_expenses/short-term_liabilities	-0.000124	0.005{
operating_expenses_/_total_liabilities	-0.000522	0.083{
profit_on_sales/total_assets	-0.018639	0.313{
total_sales/total_assets	-0.017789	-0.279{
(current_assets-inventories)/long-term_liabilities	-0.006016	0.001{
constant_capital/total_assets	-0.020650	-0.233{
profit_on_sales/sales	-0.005304	0.000{
(current_assets-inventory-receivables)/short-term liabilities	0.009845	0.002{
total_liabilities/((profit_on_operating_activities+depreciation)∗(12/365))	0.000513	0.0001
profit_on_operating_activities/sales	-0.005691	0.002{
rotation_receivables+inventory_turnover_in_days	0.006035	0.000(

rotation_receivables+inventory_turnover_in_days	-0.006935	-0.0000
(receivables*365)/sales	-0.006808	-0.0000
net_profit/inventory	0.008052	0.0009
(current_assets-inventory)/short-term_liabilities	0.004457	0.0006
(inventory*365)/cost_of_products_sold	-0.002627	-0.0008
EBITDA_(profit_on_operating_activities-depreciation)/total_assets	-0.013376	0.5832
EBITDA_(profit_on_operating_activities-depreciation)/sales	0.006215	0.0011
current_assets/total_liabilities	0.003010	0.0009
short-term_liabilities/total_assets	-0.003462	-0.0564
(short-term_liabilities*365)/cost_of_products_sold)	0.001655	-0.0014
equity/fixed_assets	-0.019033	-0.0317
constant_capital/fixed_assets	-0.018786	-0.0315
working_capital	0.009445	0.0040
(sales-cost_of_products_sold)/sales	0.007262	0.0000
(current_assets-inventory-short-term_liabilities)/(sales–gross_profit-depreciation)	-0.006660	0.0077
total_costs/total_sales	-0.007609	-0.0003
long-term_liabilities/equity	-0.002881	-0.0002
sales/inventory	0.001804	0.0002
sales/receivables	0.000938	0.0005
(short-term_liabilities*365)/sales	-0.006390	0.0003
sales/short-term liabilities	-0.001623	0.0083
sales/fixed assets	-0.016681	-0.0690
class	0.043488	-0.0266
Interest_to_Sales	0.005840	-0.0000
Cash_to_assets	0.001698	-0.0343
Total_liabilities_to_Cash	-0.007498	-0.0021
cash_to_interest	0.012476	0.0014
Interest_to_cash	-0.002826	0.0025
Financial_Expense_To_Total_Asset	0.002699	0.1714
Financial_expense_to_operating	-0.009278	0.0001
Debt_Coverage_Ratio	-0.006008	0.0065
Operating_income_Margin	-0.006338	0.0030
Cash_conversion_cycle	-0.007289	-0.0003
Cash_to_Financial_Expense	0.002693	-0.0044
Financial_Leverage	-0.001960	-0.0003
working_capital_indicator	0.013728	0.0336
Return_On_equity	-0.004974	0.0043

```
correlation_matrix = polish_Dataset.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm', fmt=".2f")
plt.show
```

```
matplotlib.pyplot.show
def show(*args, **kwargs)
```

```
Display all open figures.

Parameters
----------
block : bool, optional
    Whether to wait for all figures to be closed before returning.
```




```
# based on initial observations, there are many extermine outliers in this dataset. The outliers are extreme in the 0 dataset.
#I will review each attribute and based on industry and financial experience will narrow the feature selections,
#before conducting any cleaning.
```

Approach

After careful analyzing all the variablea. The theme of selection came down to: **Reason for companies to go bankrupt:**

1. **Failure of Business Strategy:** a. Lack of governance i. Failure to understand key metrics can lead to bankruptcy. b. Market trends i. Failing to see what competitors are doing ii. Market sentiment shifts and not adjusting
2. ** Inefficient use of assets** a. Management fails to generate income from assets b. Holding inventory too long c. Negative Working Capital
3. **Too much Debt** a. Debt used to pay for Working capital b. Debt used to pay for equipment, plant c. Operating profits can't cover interest
4. ** Not enough Cash flow** a. Not enough cash on hand b. Not collecting from account receivables c. Sales are not enough to cover expenses d. Cash conversation cycle is too long e. Cash can't cover interest f. Days payable is too high

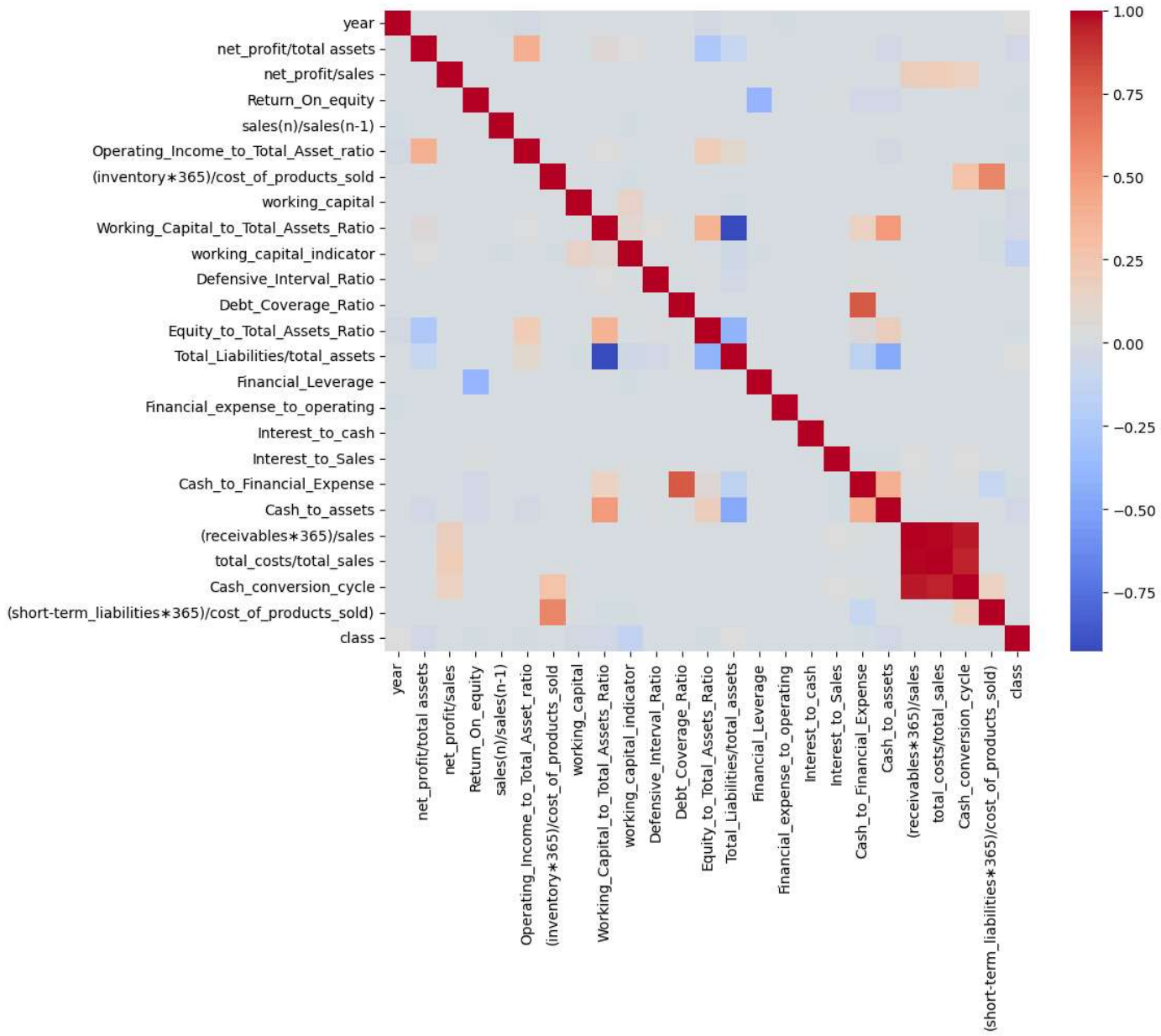
```
#analysis and data cleaning will be done on this new dataset. The exisiting dataset, will be used to compare results.
Bankrupt_model = polish_Dataset[['year','net_profit/total assets', 'net_profit/sales','Return_On_equity', 'sales(n)/sales(n-1)', 'Operating_
Bankrupt_model.head()
```

	year	net_profit/total assets	net_profit/sales	Return_On_equity	sales(n)/sales(n-1)	Operating_Income_to_Total_Asset_ratio	(inventory*365)
0	1	0.200550	0.119980	0.413683	1.2479	0.249760	
1	1	0.209120	0.123040	0.420018	1.4293	0.258340	
2	1	0.248660	0.189960	0.817738	1.4283	0.309060	
3	1	0.081483	0.062782	0.148458	1.5069	0.092704	
4	1	0.187320	0.115530	0.486231	NaN	0.187320	

```
plt.figure(figsize=(10, 8))
sns.heatmap(Bankrupt_model.corr(), annot=False, cmap='coolwarm', fmt=".2f")
plt.show
```

```
matplotlib.pyplot.show
def show(*args, **kwargs)
```

```
Display all open figures.

Parameters
----------
block : bool, optional
    Whether to wait for all figures to be closed before returning.
```




```
pd.options.display.float_format = "{:.3f}".format
Bankrupt_model.describe()
```

	year	net_profit/total assets	net_profit/sales	Return_On_equity	sales(n)/sales(n 1
count	43405.000	43397.000	43278.000	43266.000	37551.00
mean	2.940	0.035	0.139	0.010	3.88
std	1.284	2.994	48.335	15.320	228.66
min	1.000	-463.890	-1578.700	-1858.486	-1325.00
25%	2.000	0.003	0.002	0.017	0.90
50%	3.000	0.050	0.030	0.123	1.04
75%	4.000	0.130	0.078	0.289	1.20
max	5.000	94.280	9230.500	1031.911	29907.00

#exterime outliers are detected with the first analysis. will use a range with a multiper to remove the exterme outliers.

```
Bankrupt_model.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43405 entries, 0 to 43404
Data columns (total 25 columns):
 #   Column                                            Non-Null Count  Dtype
---  ------                                            --------------  -----
 0   year                                              43405 non-null  int64
 1   net_profit/total assets                           43397 non-null  float64
 2   net_profit/sales                                  43278 non-null  float64
 3   Return_On_equity                                  43266 non-null  float64
 4   sales(n)/sales(n-1)                               37551 non-null  float64
 5   Operating_Income_to_Total_Asset_ratio             43397 non-null  float64
 6   (inventory*365)/cost_of_products_sold             43108 non-null  float64
 7   working_capital                                   43404 non-null  float64
 8   Working_Capital_to_Total_Assets_Ratio             43397 non-null  float64
 9   working_capital_indicator                         43405 non-null  category
 10  Defensive_Interval_Ratio                          43316 non-null  float64
 11  Debt_Coverage_Ratio                               37652 non-null  float64
 12  Equity_to_Total_Assets_Ratio                      43397 non-null  float64
 13  Total_Liabilities/total_assets                    43397 non-null  float64
 14  Financial_Leverage                                43393 non-null  float64
 15  Financial_expense_to_operating                    37635 non-null  float64
 16  Interest_to_cash                                  23511 non-null  float64
 17  Interest_to_Sales                                 43277 non-null  float64
 18  Cash_to_Financial_Expense                         37567 non-null  float64
 19  Cash_to_assets                                    43270 non-null  float64
 20  (receivables*365)/sales                           43278 non-null  float64
 21  total_costs/total_sales                           43321 non-null  float64
 22  Cash_conversion_cycle                             42986 non-null  float64
 23  (short-term_liabilities*365)/cost_of_products_sold)  43104 non-null  float64
 24  class                                             43405 non-null  category
dtypes: category(2), float64(22), int64(1)
memory usage: 7.7 MB
```

```
#adjusting for outliers
numeric_cols = Bankrupt_model.select_dtypes(include=[float, int]).columns
numeric_cols
```

```
Index(['year', 'net_profit/total assets', 'net_profit/sales', 'Return_On_equity',
       'sales(n)/sales(n-1)', 'Operating_Income_to_Total_Asset_ratio',
       '(inventory*365)/cost_of_products_sold', 'working_capital',
       'Working_Capital_to_Total_Assets_Ratio', 'Defensive_Interval_Ratio', 'Debt_Coverage_Ratio',
       'Equity_to_Total_Assets_Ratio', 'Total_Liabilities/total_assets', 'Financial_Leverage',
       'Financial_expense_to_operating', 'Interest_to_cash', 'Interest_to_Sales',
       'Cash_to_Financial_Expense', 'Cash_to_assets', '(receivables*365)/sales',
       'total_costs/total_sales', 'Cash_conversion_cycle',
       '(short-term_liabilities*365)/cost_of_products_sold)'],
      dtype='object')
```

Start coding or generate with AI.

```
#adjusting for outliers
numeric_cols = Bankrupt_model.select_dtypes(include=[float, int]).columns
# Calculate Q1, Q3, and IQR for each column
Q1 = Bankrupt_model[numeric_cols].quantile(0.25)
Q3 = Bankrupt_model[numeric_cols].quantile(0.75)
IQR = Q3 - Q1

# Define the lower and upper bounds for outliers
lower_bound = Q1 - 5 * IQR
upper_bound = Q3 + 5 * IQR

# Create a boolean DataFrame indicating whether each value is an outlier
outliers = (Bankrupt_model[numeric_cols] < lower_bound) | (Bankrupt_model[numeric_cols] > upper_bound)

# Filter rows that contain any outliers
rows_with_outliers = outliers.any(axis=1)
Bankruptcleaned_model = Bankrupt_model[~rows_with_outliers]

Bankruptcleaned_model.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 22828 entries, 3 to 43404
Data columns (total 25 columns):
 #   Column                                            Non-Null Count  Dtype
---  ------                                            --------------  -----
 0   year                                              22828 non-null  int64
 1   net_profit/total assets                           22824 non-null  float64
 2   net_profit/sales                                  22812 non-null  float64
 3   Return_On_equity                                  22806 non-null  float64
 4   sales(n)/sales(n-1)                               20141 non-null  float64
 5   Operating_Income_to_Total_Asset_ratio             22824 non-null  float64
 6   (inventory*365)/cost_of_products_sold             22822 non-null  float64
 7   working_capital                                   22827 non-null  float64
 8   Working_Capital_to_Total_Assets_Ratio             22824 non-null  float64
 9   working_capital_indicator                         22828 non-null  category
 10  Defensive_Interval_Ratio                          22820 non-null  float64
 11  Debt_Coverage_Ratio                               20657 non-null  float64
 12  Equity_to_Total_Assets_Ratio                      22824 non-null  float64
 13  Total_Liabilities/total_assets                    22824 non-null  float64
 14  Financial_Leverage                                22822 non-null  float64
 15  Financial_expense_to_operating                    20656 non-null  float64
 16  Interest_to_cash                                  10983 non-null  float64
 17  Interest_to_Sales                                 22812 non-null  float64
 18  Cash_to_Financial_Expense                         20649 non-null  float64
 19  Cash_to_assets                                    22808 non-null  float64
 20  (receivables*365)/sales                           22812 non-null  float64
 21  total_costs/total_sales                           22813 non-null  float64
 22  Cash_conversion_cycle                             22810 non-null  float64
 23  (short-term_liabilities*365)/cost_of_products_sold)  22820 non-null  float64
 24  class                                             22828 non-null  category
dtypes: category(2), float64(22), int64(1)
memory usage: 4.2 MB
```

```
print(len(Bankrupt_model))
print(len(Bankruptcleaned_model))
```

```
43405
22828
```

```
Bankruptcleaned_model.describe()
```

	year	net_profit/total assets	net_profit/sales	Return_On_equity	sales(n)/sales(n 1
count	22828.000	22824.000	22812.000	22806.000	20141.00
mean	2.917	0.072	0.040	0.152	1.07
std	1.302	0.126	0.079	0.286	0.28
min	1.000	-0.618	-0.376	-1.338	0.00
25%	2.000	0.015	0.009	0.036	0.92
50%	3.000	0.059	0.034	0.129	1.05
75%	4.000	0.127	0.074	0.269	1.20
max	5.000	0.755	0.457	1.645	2.67

```
Bankruptcleaned_model.corr()
```

	year	net_profit/total assets	net_profit/sales	Retu
year	1.000	-0.099	-0.065	
net_profit/total assets	-0.099	1.000	0.827	
net_profit/sales	-0.065	0.827	1.000	
Return_On_equity	-0.096	0.692	0.582	
sales(n)/sales(n-1)	0.017	0.253	0.220	
Operating_Income_to_Total_Asset_ratio	-0.106	0.986	0.811	
(inventory*365)/cost_of_products_sold	0.032	-0.121	-0.038	
working_capital	0.031	0.180	0.232	
Working_Capital_to_Total_Assets_Ratio	0.030	0.329	0.284	
working_capital_indicator	0.017	0.209	0.203	
Defensive_Interval_Ratio	0.028	0.259	0.312	
Debt_Coverage_Ratio	-0.046	0.526	0.471	
Equity_to_Total_Assets_Ratio	0.016	0.291	0.311	
Total_Liabilities/total_assets	-0.016	-0.282	-0.307	
Financial_Leverage	-0.034	-0.082	-0.105	
Financial_expense_to_operating	0.006	0.060	0.068	
Interest_to_cash	0.003	-0.076	-0.068	
Interest_to_Sales	0.033	-0.103	-0.080	
Cash_to_Financial_Expense	0.027	0.101	0.099	
Cash_to_assets	0.004	0.228	0.192	
(receivables*365)/sales	0.085	-0.064	0.027	
total_costs/total_sales	0.051	-0.566	-0.673	
Cash_conversion_cycle	0.077	-0.132	-0.011	
(short-term_liabilities*365)/cost_of_products_sold)	0.024	-0.262	-0.188	
class	0.020	-0.132	-0.140	

```
corr_pairs_negative = Bankruptcleaned_model.corr().unstack().sort_values(ascending=False)
corr_pairs_negative = corr_pairs_negative[corr_pairs_negative < -0.4]
corr_pairs_negative = corr_pairs_negative[corr_pairs_negative < 1].drop_duplicates()
pd.DataFrame(corr_pairs_negative)
corr_pairs_negative
```

```
Total_Liabilities/total_assets                    working_capital_indicator                          -0.418
total_costs/total_sales                           Debt_Coverage_Ratio                                -0.418
Total_Liabilities/total_assets                    Defensive_Interval_Ratio                           -0.486
(short-term_liabilities*365)/cost_of_products_sold)  Equity_to_Total_Assets_Ratio                    -0.507
                                                  Working_Capital_to_Total_Assets_Ratio              -0.540
Defensive_Interval_Ratio                          (short-term_liabilities*365)/cost_of_products_sold)  -0.559
Operating_Income_to_Total_Asset_ratio             total_costs/total_sales                            -0.566
total_costs/total_sales                           net_profit/total assets                            -0.566
Total_Liabilities/total_assets                    Working_Capital_to_Total_Assets_Ratio              -0.637
net_profit/sales                                  total_costs/total_sales                            -0.673
Total_Liabilities/total_assets                    Equity_to_Total_Assets_Ratio                       -0.981
dtype: float64
```

```
for column in Bankruptcleaned_model.columns:
  countzero = (Bankruptcleaned_model[column] == 0).sum()
  print(f"{column} has {countzero} zero values")
```

```
year has 0 zero values
net_profit/total assets has 70 zero values
net_profit/sales has 66 zero values
Return_On_equity has 65 zero values
sales(n)/sales(n-1) has 4 zero values
```

```
Operating_Income_to_Total_Asset_ratio has 9 zero values
(inventory*365)/cost_of_products_sold has 682 zero values
working_capital has 2 zero values
Working_Capital_to_Total_Assets_Ratio has 0 zero values
working_capital_indicator has 3965 zero values
Defensive_Interval_Ratio has 11 zero values
Debt_Coverage_Ratio has 2 zero values
Equity_to_Total_Assets_Ratio has 2 zero values
Total_Liabilities/total_assets has 8 zero values
Financial_Leverage has 0 zero values
Financial_expense_to_operating has 1 zero values
Interest_to_cash has 0 zero values
Interest_to_Sales has 11828 zero values
Cash_to_Financial_Expense has 0 zero values
Cash_to_assets has 1 zero values
(receivables*365)/sales has 26 zero values
total_costs/total_sales has 0 zero values
Cash_conversion_cycle has 10 zero values
(short-term_liabilities*365)/cost_of_products_sold) has 23 zero values
class has 21703 zero values
```

```
length=len(Bankruptcleaned_model['class'])
print(f"total number of rows is: {length}")
```

```
total number of rows is: 22828
```

```
nulls=Bankruptcleaned_model.isnull().sum()
```

```
nulls
```

```
year                                                 0
net_profit/total assets                              4
net_profit/sales                                    16
Return_On_equity                                    22
sales(n)/sales(n-1)                               2687
Operating_Income_to_Total_Asset_ratio                4
(inventory*365)/cost_of_products_sold                6
working_capital                                      1
Working_Capital_to_Total_Assets_Ratio                4
working_capital_indicator                            0
Defensive_Interval_Ratio                             8
Debt_Coverage_Ratio                               2171
Equity_to_Total_Assets_Ratio                         4
Total_Liabilities/total_assets                       4
Financial_Leverage                                   6
Financial_expense_to_operating                    2172
Interest_to_cash                                 11845
Interest_to_Sales                                   16
Cash_to_Financial_Expense                         2179
Cash_to_assets                                      20
(receivables*365)/sales                             16
total_costs/total_sales                             15
Cash_conversion_cycle                               18
(short-term_liabilities*365)/cost_of_products_sold)  8
class                                                0
dtype: int64
```

```
percentage_nulls=nulls/length*100
percentage_nulls
#to determine % of null per attribute. anything over 10% will be removed. The remainder will use random forest imputation.
```

```
year                                                 0.000
net_profit/total assets                              0.018
net_profit/sales                                     0.070
Return_On_equity                                     0.096
sales(n)/sales(n-1)                                 11.771
Operating_Income_to_Total_Asset_ratio                0.018
(inventory*365)/cost_of_products_sold                0.026
working_capital                                      0.004
Working_Capital_to_Total_Assets_Ratio                0.018
working_capital_indicator                            0.000
Defensive_Interval_Ratio                             0.035
Debt_Coverage_Ratio                                  9.510
Equity_to_Total_Assets_Ratio                         0.018
Total_Liabilities/total_assets                       0.018
Financial_Leverage                                   0.026
Financial_expense_to_operating                       9.515
Interest_to_cash                                    51.888
Interest_to_Sales                                    0.070
Cash_to_Financial_Expense                            9.545
```

```
Cash_to_assets                                    0.088
(receivables*365)/sales                           0.070
total_costs/total_sales                           0.066
Cash_conversion_cycle                             0.079
(short-term_liabilities*365)/cost_of_products_sold    0.035
class                                             0.000
dtype: float64
```

```
percentage_nulls[percentage_nulls>5]
```

```
sales(n)/sales(n-1)              11.771
Debt_Coverage_Ratio               9.510
Financial_expense_to_operating    9.515
Interest_to_cash                 51.888
Cash_to_Financial_Expense         9.545
dtype: float64
```

```
Bankruptcleaned_model.drop(['sales(n)/sales(n-1)','Debt_Coverage_Ratio','Financial_expense_to_operating','Interest_to_cash','Cash_to_Financi
```

```
<ipython-input-85-65fb679f6fe4>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
  Bankruptcleaned_model.drop(['sales(n)/sales(n-1)','Debt_Coverage_Ratio','Financial_expense_to_operating','Interest_to_cash','Cash_to_F
```

```
Bankruptcleaned_model.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 22828 entries, 3 to 43404
Data columns (total 20 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   year                                            22828 non-null  int64
 1   net_profit/total assets                         22824 non-null  float64
 2   net_profit/sales                                22812 non-null  float64
 3   Return_On_equity                                22806 non-null  float64
 4   Operating_Income_to_Total_Asset_ratio           22824 non-null  float64
 5   (inventory*365)/cost_of_products_sold           22822 non-null  float64
 6   working_capital                                 22827 non-null  float64
 7   Working_Capital_to_Total_Assets_Ratio           22824 non-null  float64
 8   working_capital_indicator                       22828 non-null  category
 9   Defensive_Interval_Ratio                        22820 non-null  float64
 10  Equity_to_Total_Assets_Ratio                    22824 non-null  float64
 11  Total_Liabilities/total_assets                  22824 non-null  float64
 12  Financial_Leverage                              22822 non-null  float64
 13  Interest_to_Sales                               22812 non-null  float64
 14  Cash_to_assets                                  22808 non-null  float64
 15  (receivables*365)/sales                         22812 non-null  float64
 16  total_costs/total_sales                         22813 non-null  float64
 17  Cash_conversion_cycle                           22810 non-null  float64
 18  (short-term_liabilities*365)/cost_of_products_sold  22820 non-null  float64
 19  class                                           22828 non-null  category
dtypes: category(2), float64(17), int64(1)
memory usage: 3.4 MB
```

```
nulls=Bankruptcleaned_model.isnull().sum()
nulls
```

```
year                                              0
net_profit/total assets                           4
net_profit/sales                                 16
Return_On_equity                                 22
Operating_Income_to_Total_Asset_ratio             4
(inventory*365)/cost_of_products_sold             6
working_capital                                   1
Working_Capital_to_Total_Assets_Ratio             4
working_capital_indicator                         0
Defensive_Interval_Ratio                          8
Equity_to_Total_Assets_Ratio                      4
Total_Liabilities/total_assets                    4
Financial_Leverage                                6
Interest_to_Sales                                16
Cash_to_assets                                   20
(receivables*365)/sales                          16
total_costs/total_sales                          15
Cash_conversion_cycle                            18
(short-term_liabilities*365)/cost_of_products_sold     8
```

```
class                                                 0
dtype: int64
```

```
nulls/length*100
```

```
year                                                0.000
net_profit/total assets                             0.018
net_profit/sales                                    0.070
Return_On_equity                                    0.096
Operating_Income_to_Total_Asset_ratio               0.018
(inventory*365)/cost_of_products_sold               0.026
working_capital                                     0.004
Working_Capital_to_Total_Assets_Ratio               0.018
working_capital_indicator                           0.000
Defensive_Interval_Ratio                            0.035
Equity_to_Total_Assets_Ratio                        0.018
Total_Liabilities/total_assets                      0.018
Financial_Leverage                                  0.026
Interest_to_Sales                                   0.070
Cash_to_assets                                      0.088
(receivables*365)/sales                             0.070
total_costs/total_sales                             0.066
Cash_conversion_cycle                               0.079
(short-term_liabilities*365)/cost_of_products_sold) 0.035
class                                               0.000
dtype: float64
```

```
#using 3 different methods of imputation to see which one works best
Bankrupt_modelMMM=Bankruptcleaned_model.copy()
Bankrupt_modelKNN=Bankruptcleaned_model.copy()
Bankrupt_modelRF=Bankruptcleaned_model.copy()
```

```
from sklearn.impute import SimpleImputer
from sklearn.experimental import enable_iterative_imputer  # noqa
from sklearn.impute import IterativeImputer
from sklearn.ensemble import RandomForestRegressor
from sklearn.impute import KNNImputer
```

```
# Columns with missing values in your bankrupt_model dataset
columns_with_missing = [
    'net_profit/total assets', 'net_profit/sales', 'Return_On_equity',
    'Operating_Income_to_Total_Asset_ratio',
    '(inventory*365)/cost_of_products_sold', 'working_capital',
    'Working_Capital_to_Total_Assets_Ratio', 'working_capital_indicator',
    'Defensive_Interval_Ratio', 'Equity_to_Total_Assets_Ratio',
    'Total_Liabilities/total_assets', 'Financial_Leverage',
    'Interest_to_Sales', 'Cash_to_assets', '(receivables*365)/sales',
    'total_costs/total_sales', 'Cash_conversion_cycle',
    '(short-term_liabilities*365)/cost_of_products_sold)'
]
```

```
# Perform imputation (example using SimpleImputer with mean)
imputer = SimpleImputer(strategy='mean')
Bankrupt_modelMMM[columns_with_missing] = imputer.fit_transform(Bankrupt_modelMMM[columns_with_missing])
```

```
#performing imputation with KNN
imputer=KNNImputer(n_neighbors=5)
Bankrupt_modelKNN[columns_with_missing] = imputer.fit_transform(Bankrupt_modelKNN[columns_with_missing])
```

```
#perfomring imputation with Random Forest
estimator = RandomForestRegressor(n_estimators=3, max_depth= 3)
imputer = IterativeImputer(estimator=estimator, random_state=0)
Bankrupt_modelRF[columns_with_missing] = imputer.fit_transform(Bankrupt_modelRF[columns_with_missing])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/impute/_iterative.py:785: ConvergenceWarning: [IterativeImputer] Early stopping criteric
  warnings.warn(
```

```
Bankrupt_modelMMM.isnull().sum()
```

```
year                                                0
net_profit/total assets                             0
net_profit/sales                                    0
Return_On_equity                                    0
Operating_Income_to_Total_Asset_ratio               0
(inventory*365)/cost_of_products_sold               0
working_capital                                     0
Working_Capital_to_Total_Assets_Ratio               0
working_capital_indicator                           0
Defensive_Interval_Ratio                            0
Equity_to_Total_Assets_Ratio                        0
Total_Liabilities/total_assets                      0
Financial_Leverage                                  0
Interest_to_Sales                                   0
Cash_to_assets                                      0
(receivables*365)/sales                             0
total_costs/total_sales                             0
Cash_conversion_cycle                               0
(short-term_liabilities*365)/cost_of_products_sold)    0
class                                               0
dtype: int64
```

```
Bankrupt_modelMMM.describe()
```

	year	net_profit/total assets	net_profit/sales	Return_On_equity	Operating_Income
count	22828.000	22828.000	22828.000	22828.000	
mean	2.917	0.072	0.040	0.152	
std	1.302	0.126	0.079	0.286	
min	1.000	-0.618	-0.376	-1.338	
25%	2.000	0.015	0.010	0.036	
50%	3.000	0.059	0.034	0.129	
75%	4.000	0.127	0.074	0.268	
max	5.000	0.755	0.457	1.645	

```
Bankrupt_modelRF.describe()
```

	year	net_profit/total assets	net_profit/sales	Return_On_equity	Operating_Income
count	22828.000	22828.000	22828.000	22828.000	
mean	2.917	0.072	0.040	0.152	
std	1.302	0.126	0.079	0.286	
min	1.000	-0.618	-0.376	-1.338	
25%	2.000	0.015	0.009	0.035	
50%	3.000	0.059	0.034	0.129	
75%	4.000	0.127	0.074	0.268	
max	5.000	0.755	0.457	1.645	

```
Bankrupt_modelRF.isnull().sum()
```

```
year                                                0
net_profit/total assets                             0
net_profit/sales                                    0
Return_On_equity                                    0
Operating_Income_to_Total_Asset_ratio               0
(inventory*365)/cost_of_products_sold               0
working_capital                                     0
Working_Capital_to_Total_Assets_Ratio               0
working_capital_indicator                           0
Defensive_Interval_Ratio                            0
Equity_to_Total_Assets_Ratio                        0
Total_Liabilities/total_assets                      0
```

```
Financial_Leverage                                  0
Interest_to_Sales                                   0
Cash_to_assets                                      0
(receivables*365)/sales                             0
total_costs/total_sales                             0
Cash_conversion_cycle                               0
(short-term_liabilities*365)/cost_of_products_sold  0
class                                               0
dtype: int64
```

```
Bankrupt_modelRF['class'].value_counts()
```

```
class
0    21703
1     1125
Name: count, dtype: int64
```

Preparing to model

This research will be looking at the following alograthims:

1.Logistic Regression
2.Decision Tree
3.Random Forest
4.AdaBoost
5.Gradient Boosting

```
#performing K-Fold cross-validation to split training sets. Using K= 10 as standard
Bankrupt_model_x =Bankrupt_modelRF.drop(['class'],axis=1)
Bankrupt_model_y =Bankrupt_modelRF['class']
```

```
#defining the number of folds
from sklearn.model_selection import KFold
kf = KFold(n_splits=10, shuffle =True, random_state=7)
```

```
for train_index, test_index in kf.split(Bankrupt_model_x):
    # Split data into train and test sets
    X_train, X_test = Bankrupt_model_x.iloc[train_index], Bankrupt_model_x.iloc[test_index]
    y_train, y_test = Bankrupt_model_y.iloc[train_index], Bankrupt_model_y.iloc[test_index]
```

```
y_train.value_counts()
```

```
class
0    19537
1     1009
Name: count, dtype: int64
```

```
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
```

```
# Initialize models
models = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'AdaBoost': AdaBoostClassifier(),
    'Gradient Boosting': GradientBoostingClassifier()
}
```

```
#Logistic Regression
classifierLR = LogisticRegression()
classifierLR.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

```
y_predLR =pd.Series(classifierLR.predict(X_test))
y_predLR
y_predLR.value_counts()
```

```
0    2282
Name: count, dtype: int64
```

```
#Decision tree :

classifierDT =  DecisionTreeClassifier(random_state=7)
classifierDT.fit(X_train, y_train)
```

```
DecisionTreeClassifier(random_state=7)
```

```
y_predDT = pd.Series(classifierDT.predict(X_test))
y_predDT
y_predDT.value_counts()
```

```
0    2155
1     127
Name: count, dtype: int64
```

```
#Random Forest
classifierRF = RandomForestClassifier(random_state=7)
classifierRF.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=7)
```

```
y_predRF = pd.Series(classifierRF.predict(X_test))
y_predRF
y_predRF.value_counts()
```

```
0    2259
1      23
Name: count, dtype: int64
```

```
#AdaBoost
classifierAB = AdaBoostClassifier(random_state=7)
classifierAB.fit(X_train, y_train)
```

```
AdaBoostClassifier(random_state=7)
```

```
y_predAB = pd.Series(classifierAB.predict(X_test))
y_predAB
y_predAB.value_counts()
```

```
0    2267
1      15
Name: count, dtype: int64
```

```
#Gradient Boosting
classifierGB = GradientBoostingClassifier(random_state=7)
classifierGB.fit(X_train, y_train)
```

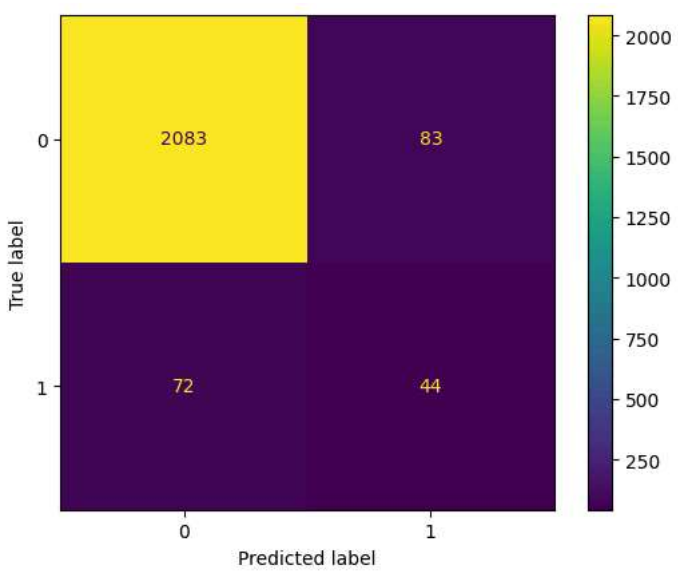
```
GradientBoostingClassifier
GradientBoostingClassifier(random_state=7)
```

```
y_predGB = pd.Series(classifierGB.predict(X_test))
y_predGB
y_predGB.value_counts()
```

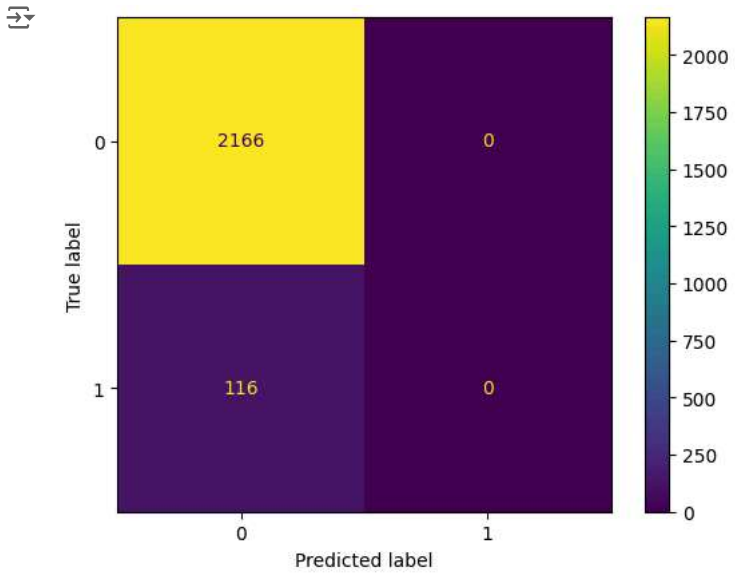
```
0    2259
1      23
Name: count, dtype: int64
```

Performance Metrics

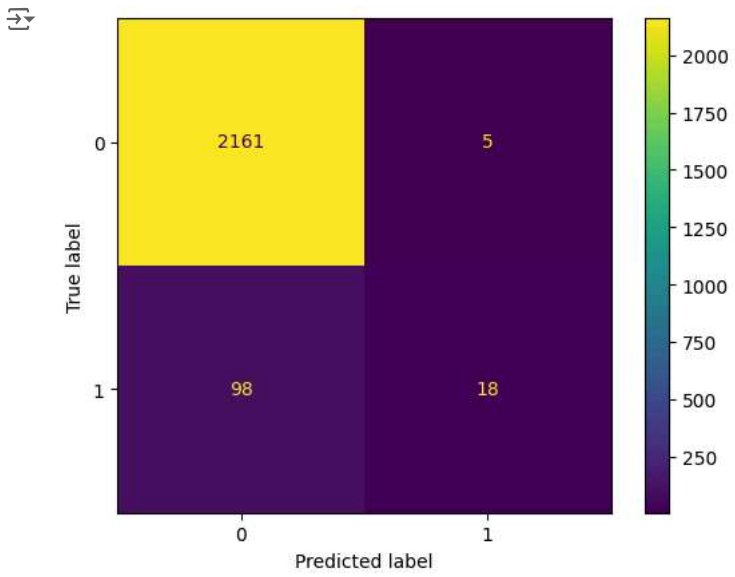
```
#Confusion matrix for Decision Tree
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm= confusion_matrix(y_test,y_predDT, labels=classifierDT.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classifierDT.classes_)
disp.plot()
plt.show()
```



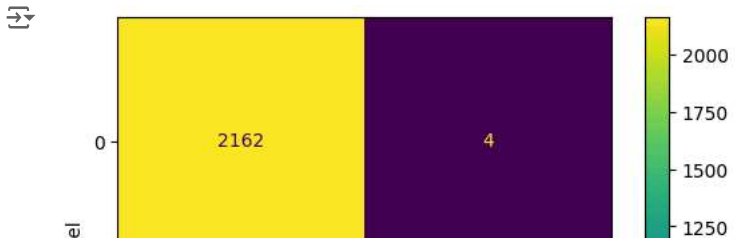

```
#Confusion matrix for Logistic Regression
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm= confusion_matrix(y_test,y_predLR, labels=classifierLR.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classifierLR.classes_)
disp.plot()
plt.show()
```




```
#Confusion matrix for Random Forest
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm= confusion_matrix(y_test,y_predRF, labels=classifierRF.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classifierRF.classes_)
disp.plot()
plt.show()
```




```
#Confusion matrix for AdaBoost
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm= confusion_matrix(y_test,y_predAB, labels=classifierAB.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classifierAB.classes_)
disp.plot()
plt.show()
```




```
#Confusion matrix for Gradient Boost
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm= confusion_matrix(y_test,y_predGB, labels=classifierGB.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classifierGB.classes_)
```