Model Developed by
Name: **Umair Ali**
Contact: **+923480233673**
Email: **uape00@gmail.com**
GitHub: **https://github.com/1umairali/models**

This project focused on identifying companies at risk of bankruptcy by analyzing financial data with machine learning classifiers. The process involved preparing and transforming the dataset, selecting meaningful features, and applying multiple classifiers such as Support Vector Classifier, Logist Regression, KNN, Naives Bayes Classifiers, Decision Tree Classifier, and Random Forest Classifier. By comparing model performance through metrics like accuracy, recall, and precision, the most reliable prediction method was selected. The outcome reflects practical experience in financial data analysis and risk prediction using classical machine learning methods.

In [1]:
```
# import libraries, others will be imported below
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# import dataframe
df = pd.read_csv(r'https://raw.githubusercontent.com/1umairali/models/main/company_bankruptcy_prediction/companies_dataset.csv
df
```

Out[2]:

	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-tax net Interest Rate	Non-industry income and expenditure/revenue	...	Liabi As
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887	0.808809	0.302646	...	
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380	0.809301	0.303556	...	
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403	0.808388	0.302035	...	
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967	0.808966	0.303350	...	
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366	0.809304	0.303475	...	
...	...	...	...	...	...	...	...	...	...	...	...	
6814	0	0.493687	0.539468	0.543230	0.604455	0.604462	0.998992	0.797409	0.809331	0.303510	...	
6815	0	0.475162	0.538269	0.524172	0.598308	0.598308	0.998992	0.797414	0.809327	0.303520	...	
6816	0	0.472725	0.533744	0.520638	0.610444	0.610213	0.998984	0.797401	0.809317	0.303512	...	
6817	0	0.506264	0.559911	0.554045	0.607850	0.607850	0.999074	0.797500	0.809399	0.303498	...	
6818	0	0.493053	0.570105	0.549548	0.627409	0.627409	0.998080	0.801987	0.813800	0.313415	...	

6819 rows × 95 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6819 entries, 0 to 6818
Data columns (total 95 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   Equity to Liability                                     6819 non-null   float64
dtypes: float64(93), int64(2)
memory usage: 4.9 MB
```

Note: see columns names (from 1 to 95), their is one button space before names see first column name (Bankrupt?) and others.

In [4]:

```python
# check null values in each columns
df.isnull().sum()
#df.isnull().sum().sum()
```

Out[4]:

```
Bankrupt?                                                  0
 ROA(C) before interest and depreciation before interest   0
 ROA(A) before interest and % after tax                    0
 ROA(B) before interest and depreciation after tax         0
 Operating Gross Margin                                    0
                                                          ..
 Net Income to Stockholder's Equity                        0
 Liability to Equity                                       0
 Degree of Financial Leverage (DFL)                        0
 Interest Coverage Ratio (Interest expense to EBIT)        0
 Equity to Liability                                       0
Length: 95, dtype: int64
```

In [5]:
```python
# see all columns
pd.set_option('display.max_columns', None)

# see all rows
#pd.set_option('display.max_rows', None)

df.describe()
```

Out[5]:

	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-tax net Interest Rate	Nor in expenditur
count	6819.000000	6819.000000	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	0.797190	0.809084	
std	0.176710	0.060686	0.065620	0.061595	0.016934	0.016916	0.013010	0.012869	0.013601	
min	0.000000	0.000000	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	0.797386	0.809312	
50%	0.000000	0.502706	0.559802	0.552278	0.605997	0.605976	0.999022	0.797464	0.809375	
75%	0.000000	0.535563	0.589157	0.584105	0.613914	0.613842	0.999095	0.797579	0.809469	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

Data Visualization

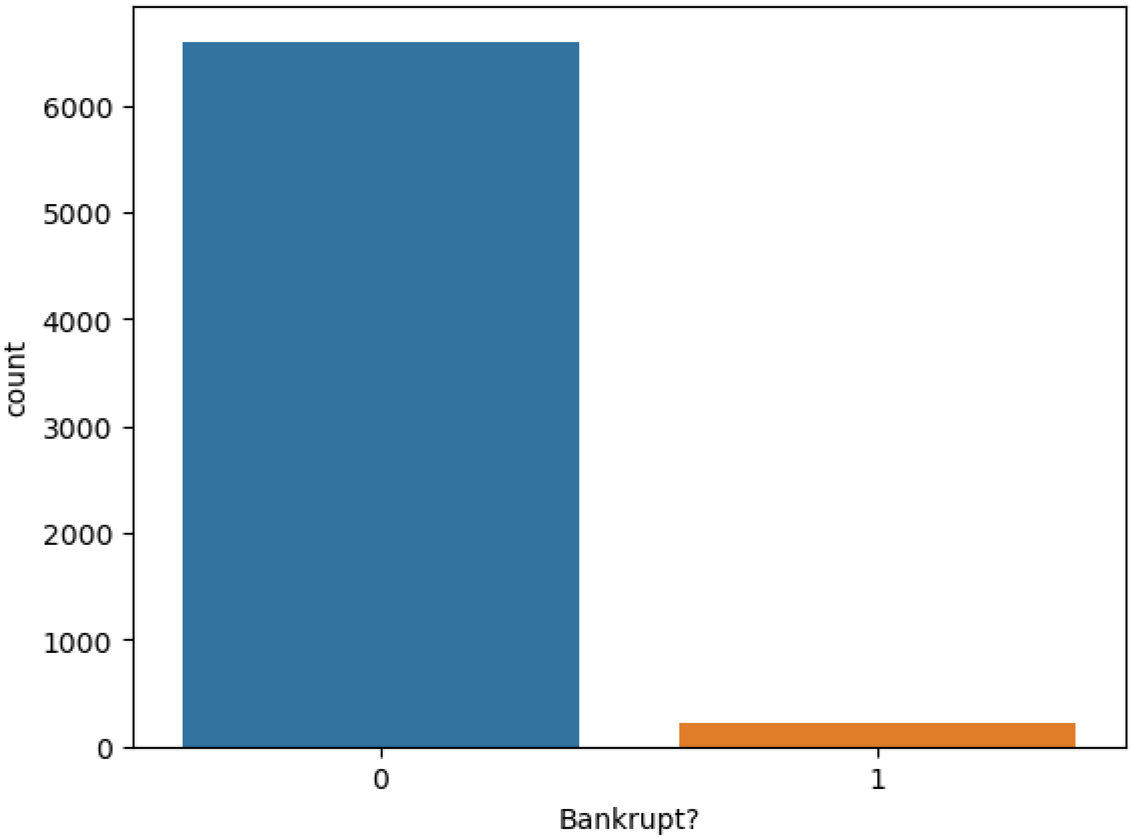
Note: see pairplot of bankrupt dataframe remove **#** from **plt** and **sns** code

i have no fastest GPU, and its take too much time.

In [6]:
```
# Pairplot of Bankrupt dataframe
#plt.figure(figsize=(60,60))
#sns.pairplot(df, hue = 'Bankrupt?')
```

In [7]:
```
# bankrupt countplot
sns.countplot(x=df["Bankrupt?"])
```

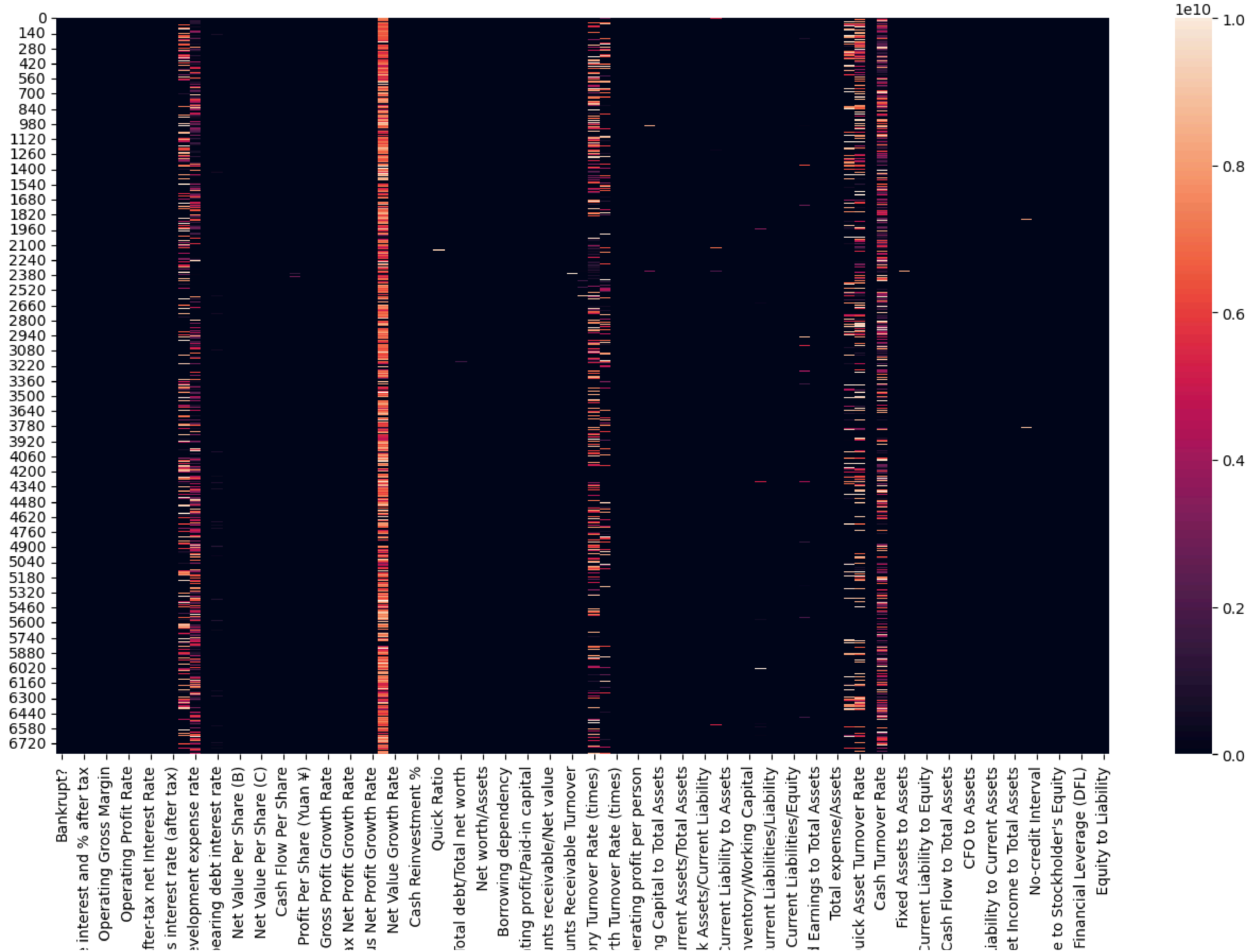
Out[7]:
```
<Axes: xlabel='Bankrupt?', ylabel='count'>
```



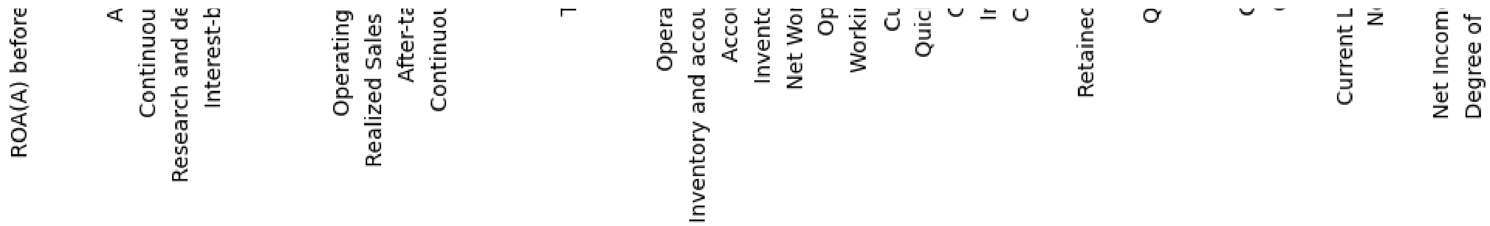
Heatmap

In [8]:
```
# heatmap of DataFrame
plt.figure(figsize=(16,9))
sns.heatmap(df)
```

Out[8]:
```
<Axes: >
```



1e10
1.0
0.8
0.6
0.4
0.2
0.0
Bankrupt?
Operating Gross Margin
Operating Profit Rate
Net Value Per Share (B)
Net Value Per Share (C)
Cash Flow Per Share
Profit Per Share (Yuan ¥)
Gross Profit Growth Rate
Net Value Growth Rate
Cash Reinvestment %
Quick Ratio
Net worth/Assets
Borrowing dependency
Cash Turnover Rate
Fixed Assets to Assets
Cash Flow to Total Assets
CFO to Assets
No-credit Interval
Financial Leverage (DFL)
Equity to Liability



Heatmap of Corelation Matrix

In [9]:

```
# Corelation Matrix
df.corr()
```

Out[9]:

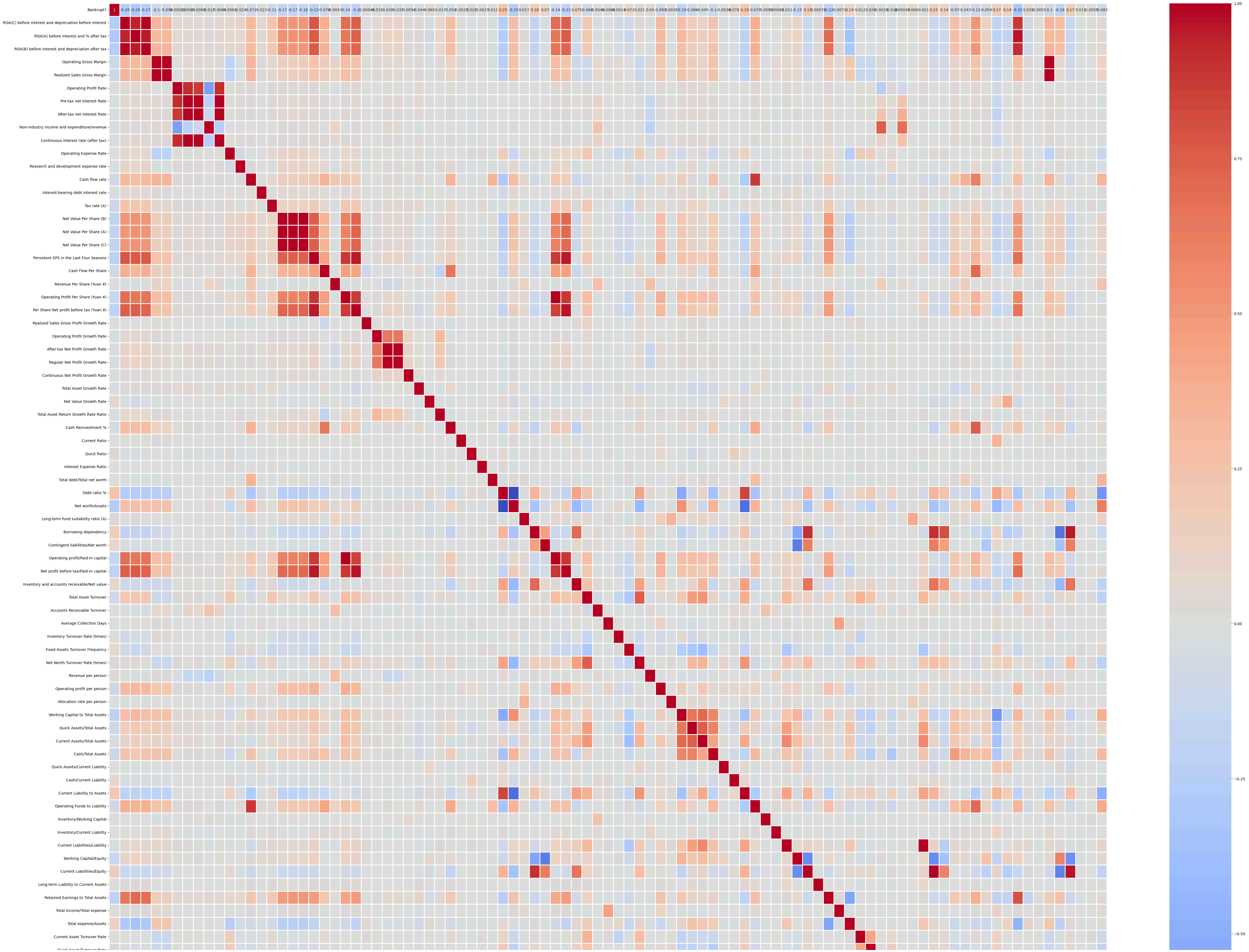
	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industr income an expenditure/revenu
Bankrupt?	1.000000	-0.260807	-0.282941	-0.273051	-0.100043	-0.099445	-0.000230	-0.008517	-0.008857	-0.01659
ROA(C) before interest and depreciation before interest	-0.260807	1.000000	0.940124	0.986849	0.334719	0.332755	0.035725	0.053419	0.049222	0.02050
ROA(A) before interest and % after tax	-0.282941	0.940124	1.000000	0.955741	0.326969	0.324956	0.032053	0.053518	0.049474	0.02964
ROA(B) before interest and depreciation after tax	-0.273051	0.986849	0.955741	1.000000	0.333749	0.331755	0.035212	0.053726	0.049952	0.02236
Operating Gross Margin	-0.100043	0.334719	0.326969	0.333749	1.000000	0.999518	0.005745	0.032493	0.027175	0.05143
...	...	...	...	...	...	...	...	...	...	
Net Income to Stockholder's Equity	-0.180987	0.274287	0.291744	0.280617	0.075304	0.074891	0.006216	0.011343	0.010648	0.00769
Liability to Equity	0.166812	-0.143629	-0.141039	-0.142838	-0.085434	-0.085407	0.001541	-0.004043	-0.004390	-0.01189

	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-tax net Interest Rate	Non-industr income an expenditure/revenu
Degree of Financial Leverage (DFL)	0.010508	-0.016575	-0.011515	-0.014663	-0.011806	-0.011268	0.000935	0.000855	0.000927	-0.00055
Interest Coverage Ratio (Interest expense to EBIT)	-0.005509	0.010573	0.013372	0.011473	-0.001167	-0.001158	0.000393	0.000984	0.000957	0.00102
Equity to Liability	-0.083048	0.052416	0.057887	0.056430	0.120029	0.120196	-0.017071	-0.014559	-0.010900	0.01229

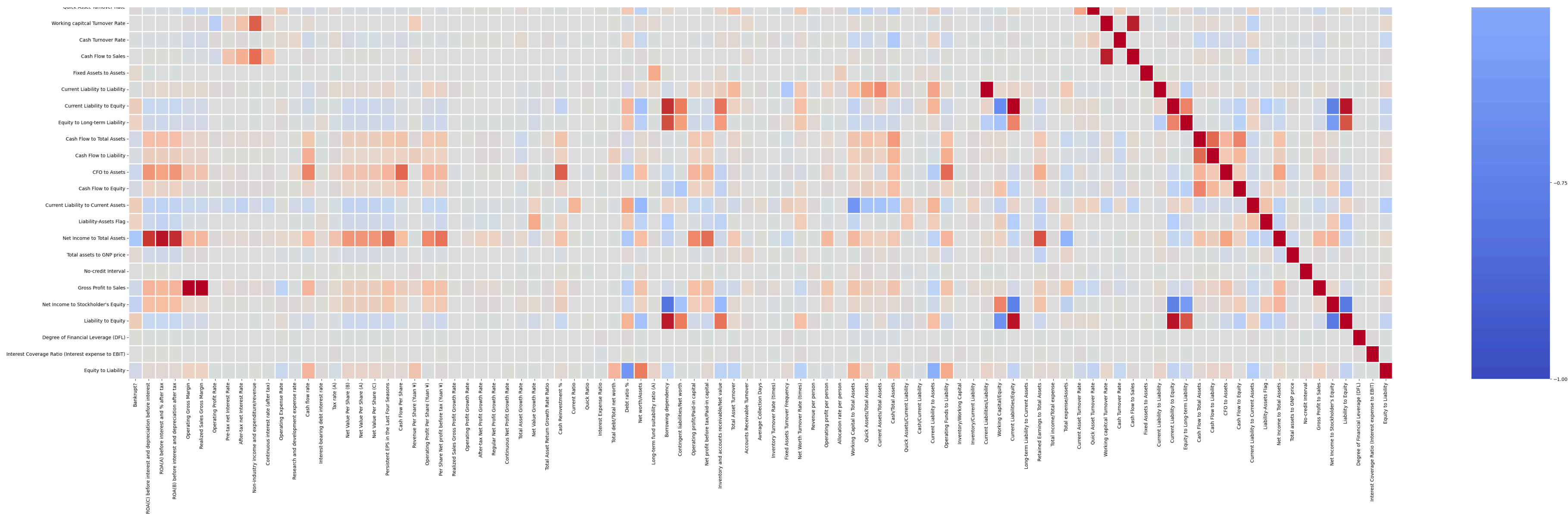
95 rows × 95 columns

In [10]:
```
# Heatmap of Correlation matrix
plt.figure(figsize=(60,60))
sns.heatmap(df.corr(), annot = True, cmap ='coolwarm', linewidths=2)
```

Out[10]: <Axes: >



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-tax net Interest Rate
Non-industry income and expenditure/revenue
Continuous interest rate (after tax)
Operating Expense Rate
Research and development expense rate
Cash flow rate
Interest-bearing debt interest rate
Tax rate (A)
Net Value Per Share (B)
Net Value Per Share (A)
Net Value Per Share (C)
Persistent EPS in the Last Four Seasons
Cash Flow Per Share
Revenue Per Share (Yuan ¥)
Operating Profit Per Share (Yuan ¥)
Per Share Net profit before tax (Yuan ¥)
Realized Sales Gross Profit Growth Rate
Operating Profit Growth Rate
After-tax Net Profit Growth Rate
Regular Net Profit Growth Rate
Continuous Net Profit Growth Rate
Total Asset Growth Rate
Net Value Growth Rate
Total Asset Return Growth Rate Ratio
Cash Reinvestment %
Current Ratio
Quick Ratio
Interest Expense Ratio
Total debt/Total net worth
Debt ratio %
Net worth/Assets
Long-term fund suitability ratio (A)
Borrowing dependency
Contingent liabilities/Net worth
Operating profit/Paid-in capital
Net profit before tax/Paid-in capital
Inventory and accounts receivable/Net value
Total Asset Turnover
Accounts Receivable Turnover
Average Collection Days
Inventory Turnover Rate (times)
Fixed Assets Turnover Frequency
Net Worth Turnover Rate (times)
Revenue per person
Operating profit per person
Allocation rate per person
Working Capital to Total Assets
Quick Assets/Total Assets
Current Assets/Total Assets
Cash/Total Assets
Quick Assets/Current Liability
Cash/Current Liability
Current Liability to Assets
Operating Funds to Liability
Inventory/Working Capital
Inventory/Current Liability
Current Liabilities/Liability
Working Capital/Equity
Current Liabilities/Equity
Long-term Liability to Current Assets
Retained Earnings to Total Assets
Total income/Total expense
Total expense/Assets
Current Asset Turnover Rate
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50



Split Data into Train and Test

In [11]:
```
# drop dependent (Bankrupt) column, it will assign to y
X = df.drop(['Bankrupt?'], axis = 1)
X.head(6)
```

Out[11]:

	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-tax net Interest Rate	Non-industry income and expenditure/revenue	Continuous interest rate (after tax)	Operati Expense Ra
0	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887	0.808809	0.302646	0.780985	1.256970e-
1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380	0.809301	0.303556	0.781506	2.897850e-
2	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403	0.808388	0.302035	0.780284	2.361300e-
3	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967	0.808966	0.303350	0.781241	1.078890e-
4	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366	0.809304	0.303475	0.781550	7.890000e+
5	0.388680	0.415177	0.419134	0.590171	0.590251	0.998758	0.796903	0.808771	0.303116	0.781069	1.571500e-

In [12]:

```
# assign only 'Bankrupt' column to y
y = df['Bankrupt?']
y.head(6)
```

Out[12]:

```
0    1
1    1
2    1
3    1
4    1
5    1
Name: Bankrupt?, dtype: int64
```

In [13]:

```
# split dataset into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state= 5)
```

In [14]:

```
X_train
```

Out[14]:

	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-tax net Interest Rate	Non-industry income and expenditure/revenue	Continuous interest rate (after tax)	Ope Expens
1022	0.580169	0.629961	0.611596	0.610271	0.610271	0.999076	0.797786	0.809593	0.303994	0.781873	1.6073
890	0.578657	0.629252	0.614969	0.614062	0.614011	0.999137	0.797629	0.809489	0.303592	0.781756	1.5981
957	0.338956	0.428314	0.381980	0.605464	0.605421	0.998973	0.795970	0.808022	0.301036	0.779747	1.8525
5551	0.515039	0.577137	0.562450	0.605320	0.605320	0.999091	0.797493	0.809393	0.303448	0.781660	6.53000
2954	0.554770	0.600578	0.594197	0.606156	0.606156	0.999083	0.797584	0.809459	0.303625	0.781729	8.68000
...	...	...	...	...	...	...	...	...	...	...	
3046	0.451519	0.511284	0.497939	0.619056	0.619056	0.998836	0.797296	0.809206	0.303640	0.781456	5.4187
1725	0.503632	0.565362	0.548852	0.645822	0.645822	0.999045	0.797486	0.809385	0.303535	0.781639	7.6169
4079	0.511724	0.592510	0.560415	0.601493	0.601500	0.999006	0.797455	0.809368	0.303561	0.781613	8.93000
2254	0.462390	0.532054	0.513732	0.603915	0.603915	0.999024	0.797395	0.809307	0.303418	0.781545	1.1008
2915	0.484132	0.556204	0.538519	0.583584	0.583584	0.997241	0.805640	0.817080	0.321553	0.789747	1.5682

5455 rows × 94 columns

In [15]:

```
X_test
```

Out[15]:

	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-tax net Interest Rate	Non-industry income and expenditure/revenue	Continuous interest rate (after tax)	Ope Expens
4214	0.542924	0.605266	0.594143	0.603266	0.603317	0.999017	0.797485	0.809395	0.303591	0.781662	1.0765
4797	0.520304	0.582752	0.568499	0.609356	0.609356	0.999146	0.797625	0.809500	0.303565	0.781732	7.50000
2458	0.420368	0.471544	0.462766	0.582431	0.582431	0.998163	0.796295	0.808286	0.303297	0.780464	6.2686
4395	0.605518	0.669701	0.648054	0.605486	0.605479	0.999075	0.797645	0.809519	0.303749	0.781789	8.45000
4955	0.652367	0.721871	0.701536	0.641945	0.641260	0.999496	0.798032	0.809858	0.303544	0.782157	2.3682
...	...	...	...	...	...	...	...	...	...	...	
4140	0.519670	0.586840	0.577226	0.609659	0.609659	0.999031	0.797458	0.809380	0.303514	0.781649	1.9496
4696	0.600741	0.673626	0.670432	0.615525	0.614977	0.999207	0.798059	0.809964	0.304196	0.782286	1.0442
2988	0.612490	0.636557	0.646287	0.612714	0.612714	0.999179	0.797728	0.809575	0.303676	0.781852	9.61000
3349	0.481792	0.532926	0.535949	0.621961	0.621961	0.998953	0.797388	0.809314	0.303554	0.781570	4.7040
2716	0.562229	0.607065	0.602388	0.618040	0.618040	0.999151	0.797678	0.809523	0.303648	0.781794	2.1022

1364 rows × 94 columns

In [16]:

```
y_train
```

Out[16]:
```
1022    0
890     0
957     0
5551    0
2954    0
       ..
3046    0
1725    0
4079    0
2254    0
2915    0
Name: Bankrupt?, Length: 5455, dtype: int64
```

In [17]:
```
y_test
```

Out[17]:
```
4214    0
4797    0
2458    0
4395    0
4955    0
       ..
4140    0
4696    0
2988    0
3349    0
2716    0
Name: Bankrupt?, Length: 1364, dtype: int64
```

Standard Scaling

In [18]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_sc = sc.fit_transform(X_train)
X_test_sc = sc.transform(X_test)
```

ML Model Building

In [19]:
```
# import for confusion_matrix, classification_report, accuracy_score
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

Support Vector Classifier

In [20]:
```
# Train with normal train & test data
from sklearn.svm import SVC
svc_classifier = SVC()
svc_classifier.fit(X_train, y_train)
y_pred_scv = svc_classifier.predict(X_test)
accuracy_score(y_test, y_pred_scv)
```

Out[20]:
```
0.966275659824047
```

In [21]:
```
# Train with standard scalled train & test data
from sklearn.svm import SVC
svc_classifier2 = SVC()
svc_classifier2.fit(X_train_sc, y_train)
y_pred_svc_sc = svc_classifier2.predict(X_test_sc)
accuracy_score(y_test, y_pred_svc_sc)
```

Out[21]:
```
0.9655425219941349
```

Logistic Regression

In [22]:
```
# Train with normal train & test data
from sklearn.linear_model import LogisticRegression
lr_classifier = LogisticRegression()
lr_classifier.fit(X_train, y_train)
y_pred_lr = lr_classifier.predict(X_test)
accuracy_score(y_test, y_pred_lr)
```

```
C:\Users\Umair Ali\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to conve
rge (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(
```

Out[22]: 0.9611436950146628

In [23]:

```python
# Train with standard scalled train & test data
from sklearn.linear_model import LogisticRegression
lr_classifier2 = LogisticRegression()
lr_classifier2.fit(X_train_sc, y_train)
y_pred_lr_sc = lr_classifier2.predict(X_test_sc)
accuracy_score(y_test, y_pred_lr_sc)
```

```
C:\Users\Umair Ali\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to conve
rge (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(
```

Out[23]: 0.9633431085043989

KNN - K-Nearesr Neighbor Classifier

In [24]:

```python
# train with normal test and train data
from sklearn.neighbors import KNeighborsClassifier
knn_classifier = KNeighborsClassifier()
knn_classifier.fit(X_train, y_train)
y_pred_knn = knn_classifier.predict(X_test)
accuracy_score(y_test, y_pred_knn)
```

Out[24]: 0.967008797653959

In [25]:
```
# Train with standard scaled Data
knn_classifier2 = KNeighborsClassifier()
knn_classifier2.fit(X_train_sc, y_train)
y_pred_knn_sc = knn_classifier2.predict(X_test_sc)
accuracy_score(y_test, y_pred_knn_sc)
```

Out[25]: 0.966275659824047

Naive Bayes Classifier

In [26]:
```
# Naive Bayes Classifier
from sklearn.naive_bayes import GaussianNB
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
y_pred_nb = nb_classifier.predict(X_test)
accuracy_score(y_test, y_pred_nb)
```

Out[26]: 0.05791788856304985

In [27]:
```
# Train with Standard scaled Data
nb_classifier2 = GaussianNB()
nb_classifier2.fit(X_train_sc, y_train)
y_pred_nb_sc = nb_classifier2.predict(X_test_sc)
accuracy_score(y_test, y_pred_nb_sc)
```

Out[27]: 0.6906158357771262

Decision Tree Classifier

In [28]:
```
# Train with normal test and train data
from sklearn.tree import DecisionTreeClassifier
dt_classifier = DecisionTreeClassifier()
dt_classifier.fit(X_train, y_train)
```

```
y_pred_dt = dt_classifier.predict(X_test)
accuracy_score(y_test, y_pred_dt)
```

Out[28]: 0.9420821114369502

In [29]:
```
# Train with Standard scaled Data
dt_classifier2 = DecisionTreeClassifier()
dt_classifier2.fit(X_train_sc, y_train)
y_pred_dt_sc = dt_classifier2.predict(X_test_sc)
accuracy_score(y_test, y_pred_dt_sc)
```

Out[29]: 0.9442815249266863

Random Forest Classifier

In [30]:
```
# train with normal data
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, y_train)
y_pred_rf = rf_classifier.predict(X_test)
accuracy_score(y_test, y_pred_rf)
```

Out[30]: 0.967741935483871

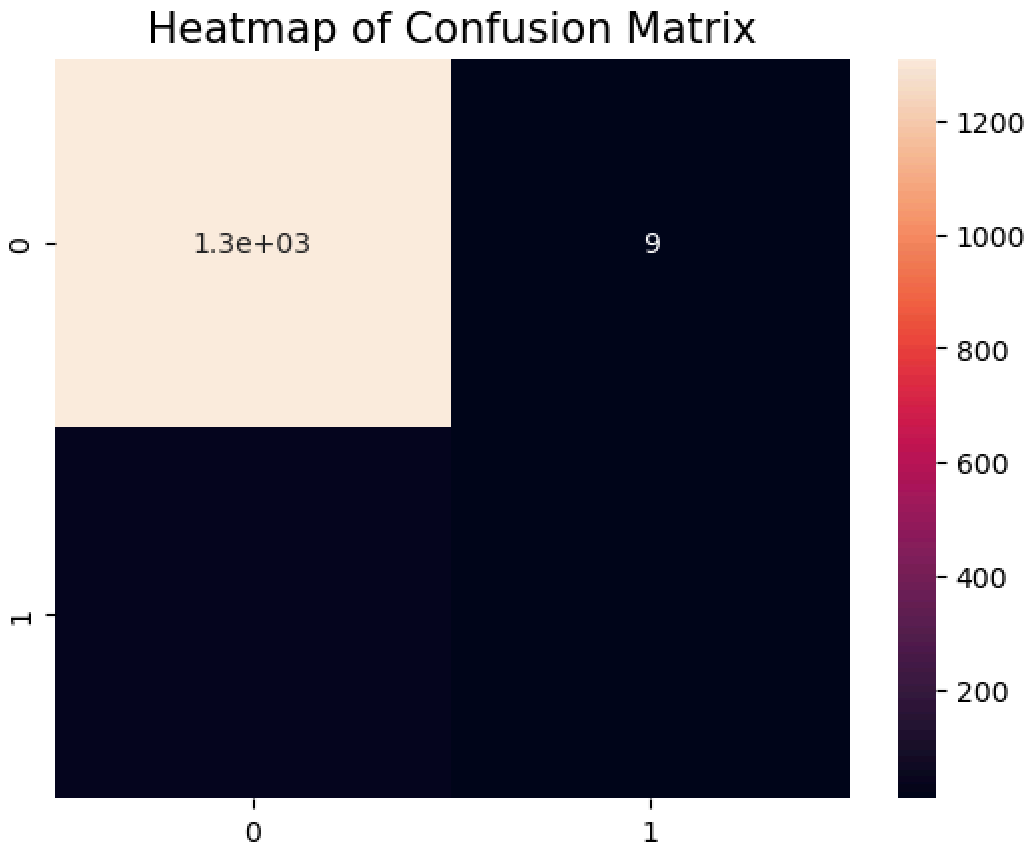
In [31]:
```
# Train with Standard scaled Data
rf_classifier2 = RandomForestClassifier()
rf_classifier2.fit(X_train_sc, y_train)
y_pred_rf_sc = rf_classifier2.predict(X_test_sc)
accuracy_score(y_test, y_pred_rf_sc)
```

Out[31]: 0.967008797653959

Confusion Matrix

In [32]:
```
cm = confusion_matrix(y_test, y_pred_knn_sc)
plt.title('Heatmap of Confusion Matrix', fontsize = 15)
```

```python
sns.heatmap(cm, annot = True)
plt.show()
```



Classification Report Of model

In [33]:
```python
print(classification_report(y_test, y_pred_knn_sc))
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1318
           1       0.50      0.20      0.28        46

    accuracy                           0.97      1364
   macro avg       0.74      0.59      0.63      1364
weighted avg       0.96      0.97      0.96      1364
```

Cross-validation of the ML model

In [34]:
```
# Cross validation
from sklearn.model_selection import cross_val_score
cross_validation = cross_val_score(estimator = knn_classifier2, X = X_train, y = y_train, cv = 10)
print("Cross validation accuracy of XGBoost model = ", cross_validation)
print("\nCross validation mean accuracy of XGBoost model = ", cross_validation.mean())
```

```
Cross validation accuracy of XGBoost model =  [0.96886447 0.96520147 0.96520147 0.96520147 0.96703297 0.96880734
 0.96880734 0.96880734 0.96880734 0.96880734]

Cross validation mean accuracy of XGBoost model =  0.9675538528749538
```

Test Model

In [35]:
```
# have one bank_detail data in list
bank_detail = [0.703325, 1.649695, 0.538953, 0.608541, 0.687929, 0.909132, 1.097677, 0.895028, 0.430365, 0.681804,
              6.7300100, 3.87000, 1.465031, 0.08019, 0.41237, 0.212043, 1.012043, 0.812043, 0.293845, 0.332055, 0.174338,
              0.144711, 1.249529, 0.072118, 0.848445, 0.650239, 0.68062, 0.214618, 8.850100, 1.000603, 0.464632, 0.390047,
        0.017554, 0.105742, 0.730689, 0.012266, 1.056665, 0.243335, 0.005314, 1.0076, 0.095366, 1.054607, 0.848616,
        0.207879, 0.181906, 1.000668, 0.809473, 0.008183, 0.008443, 0.156613, 1.015146, 0.480602, 0.045419, 0.751738,
        0.459171, 0.588765, 0.118948, 0.906029, 0.088745, 0.127809, 0.849635, 0.47749, 0.704292, 0.118056, 0.784651, 0.383724,
        0.08199, 0.451636, 0.00884, 0.017853, 0.080123, 0.001105, 0.793934, 10300000.0, 0.571556, 0.248902, 0.118056, 0.353724
        0.118113, 0.410111, 0.453128, 0.71008, 0.358077,0.138366, 1, 0.156641, 1.002147, 0.123729,1.608543,0.841003,0.882225,
        0.08681, 0, 0.023005]
```

In [36]:
```
# convert list into array and scale
bank_detail_sc = sc.transform(np.array([bank_detail]))
```

```
bank_detail_sc
```

```
C:\Users\Umair Ali\anaconda3\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but Stand
ardScaler was fitted with feature names
  warnings.warn(
```

Out[36]:

```
array([[ 3.28810929e+00,  1.66831546e+01, -2.32684771e-01,
         3.74080455e-02,  4.50922915e+00, -6.60566420e+00,
         2.60918901e+01,  8.14606511e+00,  1.07549678e+01,
        -8.95180778e+00, -6.18184085e-01, -7.51130065e-01,
         6.14664719e+01, -1.56978275e-01,  2.15795740e+00,
         6.30683532e-01,  2.39659102e+01,  1.81257640e+01,
         1.89899670e+00,  4.65674625e-01, -2.87334682e-02,
         1.22884211e+00,  3.09375648e+01,  3.69128851e+00,
         4.03832783e-02, -2.78373183e+00, -6.07620560e-01,
        -2.69175201e-01, -1.88966898e+00, -1.53406492e-02,
         1.87607485e+01,  5.00388186e-01, -1.35407410e-02,
        -3.66813940e-02,  8.34701166e+00, -2.80818877e-02,
         1.74174368e+01, -1.18791261e+01, -1.20203720e-01,
         3.83323054e+01,  6.56620609e+00,  3.27457347e+01,
         2.08853583e+01, -1.41401050e+01,  3.83219577e-01,
        -4.98835590e-02, -3.80040535e-02, -6.56374034e-01,
        -4.01857722e-01,  3.08585436e+00, -1.90343649e-02,
         2.54152385e+00, -4.05946934e-02, -1.05550256e+00,
         2.90760498e-01,  3.00290045e-01, -3.79690384e-02,
        -2.34095857e-02, -6.92740823e-02,  7.31179008e-01,
         1.44281078e+01,  1.71439636e+01, -9.50519046e-02,
        -3.10406851e+00,  3.95900191e+00,  4.00421504e+00,
        -9.44665714e-02, -1.94744891e+01,  4.63223555e-01,
        -4.14449466e-01, -4.31372189e-01, -6.44890360e-01,
         3.34788204e+01, -8.36625585e-01, -1.94867342e+01,
        -1.35407411e-02, -3.10406851e+00,  1.71168333e+00,
         1.11650987e-01, -4.99310473e+00, -2.87686448e-01,
         1.98416253e+00,  4.03618763e+00,  3.55657319e+00,
         2.60935816e+01, -1.61622573e+01, -4.85781038e-02,
        -3.97676140e+01,  5.62886461e+01,  3.66057934e-02,
         4.06492851e+01,  5.32346730e+00, -3.89290187e+01,
        -4.79388119e-01]])
```

In [37]:

```
# predict bankruptency result scale data
predict = knn_classifier2.predict(bank_detail_sc)
```

```
predict
```

Out[37]:
```
array([0], dtype=int64)
```

In [38]:
```
# write if else statement to print result in clear format
if predict[0] == 0:
    print ('NOT Bankrupted')
else:
    print ("Bankrupted")
```

```
NOT Bankrupted
```

In [39]:
```
# confusion matrix
print('Confusion matrix of KNN SC model: \n',confusion_matrix(y_test, y_pred_knn_sc),'\n')

# show the accuracy
print('Accuracy of KNN SC model = ',accuracy_score(y_test, y_pred_knn_sc))
```

```
Confusion matrix of KNN SC model: 
 [[1309    9]
 [  37    9]] 

Accuracy of KNN SC model =  0.966275659824047
```