Final Project DSC 540

Vismay Patel

EDA ANALYSIS AND IMPLEMENTATION OF DIFFERENT TECHNIQUES ON BANKRUPTCY DATASET

In [10]:
```
# IMPORTING essential libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the dataset as a csv file

In [11]:
```
bankrupt = pd.read_csv("D:/DEPAUL/DePaul 6th quaeter (Winter 2021 JAN)/DSC 540/PROJECT/data.csv")
bankrupt.head()
```

Out[11]:

	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	...	Net Income to Total Assets	Total assets to GNP price	No-credit Interval	Gross Profit to Sales	Net Stockl
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887	0.808809	0.302646	...	0.716845	0.009219	0.622879	0.601453	0
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380	0.809301	0.303556	...	0.795297	0.008323	0.623652	0.610237	0
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403	0.808388	0.302035	...	0.774670	0.040003	0.623841	0.601449	0
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967	0.808966	0.303350	...	0.739555	0.003252	0.622929	0.583538	0
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366	0.809304	0.303475	...	0.795016	0.003878	0.623521	0.598782	0

5 rows × 96 columns

In [12]:
```
bankrupt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6819 entries, 0 to 6818
Data columns (total 96 columns):
 #   Column                                                    Non-Null Count  Dtype
---  ------                                                    --------------  -----
 0   Bankrupt?                                                 6819 non-null   int64
 1    ROA(C) before interest and depreciation before interest  6819 non-null   float64
 2    ROA(A) before interest and % after tax                   6819 non-null   float64
 3    ROA(B) before interest and depreciation after tax        6819 non-null   float64
 4    Operating Gross Margin                                   6819 non-null   float64
 5    Realized Sales Gross Margin                              6819 non-null   float64
 6    Operating Profit Rate                                    6819 non-null   float64
 7    Pre-tax net Interest Rate                                6819 non-null   float64
 8    After-tax net Interest Rate                              6819 non-null   float64
```

9	Non-industry income and expenditure/revenue	6819 non-null	float64
10	Continuous interest rate (after tax)	6819 non-null	float64
11	Operating Expense Rate	6819 non-null	float64
12	Research and development expense rate	6819 non-null	float64
13	Cash flow rate	6819 non-null	float64
14	Interest-bearing debt interest rate	6819 non-null	float64
15	Tax rate (A)	6819 non-null	float64
16	Net Value Per Share (B)	6819 non-null	float64
17	Net Value Per Share (A)	6819 non-null	float64
18	Net Value Per Share (C)	6819 non-null	float64
19	Persistent EPS in the Last Four Seasons	6819 non-null	float64
20	Cash Flow Per Share	6819 non-null	float64
21	Revenue Per Share (Yuan ¥)	6819 non-null	float64
22	Operating Profit Per Share (Yuan ¥)	6819 non-null	float64
23	Per Share Net profit before tax (Yuan ¥)	6819 non-null	float64
24	Realized Sales Gross Profit Growth Rate	6819 non-null	float64
25	Operating Profit Growth Rate	6819 non-null	float64
26	After-tax Net Profit Growth Rate	6819 non-null	float64
27	Regular Net Profit Growth Rate	6819 non-null	float64
28	Continuous Net Profit Growth Rate	6819 non-null	float64
29	Total Asset Growth Rate	6819 non-null	float64
30	Net Value Growth Rate	6819 non-null	float64
31	Total Asset Return Growth Rate Ratio	6819 non-null	float64
32	Cash Reinvestment %	6819 non-null	float64
33	Current Ratio	6819 non-null	float64
34	Quick Ratio	6819 non-null	float64
35	Interest Expense Ratio	6819 non-null	float64
36	Total debt/Total net worth	6819 non-null	float64
37	Debt ratio %	6819 non-null	float64
38	Net worth/Assets	6819 non-null	float64
39	Long-term fund suitability ratio (A)	6819 non-null	float64
40	Borrowing dependency	6819 non-null	float64
41	Contingent liabilities/Net worth	6819 non-null	float64
42	Operating profit/Paid-in capital	6819 non-null	float64
43	Net profit before tax/Paid-in capital	6819 non-null	float64
44	Inventory and accounts receivable/Net value	6819 non-null	float64
45	Total Asset Turnover	6819 non-null	float64
46	Accounts Receivable Turnover	6819 non-null	float64
47	Average Collection Days	6819 non-null	float64
48	Inventory Turnover Rate (times)	6819 non-null	float64
49	Fixed Assets Turnover Frequency	6819 non-null	float64
50	Net Worth Turnover Rate (times)	6819 non-null	float64
51	Revenue per person	6819 non-null	float64
52	Operating profit per person	6819 non-null	float64
53	Allocation rate per person	6819 non-null	float64
54	Working Capital to Total Assets	6819 non-null	float64
55	Quick Assets/Total Assets	6819 non-null	float64
56	Current Assets/Total Assets	6819 non-null	float64
57	Cash/Total Assets	6819 non-null	float64
58	Quick Assets/Current Liability	6819 non-null	float64
59	Cash/Current Liability	6819 non-null	float64
60	Current Liability to Assets	6819 non-null	float64
61	Operating Funds to Liability	6819 non-null	float64
62	Inventory/Working Capital	6819 non-null	float64
63	Inventory/Current Liability	6819 non-null	float64
64	Current Liabilities/Liability	6819 non-null	float64
65	Working Capital/Equity	6819 non-null	float64
66	Current Liabilities/Equity	6819 non-null	float64
67	Long-term Liability to Current Assets	6819 non-null	float64
68	Retained Earnings to Total Assets	6819 non-null	float64

```
 69   Total income/Total expense                          6819 non-null   float64
 70   Total expense/Assets                                6819 non-null   float64
 71   Current Asset Turnover Rate                         6819 non-null   float64
 72   Quick Asset Turnover Rate                           6819 non-null   float64
 73   Working capitcal Turnover Rate                      6819 non-null   float64
 74   Cash Turnover Rate                                  6819 non-null   float64
 75   Cash Flow to Sales                                  6819 non-null   float64
 76   Fixed Assets to Assets                              6819 non-null   float64
 77   Current Liability to Liability                      6819 non-null   float64
 78   Current Liability to Equity                         6819 non-null   float64
 79   Equity to Long-term Liability                       6819 non-null   float64
 80   Cash Flow to Total Assets                           6819 non-null   float64
 81   Cash Flow to Liability                              6819 non-null   float64
 82   CFO to Assets                                       6819 non-null   float64
 83   Cash Flow to Equity                                 6819 non-null   float64
 84   Current Liability to Current Assets                 6819 non-null   float64
 85   Liability-Assets Flag                               6819 non-null   int64
 86   Net Income to Total Assets                          6819 non-null   float64
 87   Total assets to GNP price                           6819 non-null   float64
 88   No-credit Interval                                  6819 non-null   float64
 89   Gross Profit to Sales                               6819 non-null   float64
 90   Net Income to Stockholder's Equity                  6819 non-null   float64
 91   Liability to Equity                                 6819 non-null   float64
 92   Degree of Financial Leverage (DFL)                  6819 non-null   float64
 93   Interest Coverage Ratio (Interest expense to EBIT)  6819 non-null   float64
 94   Net Income Flag                                     6819 non-null   int64
 95   Equity to Liability                                 6819 non-null   float64
dtypes: float64(93), int64(3)
memory usage: 5.0 MB
```

No missing values in the dataset

In [13]:
```
bankrupt.describe()
```

Out[13]:

	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	...	Net Income to Total Assets	Total ass to GNP pr
count	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	...	6819.000000	6.819000e
mean	0.032263	0.505180	0.558625	0.553589	0.607948	0.607929	0.998755	0.797190	0.809084	0.303623	...	0.807760	1.862942e
std	0.176710	0.060686	0.065620	0.061595	0.016934	0.016916	0.013010	0.012869	0.013601	0.011163	...	0.040332	3.764501e
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000e
25%	0.000000	0.476527	0.535543	0.527277	0.600445	0.600434	0.998969	0.797386	0.809312	0.303466	...	0.796750	9.036205e
50%	0.000000	0.502706	0.559802	0.552278	0.605997	0.605976	0.999022	0.797464	0.809375	0.303525	...	0.810619	2.085213e
75%	0.000000	0.535563	0.589157	0.584105	0.613914	0.613842	0.999095	0.797579	0.809469	0.303585	...	0.826455	5.269777e
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	9.820000e

8 rows × 96 columns

Changing column names

In [14]:
```
bankrupt.columns
```

Out[14]:
```
Index(['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',
       ' Quick Asset Turnover Rate', ' Working capitcal Turnover Rate',
       ' Cash Turnover Rate', ' Cash Flow to Sales', ' Fixed Assets to Assets',
       ' Current Liability to Liability', ' Current Liability to Equity',
       ' Equity to Long-term Liability', ' Cash Flow to Total Assets',
       ' Cash Flow to Liability', ' CFO to Assets', ' Cash Flow to Equity',
       ' Current Liability to Current Assets', ' Liability-Assets Flag',
       ' Net Income to Total Assets', ' Total assets to GNP price',
       ' No-credit Interval', ' Gross Profit to Sales',
       ' Net Income to Stockholder's Equity', ' Liability to Equity',
       ' Degree of Financial Leverage (DFL)',
       ' Interest Coverage Ratio (Interest expense to EBIT)',
```

```
       ' Net Income Flag', ' Equity to Liability'],
      dtype='object')
```

Below are the new columns that will be assigned to the dataset

In [15]:

```python
bankrupt.columns = [
                    'Bankrupt',
                    'Return_on_total_assetC',
                    'Return_on_total_assetA',
                    'Return_on_total_assetB',
                    'GrossProfit/NetSales',
                    'Realized_GrossProfit/NetSales',
                    'Operating_Income/NetSales',
                    'Pre-TaxIncome/NetSales',
                    'NetIncome/NetSales',
                    'Net_Non-operating_Income_Ratio',
                    'Net_Income-Exclude_DisposalGain/Loss/Net_Sales',
                    'Operating_Expenses/Net Sales',
                    'Research_and_DevelopmentExpenses/Net Sales',
                    'CashFlow_from_Operating/Current Liabilities',
                    'Interest-bearing Debt/Equity',
                    'Effective Tax Rate',
                    'Net_Val_Per_ShareB',
                    'Net_Val_per_ShareA',
                    'Net_Val_Per_ShareC',
                    'EPS-Net_Income',
                    'Cash_Flow_Per_Share',
                    'Revenue_Per_Share(Yuan¥)',
                    'Operating_Income_Per_Share(Yuan¥)',
                    'Pretax_Income_Per_Share(Yuan¥)',
                    'Realized_Sales_Gross_Profit_Growth_Rate',
                    'Operating_Income_Growth',
                    'After_Tax_Net_Income_Growth',
                    'Regular_Net_Profit_Growth_Rate',
                    'Continuous_Net_Profit_Growth_Rate',
                    'Total_Asset_Growth',
                    'Total_Equity_Growth',
                    'Return_on_Total_Asset_Growth',
                    'Cash_Reinvestment_Ratio',
                    'Current_Ratio',
                    'Quick_Ratio',
                    'Interest_Expense_Ratio',
                    'Total_Liability/Equity_Ratio',
                    'Debt_ratio%',
                    'Networth/Assets',
                    'Long-term_fund_suitability_ratio(A)',
                    'Borrowing_dependency',
                    'Contingent_Liability/Equity',
                    'Operating_Income/Capital',
                    'Pretax_Income/Capital',
                    '(Inventory+Accounts_Receivables)/Equity',
                    'Total_Asset_Turnover',
                    'Accounts_Receivable_Turnover',
                    'Average_Collection_Days',
                    'Inventory_Turnover_Rate(times)',
                    'Fixed_Assets_Turnover_Frequency',
```

```
                    'Equity_Turnover',
                    'Revenue_per_person',
                    'Operating_profit_per_person',
                    'FixedAssets_Per_Employee',
                    '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_CurrentAssets',
                    'Retained_Earnings_to_Total_Assets',
                    'Total_income/Total_expense',
                    'Total_expense/Assets',
                    'CurrentAsset_TurnoverRate',
                    'QuickAsset_TurnoverRate',
                    'Workingcapitcal_TurnoverRate',
                    'Cash_Turnover_Rate',
                    'Cash_Flow_to_Sales',
                    'Fixed_Assets_to_Assets',
                    'Current_Liability_to_Liability',
                    'Current_Liability_to_Equity',
                    'Equity_to_Long-term_Liability',
                    'Cash_Flow_to_Total_Assets',
                    'Cash_Flow_to_Liability',
                    'CFO_to_Assets',
                    'Cash_Flow_to_Equity',
                    'Current_Liability_to_current_Asset',
                    'Liability-Assets_Flag',
                    'Net_Income_to_Total_asset',
                    'Total_assets_to_GNP_prine',
                    'No-credit_Interval',
                    'Gross_Profit_to_Sales',
                    'Net_Income_to_Stockholders_Equity',
                    'Liability_to_equity',
                    'Degree_of_Financial_Leverage',
                    'Interest_Coverage_Ratio',
                    'Net_Income_Flag',
                    'Equity_to_Liability']
```

In [16]:

```
bankrupt.head()
```

Out[16]:

	Bankrupt	Return_on_total_assetC	Return_on_total_assetA	Return_on_total_assetB	GrossProfit/NetSales	Realized_GrossProfit/NetSales	Operating_Income/NetSales	TaxIncome/
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	

	Bankrupt	Return_on_total_assetC	Return_on_total_assetA	Return_on_total_assetB	GrossProfit/NetSales	Realized_GrossProfit/NetSales	Operating_Income/NetSales	TaxIncome/
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	

5 rows × 96 columns

Next steps:

Check if data is balanced or not.

Correlation between the variables.

Visualization to understand the nature of variables with the y attribute and relation with each other.

Parameter selection: PCA and chi-square method.

Modeling a decision tree classifier.

Modeling a SVM machine with linear and rdf kernels.

In [17]:
```
bankrupt['Bankrupt'].value_counts().plot(kind='bar', figsize = (10,10))
```

Out[17]:
```
<AxesSubplot:>
```

From the above plot we can see that the data is imbalanced, and we will move forward with Exploratory analysis to get more insight on the same.

Correlation plot using heat map

In [9]:
```
%%time
corr_plot = bankrupt.corr()
plt.figure(figsize = (40,20))
sns.heatmap(corr_plot, annot = True)
plt.title("Heatmap for correlation between attributes")
```

```
Wall time: 4.82 s
```

Out[9]:
```
Text(0.5, 1.0, 'Heatmap for correlation between attributes')
```

Generating histograms

Columns 2-11 histogram (**Note that column 1 is Y/Dependant variable (Bankrupt))

In [10]:
```
# Generating histograms by  selecting a set of 10 columns so that it is easy to
# visualize the column values.
# we will input the columns in a list and run a for loop to generate the
# histograms, and similar will be done will all the other variables.

viz1 = ['Return_on_total_assetC',
                'Return_on_total_assetA',
```

```
                    'Return_on_total_assetB',
                    'GrossProfit/NetSales',
                    'Realized_GrossProfit/NetSales',
                    'Operating_Income/NetSales',
                    'Pre-TaxIncome/NetSales',
                    'NetIncome/NetSales',
                    'Net_Non-operating_Income_Ratio',
                    'Net_Income-Exclude_DisposalGain/Loss/Net_Sales']

for i in viz1:
    plt.figure()
    sns.histplot(x = i, data=bankrupt, bins = 70)
```

Count
Return_on_total_assetB

Count
GrossProfit/NetSales

Count
Realized_GrossProfit/NetSales

7000
6000
5000
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Operating_Income/NetSales

7000
6000
5000
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Pre-TaxIncome/NetSales

7000
6000
5000
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
NetIncome/NetSales

Columns 2-11 distribution plot (**Note that column 1 is Y/Dependant variable (Bankrupt))

In [11]:

```
viz1 = ['Return_on_total_assetC',
                    'Return_on_total_assetA',
                    'Return_on_total_assetB',
                    'GrossProfit/NetSales',
                    'Realized_GrossProfit/NetSales',
                    'Operating_Income/NetSales',
                    'Pre-TaxIncome/NetSales',
                    'NetIncome/NetSales',
                    'Net_Non-operating_Income_Ratio',
                    'Net_Income-Exclude_DisposalGain/Loss/Net_Sales']

for i in viz1:
    plt.figure()
    sns.displot(x = i, data = bankrupt, kind = "ecdf", hue = "Bankrupt")
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

Column 2-11 pariplot

In [12]:

```
# 0 here is depencant variable bankrupt.

data1 = bankrupt.iloc[:, 0:11]

sns.pairplot(data1, hue = 'Bankrupt')
```

Out[12]: <seaborn.axisgrid.PairGrid at 0x2601de09a00>

Columns 2-11 correlation plot using heatmap

In [13]:

```
# Below is the heat map from columns 0-11 where Bankrupt is dependant variable

relation = bankrupt.iloc[:,0:11]
c_plot = relation.corr()
plt.figure(figsize = (20,10))
sns.heatmap(c_plot, annot = True)
```

Out[13]:

```
<AxesSubplot:>
```

Columns 12-21 histogram

In [14]:
```
viz2 = ['Operating_Expenses/Net Sales',
                    'Research_and_DevelopmentExpenses/Net Sales',
                    'CashFlow_from_Operating/Current Liabilities',
                    'Interest-bearing Debt/Equity',
                    'Effective Tax Rate',
                    'Net_Val_Per_ShareB',
                    'Net_Val_per_ShareA',
                    'Net_Val_Per_ShareC',
```

```
                    'EPS-Net_Income',
                    'Cash_Flow_Per_Share']
for j in viz2:
    plt.figure()
    sns.histplot(x = j, data=bankrupt, bins = 70)
```

Count
Interest-bearing Debt/Equity
1e9

Count
Effective Tax Rate

Count
Net_Val_Per_ShareB

Count
Net_Val_per_ShareA

Count
Net_Val_Per_ShareC

Count
EPS-Net_Income

Columns 12-21 Distribution plot with KDE curve

In [15]:

```
viz2 = ['Operating_Expenses/Net Sales',
                     'Research_and_DevelopmentExpenses/Net Sales',
                     'CashFlow_from_Operating/Current Liabilities',
                     'Interest-bearing Debt/Equity',
                     'Effective Tax Rate',
                     'Net_Val_Per_ShareB',
                     'Net_Val_per_ShareA',
                     'Net_Val_Per_ShareC',
                     'EPS-Net_Income',
                     'Cash_Flow_Per_Share']
for j in viz2:
    plt.figure()
    sns.displot(x = j, data = bankrupt, kind = "kde", hue = "Bankrupt")
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

Column 12-21 Correlation plot using heat map

In [17]:
```
relation = bankrupt.iloc[:,11:21]
c_plot = relation.corr()
plt.figure(figsize = (20,10))
sns.heatmap(c_plot, annot = True)
```

Out[17]: <AxesSubplot:>

Column 22-31 histogram

In [18]:
```
viz3 = ['Revenue_Per_Share(Yuan¥)',
                    'Operating_Income_Per_Share(Yuan¥)',
                    'Pretax_Income_Per_Share(Yuan¥)',
                    'Realized_Sales_Gross_Profit_Growth_Rate',
                    'Operating_Income_Growth',
                    'After_Tax_Net_Income_Growth',
```

```
                    'Regular_Net_Profit_Growth_Rate',
                    'Continuous_Net_Profit_Growth_Rate',
                    'Total_Asset_Growth',
                    'Total_Equity_Growth']
for k in viz3:
    plt.figure()
    sns.histplot(x = k, data=bankrupt, bins = 70)
```

Count
Pretax_Income_Per_Share(Yuan¥)

Count
Realized_Sales_Gross_Profit_Growth_Rate

Count
Operating_Income_Growth

Count
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0
After_Tax_Net_Income_Growth

Count
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0
Regular_Net_Profit_Growth_Rate

Count
7000
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0
Continuous_Net_Profit_Growth_Rate

Column 22-31 Distribution plot

In [19]:
```
viz3 = ['Revenue_Per_Share(Yuan¥)',
                    'Operating_Income_Per_Share(Yuan¥)',
                    'Pretax_Income_Per_Share(Yuan¥)',
                    'Realized_Sales_Gross_Profit_Growth_Rate',
                    'Operating_Income_Growth',
                    'After_Tax_Net_Income_Growth',
                    'Regular_Net_Profit_Growth_Rate',
                    'Continuous_Net_Profit_Growth_Rate',
                    'Total_Asset_Growth',
                    'Total_Equity_Growth']
for l in viz3:
    plt.figure()
    sns.displot(x = l, data = bankrupt, kind = "ecdf", hue = "Bankrupt")
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

Column 21-31 Correlation matrix using heat map

In [20]:
```
relation = bankrupt.iloc[:,21:31]
c_plot = relation.corr()
plt.figure(figsize = (20,10))
sns.heatmap(c_plot, annot = True)
```

Out[20]:
```
<AxesSubplot:>
```

Columns 32-41 histogram

In [21]:
```
viz4 = ['Return_on_Total_Asset_Growth',
                    'Cash_Reinvestment_Ratio',
                    'Current_Ratio',
                    'Quick_Ratio',
                    'Interest_Expense_Ratio',
                    'Total_Liability/Equity_Ratio',
                    'Debt_ratio%',
                    'Networth/Assets',
                    'Long-term_fund_suitability_ratio(A)',
```

```
                    'Borrowing_dependency']
for m in viz4:
    plt.figure()
    sns.histplot(x = m, data=bankrupt, bins = 70)
```

7000
6000
5000
4000
3000
2000
1000
0
Count
0
2
4
6
8
Quick_Ratio
1e9

6000
5000
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Interest_Expense_Ratio

7000
6000
5000
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Total_Liability/Equity_Ratio
1e10

Count
Debt_ratio%

Count
Networth/Assets

Count
Long-term_fund_suitability_ratio(A)

Column 32-41 distribution plot

In [22]:
```
viz4 = ['Return_on_Total_Asset_Growth',
                    'Cash_Reinvestment_Ratio',
                    'Current_Ratio',
                    'Quick_Ratio',
                    'Interest_Expense_Ratio',
                    'Total_Liability/Equity_Ratio',
                    'Debt_ratio%',
                    'Networth/Assets',
                    'Long-term_fund_suitability_ratio(A)',
                    'Borrowing_dependency']
for m in viz4:
    plt.figure()
    sns.displot(x = m, data = bankrupt, kind = "ecdf", hue = "Bankrupt")
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

Columns 32-41 Correlation plot using heatmap

In [23]:
```
relation = bankrupt.iloc[:,31:41]
c_plot = relation.corr()
plt.figure(figsize = (20,10))
sns.heatmap(c_plot, annot = True)
```

Out[23]:
```
<AxesSubplot:>
```

Column 42-51 histogram

In [24]:
```
viz5 = ['Contingent_Liability/Equity',
                    'Operating_Income/Capital',
                    'Pretax_Income/Capital',
                    '(Inventory+Accounts_Receivables)/Equity',
                    'Total_Asset_Turnover',
                    'Accounts_Receivable_Turnover',
                    'Average_Collection_Days',
                    'Inventory_Turnover_Rate(times)',
                    'Fixed_Assets_Turnover_Frequency',
                    'Equity_Turnover']
```

```
for n in viz5:
    plt.figure()
    sns.histplot(x = n, data=bankrupt, bins = 70)
```

Count
(Inventory+Accounts_Receivables)/Equity

Count
Total_Asset_Turnover

Count
Accounts_Receivable_Turnover
1e10

Count
Average_Collection_Days
1e10

Count
Inventory_Turnover_Rate(times)
1e10

Count
Fixed_Assets_Turnover_Frequency
1e10

Column 42-51 distribution plot

In [25]:
```
viz5 = ['Contingent_Liability/Equity',
                    'Operating_Income/Capital',
                    'Pretax_Income/Capital',
                    '(Inventory+Accounts_Receivables)/Equity',
                    'Total_Asset_Turnover',
                    'Accounts_Receivable_Turnover',
                    'Average_Collection_Days',
                    'Inventory_Turnover_Rate(times)',
                    'Fixed_Assets_Turnover_Frequency',
                    'Equity_Turnover']
for n in viz5:
    plt.figure()
    sns.displot(x = n, data = bankrupt, kind = "ecdf", hue = "Bankrupt")
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

Column 42-51 Correlation plot using heat map

In [26]:

```
relation = bankrupt.iloc[:,41:51]
c_plot = relation.corr()
plt.figure(figsize = (20,10))
sns.heatmap(c_plot, annot = True)
```

Out[26]:

```
<AxesSubplot:>
```

Column 52-61 histogram

In [27]:
```
viz6 = ['Revenue_per_person',
                    'Operating_profit_per_person',
                    'FixedAssets_Per_Employee',
                    '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']
for o in viz6:
    plt.figure()
    sns.histplot(x = o, data=bankrupt, bins=70)
```

Count
Working_Capital_to_Total_Assets

Count
Quick_Assets/Total_Assets

Count
Current_Assets/Total_Assets

Count
Cash/Total_Assets

Count
Quick_Assets/Current_Liability
1e9

Count
Cash/Current_Liability
1e10

Column 52-61 distribution plot

In [28]:
```
viz6 = ['Revenue_per_person',
                     'Operating_profit_per_person',
                     'FixedAssets_Per_Employee',
                     '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']

for o in viz6:
    plt.figure()
    sns.displot(x = o, data = bankrupt, kind = "ecdf", hue = "Bankrupt")
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

---

column 52-61 correlation plot using heat map

In [29]:
```
relation = bankrupt.iloc[:,51:61]
c_plot = relation.corr()
plt.figure(figsize = (20,10))
sns.heatmap(c_plot, annot = True)
```

Out[29]:
```
<AxesSubplot:>
```

Column 62-71 histogram

In [30]:
```
viz7 = ['Operating_Funds_to_Liability',
                    'Inventory/Working_Capital',
                    'Inventory/Current_Liability',
                    'Current_Liabilities/Liability',
                    'Working_Capital/Equity',
                    'Current_Liabilities/Equity',
                    'Long-term_Liability_to_CurrentAssets',
                    'Retained_Earnings_to_Total_Assets',
                    'Total_income/Total_expense',
                    'Total_expense/Assets']
```

```
for p in viz7:
    plt.figure()
    sns.histplot(x = p, data = bankrupt, bins=70)
```

Count
Current_Liabilities/Liability

Count
Working_Capital/Equity

Count
Current_Liabilities/Equity

Count
Long-term_Liability_to_CurrentAssets
1e10

Count
Retained_Earnings_to_Total_Assets

Count
Total_income/Total_expense

Column 62-71 distribution plot

In [31]:
```
viz7 = ['Operating_Funds_to_Liability',
                     'Inventory/Working_Capital',
                     'Inventory/Current_Liability',
                     'Current_Liabilities/Liability',
                     'Working_Capital/Equity',
                     'Current_Liabilities/Equity',
                     'Long-term_Liability_to_CurrentAssets',
                     'Retained_Earnings_to_Total_Assets',
                     'Total_income/Total_expense',
                     'Total_expense/Assets']

for p in viz7:
    plt.figure()
    sns.displot(x = p, data = bankrupt, kind="ecdf", hue = 'Bankrupt')
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

In [ ]:

Columns 61 - 71 correlation plot using heat map

In [32]:

```
relation = bankrupt.iloc[:,61:71]
c_plot = relation.corr()
plt.figure(figsize = (20,10))
sns.heatmap(c_plot, annot = True)
```

Out[32]: <AxesSubplot:>

Column 72-81 histogram

In [33]:
```
viz8 = ['CurrentAsset_TurnoverRate',
                    'QuickAsset_TurnoverRate',
                    'Workingcapitcal_TurnoverRate',
                    'Cash_Turnover_Rate',
                    'Cash_Flow_to_Sales',
                    'Fixed_Assets_to_Assets',
                    'Current_Liability_to_Liability',
                    'Current_Liability_to_Equity',
```

```
                    'Equity_to_Long-term_Liability',
                    'Cash_Flow_to_Total_Assets']

for q in viz8:
    plt.figure()
    sns.histplot(x=q, data=bankrupt, bins=70)
```

7000
6000
5000
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Workingcapitcal_TurnoverRate

2500
2000
1500
1000
500
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
1e10
Cash_Turnover_Rate

7000
6000
5000
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Cash_Flow_to_Sales

7000
6000
5000
4000
3000
2000
1000
0
Count
0
2
4
6
8
1e9
Fixed_Assets_to_Assets

800
700
600
500
400
300
200
100
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Current_Liability_to_Liability

4000
3500
3000
2500
2000
1500
1000
500
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Current_Liability_to_Equity

Column 72-81 distribution plot

In [34]:
```
viz8 = ['CurrentAsset_TurnoverRate',
                    'QuickAsset_TurnoverRate',
                    'Workingcapitcal_TurnoverRate',
                    'Cash_Turnover_Rate',
                    'Cash_Flow_to_Sales',
                    'Fixed_Assets_to_Assets',
                    'Current_Liability_to_Liability',
                    'Current_Liability_to_Equity',
                    'Equity_to_Long-term_Liability',
                    'Cash_Flow_to_Total_Assets']

for q in viz8:
    plt.figure()
    sns.displot(x=q, data=bankrupt, kind="ecdf", hue='Bankrupt')
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

Column 71-81 correlation plot using heat map

In [35]:
```
relation = bankrupt.iloc[:,71:81]
c_plot = relation.corr()
plt.figure(figsize = (20,10))
sns.heatmap(c_plot, annot = True)
```

Out[35]:
```
<AxesSubplot:>
```

Column 82-91 histogram

In [36]:
```
viz9 = ['Cash_Flow_to_Liability',
                    'CFO_to_Assets',
                    'Cash_Flow_to_Equity',
                    'Current_Liability_to_current_Asset',
                    'Liability-Assets_Flag',
                    'Net_Income_to_Total_asset',
                    'Total_assets_to_GNP_prine',
                    'No-credit_Interval',
                    'Gross_Profit_to_Sales',
                    'Net_Income_to_Stockholders_Equity']
```

```
for r in viz9:
    plt.figure()
    sns.histplot( x=r, data=bankrupt, bins=70)
```

Count
Current_Liability_to_current_Asset

Count
Liability-Assets_Flag

Count
Net_Income_to_Total_asset

7000
6000
5000
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
1e10
Total_assets_to_GNP_prine

6000
5000
4000
3000
2000
1000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
No-credit_Interval

3500
3000
2500
2000
1500
1000
500
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
Gross_Profit_to_Sales

Column 82-91 distribution plot

In [37]:
```
viz9 = ['Cash_Flow_to_Liability',
                    'CFO_to_Assets',
                    'Cash_Flow_to_Equity',
                    'Current_Liability_to_current_Asset',
                    'Liability-Assets_Flag',
                    'Net_Income_to_Total_asset',
                    'Total_assets_to_GNP_prine',
                    'No-credit_Interval',
                    'Gross_Profit_to_Sales',
                    'Net_Income_to_Stockholders_Equity']

for r in viz9:
    plt.figure()
    sns.displot(x=r, data=bankrupt, kind="ecdf", hue = "Bankrupt")
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

Column 81-91 correlation plot using heatmap

In [38]:

```
relation = bankrupt.iloc[:,81:91]
c_plot = relation.corr()
plt.figure(figsize = (20,10))
sns.heatmap(c_plot, annot = True)
```

Out[38]:

```
<AxesSubplot:>
```

Column 92-96 histogram

In [39]:
```
viz10 = ['Liability_to_equity',
                    'Degree_of_Financial_Leverage',
                    'Interest_Coverage_Ratio',
                    'Net_Income_Flag',
                    'Equity_to_Liability']

for s in viz10:
    plt.figure()
    sns.histplot(x=s, data=bankrupt, bins=70)
```

Count
Liability_to_equity

Count
Degree_of_Financial_Leverage

Count
Interest_Coverage_Ratio

Column 92-96 distribution plot

In [40]:
```
viz10 = ['Liability_to_equity',
                    'Degree_of_Financial_Leverage',
                    'Interest_Coverage_Ratio',
                    'Net_Income_Flag',
                    'Equity_to_Liability']

for s in viz10:
    plt.figure()
    sns.displot(x=s, data=bankrupt, kind="ecdf", hue="Bankrupt")
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

Column 91-96 correlation plot using heat map

In [41]:
```
relation = bankrupt.iloc[:,91:96]
c_plot = relation.corr()
plt.figure(figsize = (20,10))
sns.heatmap(c_plot, annot = True)
```

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

After looking at all the visualizations and correlation plots we can conclue that the data points are either extremely correlated with some of the components or show no correlation with any of the components.

So, we need to use SMOTE analysis to balance the data.

SMOTE is performed on training data so we will split the data into test and training set first and then using the smote training data we will perform the models and further analysis.

However, one point to be noted is LDA and QDA analysis is done without SMOTE just to see how the variables are related / fit to the depedant variable despite the variation in the dataset.

In [ ]:

In [18]:
```
from collections import Counter
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
```

```
from matplotlib import pyplot
from numpy import where
```

In [19]:
```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 33)

#Y = bankrupt[["Bankrupt"]]
Y = bankrupt.iloc[:, 0]
X = bankrupt.iloc[:, 1:95]
```

In [20]:
```
x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size=0.3, random_state=33)
```

In [21]:
```
newX_train, newY_train = sm.fit_sample(x_train,y_train.values.ravel())
```

In [22]:
```
pd.Series(newY_train).value_counts().plot.bar()
```

Out[22]: <AxesSubplot:>

In [23]:
```
x_test.shape
```

Out[23]: (2046, 94)

In [ ]:

Linear and multivariate discriminant analysis

Iam converting all the negative variables her into positive variables for chi-square feature selection it takes positive variables only

In [25]:
```
X.apply(abs)
```

Out[25]:

	Return_on_total_assetC	Return_on_total_assetA	Return_on_total_assetB	GrossProfit/NetSales	Realized_GrossProfit/NetSales	Operating_Income/NetSales	Pre TaxIncome/NetSale
0	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887
1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380

	Return_on_total_assetC	Return_on_total_assetA	Return_on_total_assetB	GrossProfit/NetSales	Realized_GrossProfit/NetSales	Operating_Income/NetSales	Pre TaxIncome/NetSales
2	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403
3	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967
4	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366
...	...	...	...	...	...	...	...
6814	0.493687	0.539468	0.543230	0.604455	0.604462	0.998992	0.797409
6815	0.475162	0.538269	0.524172	0.598308	0.598308	0.998992	0.797414
6816	0.472725	0.533744	0.520638	0.610444	0.610213	0.998984	0.797401
6817	0.506264	0.559911	0.554045	0.607850	0.607850	0.999074	0.797500
6818	0.493053	0.570105	0.549548	0.627409	0.627409	0.998080	0.801987

6819 rows × 94 columns

In [ ]:

In [26]:

```
# defining the LDA model and importing the library
from numpy import mean
from numpy import std
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
```

In [180…

```
# declaring model
LDA = LinearDiscriminantAnalysis()

# Define model evaluation method
cv = RepeatedStratifiedKFold(n_splits = 10, n_repeats=3, random_state=33)

# evaluate model
scores = cross_val_score(LDA, X, Y, scoring='accuracy', cv=cv, n_jobs=-1)

# accuracy
print('Accuracy: %3.f(%.3f)' % (mean(scores), std(scores)))

'''
# fiting LDA model
LDA.fit(newX,newY)

# making prediction
yhat = LDA.predict([newX])

# summarising prediction
print("Predicted class: %d" % yhat)
'''
```

```
Accuracy:   1(0.005)
```

Out[180…

```
'\n# fiting LDA model\nLDA.fit(newX,newY)\n\n# making prediction \nyhat = LDA.predict([newX])\n\n# summarising prediction\nprint("Predicted class: %d" % yhat)\n'
```

Tuning LDA Huperparameter

In [181…

```
from sklearn.model_selection import GridSearchCV

LDA = LinearDiscriminantAnalysis()
cv = RepeatedStratifiedKFold(n_splits = 10, n_repeats = 3, random_state = 33)
grid = dict()
grid['solver'] = ['svd','lsqr','eigen']

search = GridSearchCV(LDA, grid, scoring='accuracy', cv=cv, n_jobs = -1)

results = search.fit(X,Y)

print('Mean Accuracy: %.3f' % results.best_score_)
print('config: %s' % results.best_params_)
```

```
Mean Accuracy: 0.963
config: {'solver': 'svd'}
```

In [182…

```
# Tuning the hyper parameters for using shrinkage hyper parameter
from numpy import arange

LDA = LinearDiscriminantAnalysis(solver = 'lsqr')
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=33)

grid = dict()
grid['shrinkage'] = arange(0, 1, 0.01)

search = GridSearchCV(LDA, grid, scoring='accuracy', cv=cv, n_jobs = -1)

results = search.fit(X,Y)

print('Mean Accuracy: %.3f' % results.best_score_)
print('config: %s' % results.best_params_)
```

```
Mean Accuracy: 0.966
config: {'shrinkage': 0.45}
```

In [183…

```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

model = QuadraticDiscriminantAnalysis()

#Result = model.fit(newX,newY)
cv = RepeatedStratifiedKFold(n_splits = 20, n_repeats=10, random_state=33)

# evaluate model
scores = cross_val_score(LDA, newX, newY, scoring='accuracy', cv=cv, n_jobs=-1)
get_params = ([newX])
print("Accuracy scores", mean(scores), std(scores))
```

```
Accuracy scores 0.962824348801104 0.004189336978568809
```

In [ ]:

In [ ]:

Decision Tree classifier

In [6]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn import decomposition
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from numpy import mean
from numpy import std
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn import feature_selection
from sklearn import tree
```

In [ ]:

In [ ]:
```
# This is just the highlight of the SMOTE that was performed earlier in order to be clear about the training and testing variables

from sklearn.model_selection import train_test_split
# Grnerating Test and Training Data
x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size=0.3, random_state=33)
```

In [ ]:
```
newX_train, newY_train = sm.fit_sample(x_train,y_train.values.ravel())
```

In [90]:
```
# Training data Shape
print("X and Y training data shape {} {}".format(newX_train.shape,newY_train.shape))
```

```
X and Y training data shape (9214, 94) (9214,)
```

In [91]:
```
# Test data shape
print("X and Y testing data shape {} {}".format(x_test.shape,y_test.shape))
```

```
X and Y testing data shape (2046, 94) (2046,)
```

In [ ]:

Feature selection

In [185…
```
from sklearn import feature_selection

fs = feature_selection.SelectPercentile(feature_selection.chi2, percentile=10)
x_train_fs = fs.fit_transform(newX_train, newY_train)
```

In [186…

```
np.set_printoptions(suppress=True, precision=2, linewidth=120)
print(list(X.columns))
print(fs.get_support())
print(fs.scores_)
```

```
['Return_on_total_assetC', 'Return_on_total_assetA', 'Return_on_total_assetB', 'GrossProfit/NetSales', 'Realized_GrossProfit/NetSales', 'Operating_I
ncome/NetSales', 'Pre-TaxIncome/NetSales', 'NetIncome/NetSales', 'Net_Non-operating_Income_Ratio', 'Net_Income-Exclude_DisposalGain/Loss/Net_Sales',
'Operating_Expenses/Net Sales', 'Research_and_DevelopmentExpenses/Net Sales', 'CashFlow_from_Operating/Current Liabilities', 'Interest-bearing Debt/
Equity', 'Effective Tax Rate', 'Net_Val_Per_ShareB', 'Net_Val_per_ShareA', 'Net_Val_Per_ShareC', 'EPS-Net_Income', 'Cash_Flow_Per_Share', 'Revenue_P
er_Share(Yuan¥)', 'Operating_Income_Per_Share(Yuan¥)', 'Pretax_Income_Per_Share(Yuan¥)', 'Realized_Sales_Gross_Profit_Growth_Rate', 'Operating_Incom
e_Growth', 'After_Tax_Net_Income_Growth', 'Regular_Net_Profit_Growth_Rate', 'Continuous_Net_Profit_Growth_Rate', 'Total_Asset_Growth', 'Total_Equity
_Growth', 'Return_on_Total_Asset_Growth', 'Cash_Reinvestment_Ratio', 'Current_Ratio', 'Quick_Ratio', 'Interest_Expense_Ratio', 'Total_Liability/Equi
ty_Ratio', 'Debt_ratio%', 'Networth/Assets', 'Long-term_fund_suitability_ratio(A)', 'Borrowing_dependency', 'Contingent_Liability/Equity', 'Operatin
g_Income/Capital', 'Pretax_Income/Capital', '(Inventory+Accounts_Receivables)/Equity', 'Total_Asset_Turnover', 'Accounts_Receivable_Turnover', 'Aver
age_Collection_Days', 'Inventory_Turnover_Rate(times)', 'Fixed_Assets_Turnover_Frequency', 'Equity_Turnover', 'Revenue_per_person', 'Operating_profi
t_per_person', 'FixedAssets_Per_Employee', 'Working_Capital_to_Total_Assets', 'Quick_Assets/Total_Assets', 'Current_Assets/Total_Assets', 'Cash/Tota
l_Assets', 'Quick_Assets/Current_Liability', 'Cash/Current_Liability', 'Current_Liability_to_Assets', 'Operating_Funds_to_Liability', 'Inventory/Wor
king_Capital', 'Inventory/Current_Liability', 'Current_Liabilities/Liability', 'Working_Capital/Equity', 'Current_Liabilities/Equity', 'Long-term_Li
ability_to_CurrentAssets', 'Retained_Earnings_to_Total_Assets', 'Total_income/Total_expense', 'Total_expense/Assets', 'CurrentAsset_TurnoverRate',
'QuickAsset_TurnoverRate', 'Workingcapitcal_TurnoverRate', 'Cash_Turnover_Rate', 'Cash_Flow_to_Sales', 'Fixed_Assets_to_Assets', 'Current_Liability_
to_Liability', 'Current_Liability_to_Equity', 'Equity_to_Long-term_Liability', 'Cash_Flow_to_Total_Assets', 'Cash_Flow_to_Liability', 'CFO_to_Asset
s', 'Cash_Flow_to_Equity', 'Current_Liability_to_current_Asset', 'Liability-Assets_Flag', 'Net_Income_to_Total_asset', 'Total_assets_to_GNP_prine',
'No-credit_Interval', 'Gross_Profit_to_Sales', 'Net_Income_to_Stockholders_Equity', 'Liability_to_equity', 'Degree_of_Financial_Leverage', 'Interest
_Coverage_Ratio', 'Net_Income_Flag']
[False False False False False False False False False False False  True False False False False False False False
 False False False False False False False False False False False  True False False False False False False False False
 False False False False False False False False False False False  True False False False False False False False False
 False  True False False False  True False False False  True False False False False  True False  True False  True
 False False False False False False False False False False False  True False False False False False False False]
[3.62e+01 4.45e+01 3.68e+01 3.91e-01 3.87e-01 2.14e-05 3.33e-04 3.42e-04 4.79e-03 2.49e-04 2.66e+10 4.96e+11 3.08e-01
 4.83e+10 2.65e+02 1.19e+01 1.21e+01 1.20e+01 1.91e+01 5.42e-01 7.55e+09 1.22e+01 2.01e+01 3.00e-04 4.77e-03 3.53e-02
 3.57e-02 3.12e-03 1.11e+11 9.25e-02 6.26e-03 2.54e-01 1.39e+01 4.44e+10 6.50e-04 3.16e+10 8.55e+01 1.49e+01 7.37e-01
 1.94e+00 7.26e+00 1.19e+01 1.85e+01 2.49e-01 3.01e+01 2.47e+10 3.36e+10 3.92e+10 1.91e+12 6.61e-01 8.07e+10 1.82e+00
 1.05e+09 1.18e+01 5.72e+01 9.70e+00 1.77e+02 1.70e+10 5.08e+11 6.39e+01 2.01e+00 1.28e-04 1.82e+11 2.50e-01 3.04e-01
 1.12e+00 8.62e+10 2.16e+00 1.20e-01 2.06e+01 1.96e+10 2.19e+11 7.74e-05 3.69e+11 5.18e-06 1.06e+11 2.50e-01 1.12e+00
 5.39e+00 1.55e+00 3.20e-01 7.54e+00 1.06e-01 3.93e+01 1.00e+00 1.33e+01 1.08e+11 1.23e-03 3.91e-01 6.76e-01 1.80e+00
 1.95e-01 1.29e-07 0.00e+00]
```

In [187…

```
print(X.columns[fs.get_support()].values)
```

```
['Research_and_DevelopmentExpenses/Net Sales' 'Total_Asset_Growth' 'Fixed_Assets_Turnover_Frequency'
 'Cash/Current_Liability' 'Inventory/Current_Liability' 'Long-term_Liability_to_CurrentAssets'
 'QuickAsset_TurnoverRate' 'Cash_Turnover_Rate' 'Fixed_Assets_to_Assets' 'Total_assets_to_GNP_prine']
```

In [188…

```
for i in range(len(X.columns.values)):
    if fs.get_support()[i]:
        print("%10s %3.2f" % (X.columns.values[i], fs.scores_[i]))
```

```
Research_and_DevelopmentExpenses/Net Sales 495689040535.19
Total_Asset_Growth 111295705995.58
Fixed_Assets_Turnover_Frequency 1909676168349.49
Cash/Current_Liability 508179720932.41
Inventory/Current_Liability 181987953557.58
Long-term_Liability_to_CurrentAssets 86162894967.42
QuickAsset_TurnoverRate 219303226939.27
Cash_Turnover_Rate 368630107487.13
Fixed_Assets_to_Assets 105502057522.28
Total_assets_to_GNP_prine 107742719530.95
```

In [189…

```
from sklearn import metrics

def measure_performance(X, y, clf, show_accuracy=True, show_classification_report=True, show_confussion_matrix=True):
    y_pred = clf.predict(X)
    if show_accuracy:
         print("Accuracy:{0:.3f}".format(metrics.accuracy_score(y, y_pred)),"\n")
    if show_classification_report:
        print("Classification report")
        print(metrics.classification_report(y, y_pred),"\n")

    if show_confussion_matrix:
        print("Confussion matrix")
        print(metrics.confusion_matrix(y, y_pred),"\n")
```

In [190…

```
from sklearn import tree

dt = tree.DecisionTreeClassifier(criterion='entropy')
dt.fit(x_train_fs, newY_train)
x_test_fs = fs.transform(x_test)
measure_performance(x_test_fs, y_test, dt, show_confussion_matrix=True)
```

```
Accuracy:0.901

Classification report
              precision    recall  f1-score   support

           0       0.98      0.92      0.95      1992
           1       0.10      0.35      0.16        54

    accuracy                           0.90      2046
   macro avg       0.54      0.63      0.55      2046
weighted avg       0.96      0.90      0.93      2046


Confussion matrix
[[1824  168]
 [  35   19]]
```

using cross validation to select features

In [191…

```
#Selecting best features by percentile using cross validation
from sklearn.model_selection import cross_val_score
dt = tree.DecisionTreeClassifier(criterion='entropy')

percentiles = range(1, 100, 5)
results = []
for i in range(1, 100, 5):
    fs = feature_selection.SelectPercentile(feature_selection.chi2, percentile=i)
    x_train_fs = fs.fit_transform(newX_train, newY_train)
    scores = cross_val_score(dt, x_train_fs, newY_train, cv=5)
    print("%2d  %0.4f" % (i, scores.mean()))
    results = np.append(results, scores.mean())
```

```
 1  0.7493
 6  0.9057
11  0.9324
```

```
16  0.9384
21  0.9445
26  0.9487
31  0.9585
36  0.9573
41  0.9570
46  0.9571
51  0.9589
56  0.9565
61  0.9559
66  0.9543
71  0.9529
76  0.9520
81  0.9551
86  0.9534
91  0.9542
96  0.9579
```

In [192…

```
optimal_percentile_ind = np.where(results == results.max())[0][0]
print("Optimal percentile of features:{0}".format(percentiles[optimal_percentile_ind]), "\n")
optimal_num_features = int(percentiles[optimal_percentile_ind]*len(X.columns)/100)
print("Optimal number of features:{0}".format(optimal_num_features), "\n")

# Plot percentile of features VS. cross-validation scores
import pylab as pl
pl.figure()
pl.xlabel("Percentage of features selected")
pl.ylabel("Cross validation accuracy")
pl.plot(percentiles,results)
```

```
Optimal percentile of features:51

Optimal number of features:47
```

Out[192…

```
[<matplotlib.lines.Line2D at 0x26031e2efd0>]
```

Evaluating the best featurs on test set

In [193…

```
fs = feature_selection.SelectKBest(feature_selection.chi2, optimal_num_features)
x_train_fs = fs.fit_transform(newX_train, newY_train)
```

```
dt = tree.DecisionTreeClassifier(criterion='entropy')
dt.fit(x_train_fs, newY_train)
x_test_fs = fs.transform(x_test)
measure_performance(x_test_fs, y_test, dt, show_confussion_matrix=False)
```

```
D:\MYSOFTWARE\Anaconda\lib\site-packages\sklearn\utils\validation.py:67: FutureWarning: Pass k=47 as keyword args. From version 0.25 passing these a
s positional arguments will result in an error
  warnings.warn("Pass {} as keyword args. From version 0.25 "
Accuracy:0.923

Classification report
              precision    recall  f1-score   support

           0       0.99      0.94      0.96      1992
           1       0.17      0.48      0.25        54

    accuracy                           0.92      2046
   macro avg       0.58      0.71      0.60      2046
weighted avg       0.96      0.92      0.94      2046
```

We will now change the criterian parameter to see how the model perform with different parameter

In [194…

```
dt = tree.DecisionTreeClassifier(criterion='entropy')
scores = cross_val_score(dt, x_train_fs, newY_train, cv=5)
print("Entropy criterion accuracy on cv: {0:.3f}".format(scores.mean()))

dt = tree.DecisionTreeClassifier(criterion='gini')
scores = cross_val_score(dt, x_train_fs, newY_train, cv=5)
print("Gini criterion accuracy on cv: {0:.3f}".format(scores.mean()))
```

```
Entropy criterion accuracy on cv: 0.959
Gini criterion accuracy on cv: 0.956
```

In [195…

```
dt = tree.DecisionTreeClassifier(criterion='entropy')
dt.fit(x_train_fs, newY_train)
x_test_fs = fs.transform(x_test)
measure_performance(x_test_fs, y_test, dt, show_confussion_matrix=False, show_classification_report=True)
```

```
Accuracy:0.924

Classification report
              precision    recall  f1-score   support

           0       0.99      0.94      0.96      1992
           1       0.17      0.50      0.26        54

    accuracy                           0.92      2046
   macro avg       0.58      0.72      0.61      2046
weighted avg       0.96      0.92      0.94      2046
```

In [7]:

```
dt = tree.DecisionTreeClassifier(criterion='gini')
dt.fit(x_train_fs, newY_train)
x_test_fs = fs.transform(x_test)
measure_performance(x_test_fs, y_test, dt, show_confussion_matrix=False, show_classification_report=True)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-7-c068e7e83629> in <module>
      1 dt = tree.DecisionTreeClassifier(criterion='gini')
----> 2 dt.fit(x_train_fs, newY_train)
      3 x_test_fs = fs.transform(x_test)
      4 measure_performance(x_test_fs, y_test, dt, show_confussion_matrix=False, show_classification_report=True)

NameError: name 'x_train_fs' is not defined
```

Evaluating the accuracy base on the depth of the tree

In [196…

```
dt = tree.DecisionTreeClassifier(criterion='entropy')
dt.set_params(max_depth=3, min_samples_leaf=10)

dt.fit(newX_train, newY_train)
measure_performance(x_test, y_test, dt, show_confussion_matrix=False, show_classification_report=True)
```

```
Accuracy:0.865

Classification report
              precision    recall  f1-score   support

           0       1.00      0.86      0.93      1992
           1       0.15      0.85      0.25        54

    accuracy                           0.86      2046
   macro avg       0.57      0.86      0.59      2046
weighted avg       0.97      0.86      0.91      2046
```

In [197…

```
from sklearn.tree import export_graphviz
from six import StringIO
from IPython.display import Image
import pydotplus
```

In [198…

```
newX_train.to_numpy()
```

Out[198…

```
array([[0.53, 0.6 , 0.58, ..., 0.03, 0.57, 1.  ],
       [0.37, 0.5 , 0.41, ..., 0.03, 0.56, 1.  ],
       [0.62, 0.67, 0.64, ..., 0.03, 0.57, 1.  ],
       ...,
       [0.43, 0.49, 0.48, ..., 0.03, 0.56, 1.  ],
       [0.23, 0.24, 0.26, ..., 0.03, 0.57, 1.  ],
       [0.4 , 0.43, 0.44, ..., 0.03, 0.56, 1.  ]])
```

In [199…

```
newY_train
```

Out[199…

```
array([0, 0, 0, ..., 1, 1, 1], dtype=int64)
```

In [200…

```
from sklearn.preprocessing import MinMaxScaler

scale = MinMaxScaler()
scale.fit(newX_train)
newX_train = scale.transform(newX_train)
```

```
x_test = scale.transform(x_test)
newX_train
```

Out[200…

```
array([[0.53, 0.6 , 0.58, ..., 0.05, 0.71, 0.  ],
       [0.37, 0.5 , 0.41, ..., 0.05, 0.71, 0.  ],
       [0.62, 0.67, 0.64, ..., 0.05, 0.71, 0.  ],
       ...,
       [0.43, 0.49, 0.48, ..., 0.05, 0.71, 0.  ],
       [0.23, 0.24, 0.26, ..., 0.05, 0.71, 0.  ],
       [0.4 , 0.43, 0.44, ..., 0.05, 0.71, 0.  ]])
```

In [208…

```
y_test
```

Out[208…

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

implementing svm

In [202…

```
newX_train.shape
```

Out[202…

```
(9214, 94)
```

In [203…

```
newY_train.shape
```

Out[203…

```
(9214,)
```

In [204…

```
x_test.shape
```

Out[204…

```
(2046, 94)
```

In [205…

```
y_test.shape
```

Out[205…

```
(2046,)
```

In [206…

```
newX_train
```

Out[206…

```
array([[0.53, 0.6 , 0.58, ..., 0.05, 0.71, 0.  ],
       [0.37, 0.5 , 0.41, ..., 0.05, 0.71, 0.  ],
       [0.62, 0.67, 0.64, ..., 0.05, 0.71, 0.  ],
       ...,
       [0.43, 0.49, 0.48, ..., 0.05, 0.71, 0.  ],
       [0.23, 0.24, 0.26, ..., 0.05, 0.71, 0.  ],
       [0.4 , 0.43, 0.44, ..., 0.05, 0.71, 0.  ]])
```

In [209…

```
from sklearn.decomposition import PCA
pca = PCA(n_components=20)
pca.fit(newX_train)
variance_by_component = pca.explained_variance_ratio_
variance_by_component
```

Out[209…

```
array([0.17, 0.15, 0.11, 0.09, 0.08, 0.07, 0.06, 0.06, 0.05, 0.04, 0.03, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,
       0.  , 0.  ])
```

In [211…

```
eigenvalue = [0.17,0.15,0.11,0.09,0.08,0.07,0.06,0.06,0.05,0.04,0.03,0.01,0.01,
```

```
              0.01,0.01,0.01,0.01,0.01,0.,0.]
components = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

plt.plot(components,eigenvalue)
plt.xlabel('components')
plt.ylabel('eigenvale')
plt.show()
```

In [217…
```
sum(pca.explained_variance_ratio_[:19])
```

Out[217…
```
0.9634674387764315
```

In [218…
```
from sklearn.decomposition import PCA
pca = PCA(n_components=19)
pca.fit(newX_train)
newX_train_PCA =pca.transform(newX_train)
x_test_PCA = pca.transform(x_test)

newX_train_PCA[0]
```

Out[218…
```
array([-0.38,  0.01, -0.  , -0.29,  0.28,  0.49,  0.4 , -0.27,  0.4 ,  0.23,  0.16,  0.05,  0.01,  0.02, -0.18,  0.8 ,
        0.11,  0.01, -0.07])
```

In [219…
```
from sklearn.svm import SVC
classifier = SVC(kernel='linear')
classifier.fit(newX_train_PCA, newY_train)
```

Out[219…
```
SVC(kernel='linear')
```

In [220…
```
y_pred = classifier.predict(x_test_PCA)
```

In [221…
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[1699  293]
 [   9   45]]
              precision    recall  f1-score   support
```

```
           0       0.99      0.85      0.92      1992
           1       0.13      0.83      0.23        54

    accuracy                           0.85      2046
   macro avg       0.56      0.84      0.57      2046
weighted avg       0.97      0.85      0.90      2046
```

In [225…

```python
from sklearn.svm import SVC
from sklearn import svm
classifier = SVC(kernel='rbf')
classifier.fit(newX_train_PCA, newY_train)
```

Out[225…

```
SVC()
```

In [226…

```python
y_pred = classifier.predict(x_test_PCA)
```

In [227…

```python
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[1875  117]
 [  29   25]]
              precision    recall  f1-score   support

           0       0.98      0.94      0.96      1992
           1       0.18      0.46      0.26        54

    accuracy                           0.93      2046
   macro avg       0.58      0.70      0.61      2046
weighted avg       0.96      0.93      0.94      2046
```

NO PCA

In [228…

```python
from sklearn.svm import SVC
classifier = SVC(kernel='linear')
classifier.fit(newX_train, newY_train)
```

Out[228…

```
SVC(kernel='linear')
```

In [229…

```python
y_pred = classifier.predict(x_test)
```

In [230…

```python
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[1723  269]
 [   8   46]]
              precision    recall  f1-score   support

           0       1.00      0.86      0.93      1992
           1       0.15      0.85      0.25        54

    accuracy                           0.86      2046
```

```
   macro avg       0.57      0.86      0.59      2046
weighted avg       0.97      0.86      0.91      2046
```

In [231…

```python
from sklearn.svm import SVC
from sklearn import svm
classifier = SVC(kernel='rbf')
classifier.fit(newX_train, newY_train)
```

Out[231…

```
SVC()
```

In [232…

```python
y_pred = classifier.predict(x_test)
```

In [233…

```python
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[1758  234]
 [  16   38]]
              precision    recall  f1-score   support

           0       0.99      0.88      0.93      1992
           1       0.14      0.70      0.23        54

    accuracy                           0.88      2046
   macro avg       0.57      0.79      0.58      2046
weighted avg       0.97      0.88      0.92      2046
```

In [ ]: