Case Study: Predicting Company Bankruptcy Using Financial Data

Data Understanding and Pre - Processing.

```
# pip install numpy pandas statsmodels matplotlib seaborn
# install required modules
```

Importing Required Libraries for DataFrames and Data-loading

```
# import numpy & panda
import pandas as pd
import numpy as np
```

Importing Required Libraries for Plotting.

```
# importing matplotlib & seaborn for plotting graphs
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing Required Libraries for Testing and Modalling.

```
# import statsmodels for statistical tests.
import statsmodels
import statsmodels.api as sm
from statsmodels.formula.api import ols
import statsmodels.stats.multicomp
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

Uploading dataset on google colab

```
from google.colab import files
uploaded=files.upload() # data.csv

# Import os to show current working directory.
import os
os.getcwd()
```

Choose files data.csv

- **data.csv**(text/csv) - 11456101 bytes, last modified: 17/06/2024 - 100% done

```
Saving data.csv to data.csv
'/content'
```

Loading provided dataset (data.csv) as Dataframe using panda library.

```
df  = pd.read_csv('data.csv')
```

Getting Details of df -> Dataframe

```
df.head(20)
## Getting first 20 rows and all columns of df.
```

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Opera Pr
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.99
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.99
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.99
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.99
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.99
5	1	0.388680	0.415177	0.419134	0.590171	0.590251	0.99
6	0	0.390923	0.445704	0.436158	0.619950	0.619950	0.99
7	0	0.508361	0.570922	0.559077	0.601738	0.601717	0.99
8	0	0.488519	0.545137	0.543284	0.603612	0.603612	0.99
9	0	0.495686	0.550916	0.542963	0.599209	0.599209	0.99
10	0	0.482475	0.567543	0.538198	0.614026	0.614026	0.99
11	0	0.444401	0.549717	0.498956	0.623712	0.623712	0.99
12	0	0.491152	0.551570	0.543391	0.608131	0.608138	0.99
13	0	0.474041	0.533308	0.523690	0.600578	0.600578	0.99
14	0	0.506703	0.575829	0.569838	0.604686	0.604686	0.99
15	0	0.513821	0.571086	0.558756	0.621773	0.621773	0.99
16	0	0.488909	0.560238	0.540286	0.606524	0.606524	0.99
17	0	0.535953	0.590438	0.580920	0.618451	0.618451	0.99
18	0	0.504071	0.559802	0.558649	0.598344	0.598344	0.99
19	0	0.487398	0.543720	0.533647	0.636259	0.636252	0.99

20 rows × 96 columns

```
df.shape
## Getting shape of df.
```

```
(6819, 96)
```

```
df.size
## Getting size of df.
```

```
654624
```

```
df.info()
## Getting info about df.
```

```
 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
```

It not just shows columns info. but it also shows there are no null values in entire df created.

```
df.describe()
## Summary Statistics.
## Getting statistical info about df.
```

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realize Sale Gros Margi
count	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.00000
mean	0.032263	0.505180	0.558625	0.553589	0.607948	0.60792
std	0.176710	0.060686	0.065620	0.061595	0.016934	0.01691
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	0.000000	0.476527	0.535543	0.527277	0.600445	0.60043
50%	0.000000	0.502706	0.559802	0.552278	0.605997	0.60597
75%	0.000000	0.535563	0.589157	0.584105	0.613914	0.61384
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000

8 rows × 96 columns

```
df.isnull().sum()
## Getting count of null values in each column.
```

```
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
                                                         ..
 Liability to Equity                                      0
 Degree of Financial Leverage (DFL)                       0
 Interest Coverage Ratio (Interest expense to EBIT)       0
 Net Income Flag                                          0
 Equity to Liability                                      0
Length: 96, dtype: int64
```

It shows there are no null values in entire df.

```
df.duplicated().sum()
## Getting count of duplicated values in df.
```

```
0
```

It shows there no duplicate values in entire df.

```
df.shape
```

```
(6819, 96)
```

Observation:

1. df has 6819 records & 96 columns. Columns includes financial attributes and their bankruptcy status in Bankrupt?.

2. LDatatype are float64 or int64.

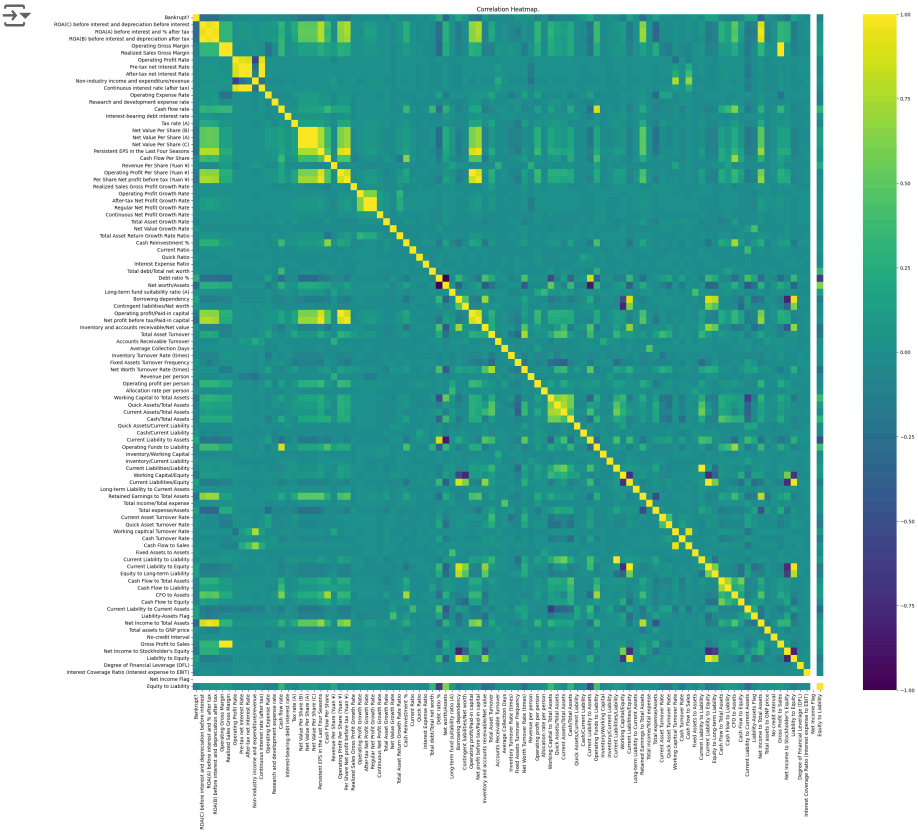
3. There is no null or duplicated values.

Problem:

The target column Bankrupt? is imbalanced, with only 3% of total values is 1, which indicates the bankrupt status.

EXPLORATORY DATA ANALYSIS.

```
# Check corelations between various features of df using heatmap

# Createing correlation matrix
corr_matrix = df.corr()

# Generate the heatmap
plt.figure(figsize=(29, 25))
sns.heatmap(corr_matrix, cmap="viridis")
plt.title("Correlation Heatmap.")
plt.show()
```

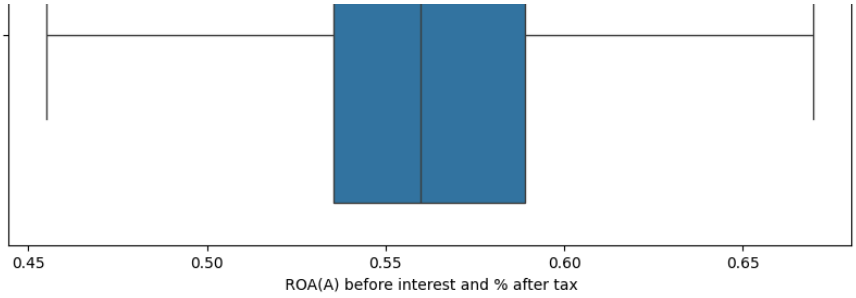

Correlation Heatmap.
Bankrupt?
ROA(C) before interest and depreciation before interest
ROA(A) before interest and % after tax
ROA(B) before interest and depreciation after tax
Operating Gross Margin
Realized Sales Gross Margin
Operating Profit Rate
Pre-tax net Interest Rate
After-tax net Interest Rate
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 capital 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
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

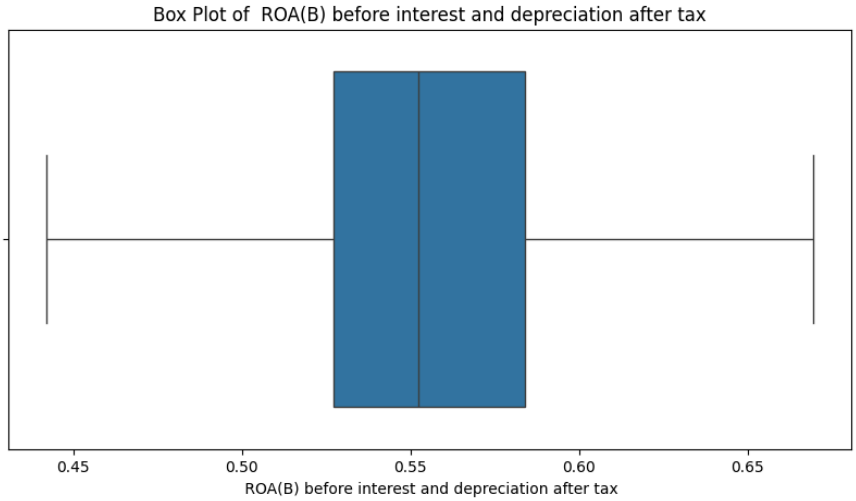
```
'''
 **Analysis:**

- There are several features that have a strong positive correlation with the target
    - Net Income to Total Assets
    - Total Debt to Total Assets
    - Debt to Equity
    - Interest Expense to Total Assets
    - Sales to Total Assets
    - Current Assets to Total Assets
    - Inventory to Total Assets
- There are also several features that have a strong negative correlation with the t
    - Return on Assets
    - Return on Equity
    - Gross Profit Margin
    - Operating Profit Margin
    - Net Profit Margin
    - Quick Ratio
    - Current Ratio
- The heatmap also reveals that there are several features that are highly correlate
    - Total Debt to Total Assets and Debt to Equity
    - Sales to Total Assets and Current Assets to Total Assets
    - Inventory to Total Assets and Current Assets to Total Assets

**Recommendations:**

- Given the strong correlations between several features and the target variable, it
- It may also be beneficial to consider using dimensionality reduction techniques, s

**Insights:**

- The heatmap provides valuable insights into the relationships between the differen
    - Identify potential risk factors for bankruptcy.
    - Develop early warning systems for bankruptcy.
    - Improve the accuracy of predictive models for bankruptcy.
'''
```

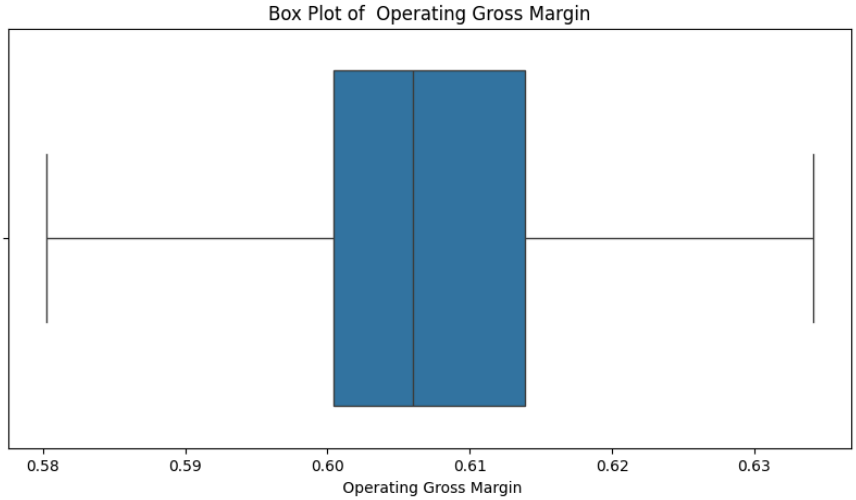
Detecting Outliers from df.

With Outliers.

```
# Detect outliers using IQR
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1

# Define lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Visualize outliers for a few selected features using box plots
selected_features = df.columns[:5]  # Replace with actual feature names or a subset
for feature in selected_features:
    plt.figure(figsize=(10, 5))
    sns.boxplot(x=df[feature])
    plt.title(f'Box Plot of {feature}')
    plt.show()
```

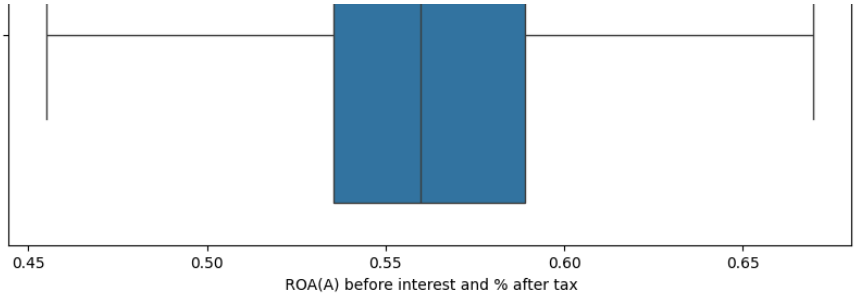

0.45
0.50
0.55
0.60
0.65
ROA(A) before interest and % after tax

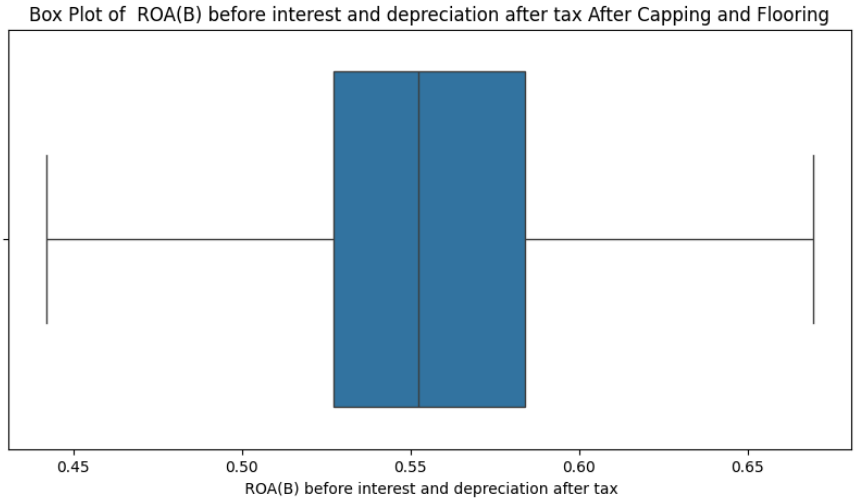

Box Plot of ROA(B) before interest and depreciation after tax
0.45
0.50
0.55
0.60
0.65
ROA(B) before interest and depreciation after tax

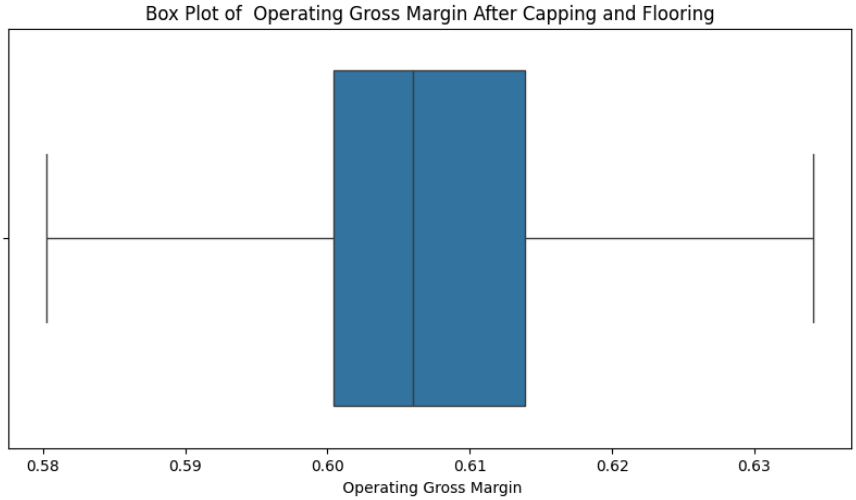

Box Plot of Operating Gross Margin
0.58
0.59
0.60
0.61
0.62
0.63
Operating Gross Margin

Without Outliers.

```
# Cap and floor outliers
for column in df.columns:
    if df[column].dtype != 'object' and column != 'Bankrupt?':
        df[column] = np.where(df[column] > upper_bound[column], upper_bound[column],
        df[column] = np.where(df[column] < lower_bound[column], lower_bound[column],

# Verify that outliers have been capped and floored
for feature in selected_features:
    plt.figure(figsize=(10, 5))
    sns.boxplot(x=df[feature])
    plt.title(f'Box Plot of {feature} After Capping and Flooring')
    plt.show()
```


0.45
0.50
0.55
0.60
0.65
ROA(A) before interest and % after tax


Box Plot of ROA(B) before interest and depreciation after tax After Capping and Flooring
0.45
0.50
0.55
0.60
0.65
ROA(B) before interest and depreciation after tax


Box Plot of Operating Gross Margin After Capping and Flooring
0.58
0.59
0.60
0.61
0.62
0.63
Operating Gross Margin

```
'''
---------------------------- **Problem:** Imbalanced dataset, Feature Selection and Out

**Solution:**
Choose Random Forest model:

For imbalanced data: Diverse trees and result voting helps this model reducing bias t
For feature selection: Random features selection and feature importance scores helps
For outliers: Result voting among trees makes this model less sensitive to outliers.

**Capping:** Replace outliers with the maximum or minimum value of the feature.
**Flooring:** Replace outliers with the median or mean of the feature.
'''
```

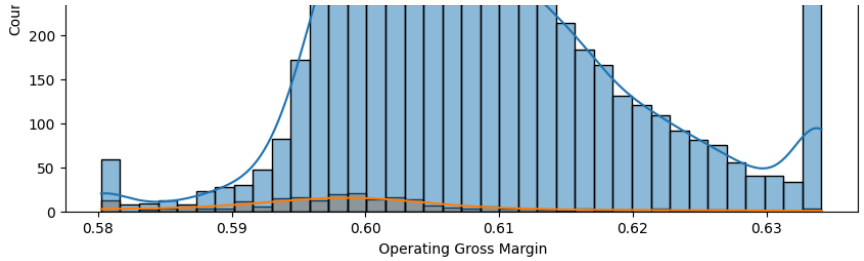
```
# Check for remaining outliers
outliers_after = ((df < lower_bound) | (df > upper_bound)).sum()
print("Number of outliers in each feature after capping and flooring:\n", outliers_a

# Generate descriptive statistics
df.describe()
```

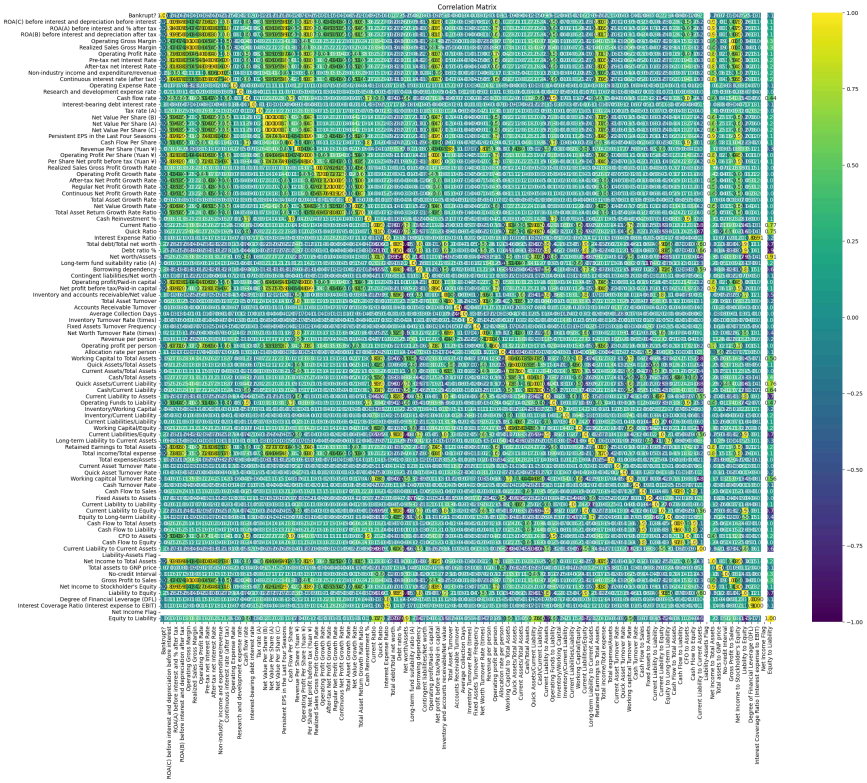
```
Number of outliers in each feature after capping and flooring:
 Bankrupt?                                                   220
 ROA(C) before interest and depreciation before interest      0
 ROA(A) before interest and % after tax                       0
 ROA(B) before interest and depreciation after tax            0
 Operating Gross Margin                                       0
                                                           ...
 Liability to Equity                                          0
 Degree of Financial Leverage (DFL)                           0
 Interest Coverage Ratio (Interest expense to EBIT)           0
 Net Income Flag                                              0
 Equity to Liability                                          0
Length: 96, dtype: int64
```

	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	Realize Sale Gros Margi
count	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.00000
mean	0.032263	0.505667	0.560507	0.554691	0.607834	0.60781
std	0.176710	0.051483	0.050087	0.050277	0.010758	0.01072
min	0.000000	0.387973	0.455122	0.442034	0.580240	0.58032
25%	0.000000	0.476527	0.535543	0.527277	0.600445	0.60043
50%	0.000000	0.502706	0.559802	0.552278	0.605997	0.60597
75%	0.000000	0.535563	0.589157	0.584105	0.613914	0.61384
max	1.000000	0.624116	0.669579	0.669348	0.634118	0.63395

8 rows × 96 columns

```python
# Visualizing the distribution of the Target Variable
plt.figure(figsize=(8, 6))
sns.countplot(x='Bankrupt?', data=df, palette=palette)
palette = ["skyblue", "coral"]
plt.xlabel('Bankrupt?')
plt.ylabel('Count')
plt.title('Distribution of Target Variable (Bankrupt?)')
plt.show()

# Visualizing the distributions of a few selected features
selected_features = df.columns[:5]  # Replace with actual feature names or a subset
for feature in selected_features:
    plt.figure(figsize=(10, 5))
    sns.histplot(df, x=feature, hue='Bankrupt?', kde=True)
    plt.title(f'Distribution of {feature}')
    plt.show()

# Calculating the Correlation Matrix
corr_matrix = df.corr()

# Visualizing the Correlation Matrix
plt.figure(figsize=(25, 20))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='viridis')
plt.title('Correlation Matrix')
plt.xticks.top = False
plt.yticks.right = False
plt.show()

# Focus on correlations with the target variable
target_corr = corr_matrix['Bankrupt?'].sort_values(ascending=False)
print(target_corr)
```






```
Bankrupt?                                  1.000000
 Borrowing dependency                      0.278367
 Total debt/Total net worth                0.272914
 Debt ratio %                              0.246535
 Liability to Equity                       0.246176
                                               ...
 Retained Earnings to Total Assets        -0.255218
 Net Income to Total Assets               -0.255797
 Persistent EPS in the Last Four Seasons  -0.256159
 Liability-Assets Flag                          NaN
 Net Income Flag                                NaN
Name: Bankrupt?, Length: 96, dtype: float64
```

```
'''
**Analysis:**

- The heatmap of the correlation matrix shows that there are several features that h
- The box plots show that there are outliers in some of the features. These outliers
- The distribution of the target variable shows that the data is imbalanced, with or
- The histograms show the distribution of the features for both bankrupt and non-bar

**Recommendations:**

- Further investigate the features that have a strong correlation with the target va
- Consider using outlier detection and removal techniques to improve the accuracy of
- Use oversampling or undersampling techniques to address the imbalance in the targe
- Use a variety of different feature selection and machine learning algorithms to fi

**Insights:**

- The data provides valuable insights into the financial characteristics of bankrupt
- The analysis of the data can be used to develop early warning systems for bankrupt
'''
```

EDA as per Target Variable -> Bankrupt?

```
df["Bankrupt?"].value_counts(normalize=True)
## Getting class frequency.
    ## 0 -> Not Bankrupt
    ## 1 -> Bankrupt
```

```
Bankrupt?
0    0.967737
1    0.032263
Name: proportion, dtype: float64
```

```
# Set benchmark for features correlation significance with target
corr_features = corr_matrix["Bankrupt?"].abs() >= 0.15
corr_features.value_counts()
    ## True -> Significant
    ## False -> Not significant
```

```
Bankrupt?
False    74
True     22
Name: count, dtype: int64
```

Identification of features (variables) in the dataset have a significant correlation with the target variable Bankrupt?.

```
df['Bankrupt?'].value_counts()
    ## 0 -> Not Bankrupt
    ## 1 -> Bankrupt
```

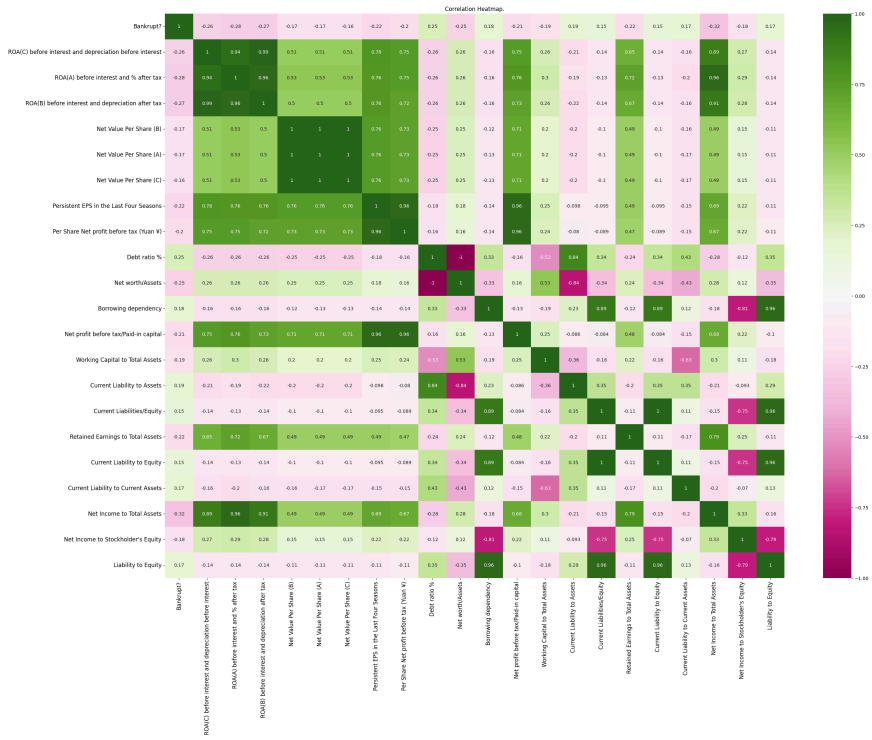
```
Bankrupt?
0    6599
1     220
Name: count, dtype: int64
```

```
# Get list of names for features passing the benchmark
feature_names = list(corr_features[corr_features].index)
feature_names
```

```
['Bankrupt?',
 ' ROA(C) before interest and depreciation before interest',
 ' ROA(A) before interest and % after tax',
 ' ROA(B) before interest and depreciation after tax',
 ' Net Value Per Share (B)',
 ' Net Value Per Share (A)',
```

```
' Net Value Per Share (C)',
' Persistent EPS in the Last Four Seasons',
' Per Share Net profit before tax (Yuan ¥)',
' Debt ratio %',
' Net worth/Assets',
' Borrowing dependency',
' Net profit before tax/Paid-in capital',
' Working Capital to Total Assets',
' Current Liability to Assets',
' Current Liabilities/Equity',
' Retained Earnings to Total Assets',
' Current Liability to Equity',
' Current Liability to Current Assets',
' Net Income to Total Assets',
" Net Income to Stockholder's Equity",
' Liability to Equity']
```

```
# Check corelations between selected features

# Create correlation matrix
mini_corr_matrix = df[feature_names].corr()

# Generate the heatmap
plt.figure(figsize=(30, 22))
sns.heatmap(mini_corr_matrix, annot = True, cmap="PiYG")
plt.title("Correlation Heatmap.")
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.xticks.top = False
plt.yticks.right = False
plt.show()
```



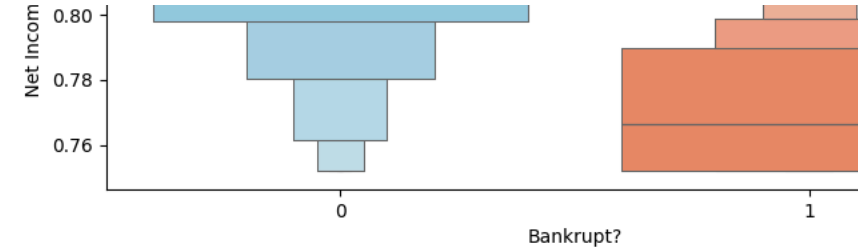

```
'''
As, 0.15 is the benchmark for correlation significance, so only 22 attributes have a

The choice based on correlation significance with the target only might not be the b
'''
```

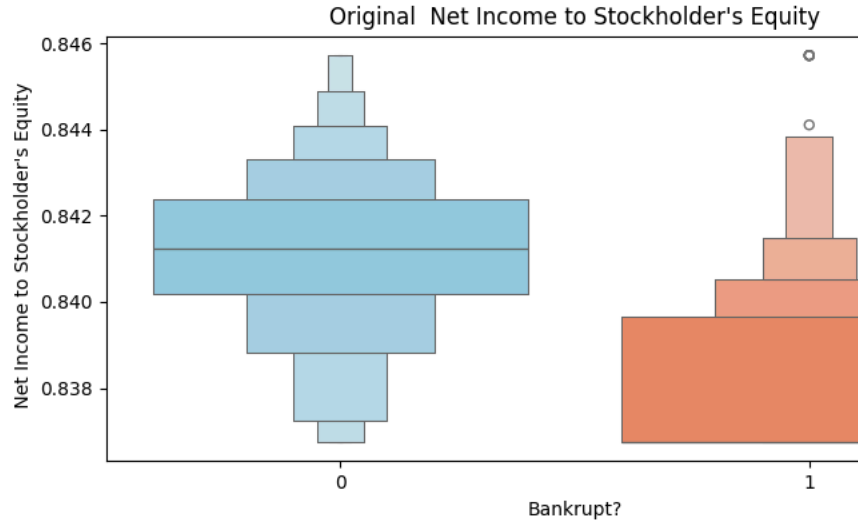
Detecting Outliers from df with target variable.

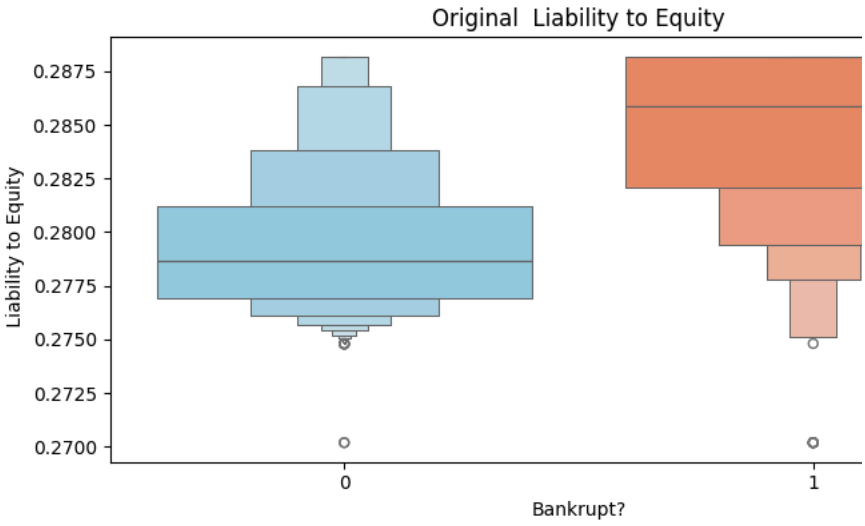
Plots of Target Variable with feature_names

```
# Create a Figure with Subplots
fig, axes = plt.subplots(len(feature_names), 2, figsize=(16, len(feature_names) * 4)

for i, feature in enumerate(feature_names):
    # Plot with original feature
    palette = ["skyblue", "coral"]
    sns.boxenplot(x="Bankrupt?", y=feature, data=df, ax=axes[i, 0], palette=palette)
    axes[i, 0].set_title(f'Original {feature}')

    # Plot after removing outliers in feature
    q1, q9 = df[feature].quantile([0.1, 0.9])
    mask = df[feature].between(q1, q9)
    palette = ["skyblue", "coral"]
    sns.boxplot(x="Bankrupt?", y=feature, data=df[mask], ax=axes[i, 1], palette=pale
    axes[i, 1].set_title(f'{feature} without Outliers')

# Adjust layout
plt.tight_layout()
plt.show()
```

[4]


Net Incom
0.80
0.78
0.76
0
1
Bankrupt?

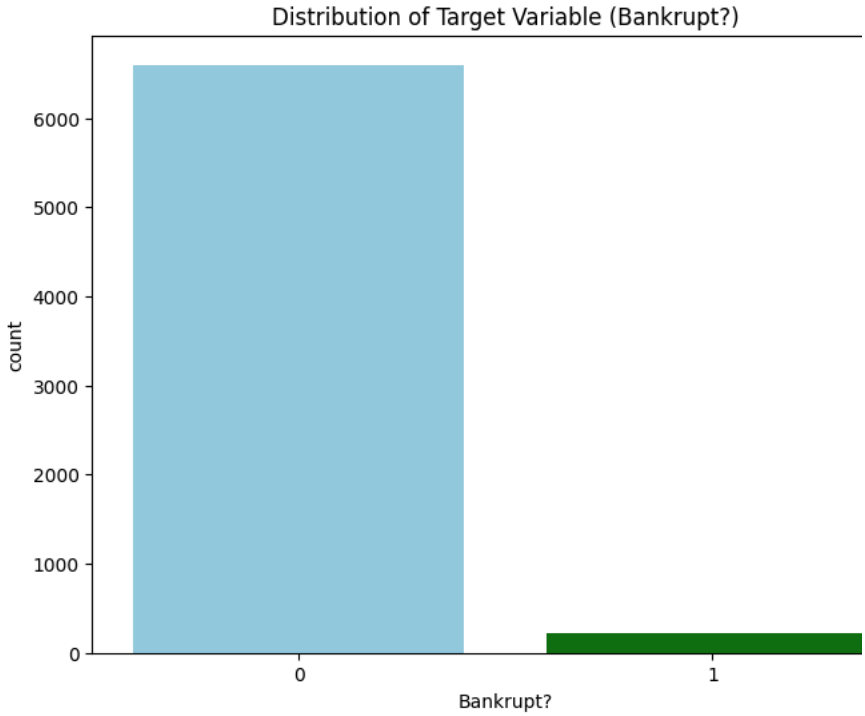

Original Net Income to Stockholder's Equity
Net Income to Stockholder's Equity
0.846
0.844
0.842
0.840
0.838
0
1
Bankrupt?


Original Liability to Equity
Liability to Equity
0.2875
0.2850
0.2825
0.2800
0.2775
0.2750
0.2725
0.2700
0
1
Bankrupt?

```
'''
**Analysis:**

- The boxplots show that there are outliers in some of the features for both bankrup
- The outliers could potentially affect the results of any predictive model that is
- After removing the outliers, the boxplots show that the distributions of the featu

**Recommendations:**

- Consider using outlier detection and removal techniques to improve the accuracy of
- Use a variety of different feature selection and machine learning algorithms to fi

**Insights:**

- The data provides valuable insights into the financial characteristics of bankrupt
- The analysis of the data can be used to develop early warning systems for bankrupt
'''
```

```
# Visualize the distribution of the Target Variable via Count Plot.
plt.figure(figsize=(8, 6))
palette = ["skyblue", "green"]
sns.countplot(x='Bankrupt?', data=df, palette=palette)
plt.title('Distribution of Target Variable (Bankrupt?)')
plt.show()
```

```
<ipython-input-94-0a59690e38ac>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in

  sns.countplot(x='Bankrupt?', data=df, palette=palette)
```



There is Imbalance in the Target Value, we need to oversample it....

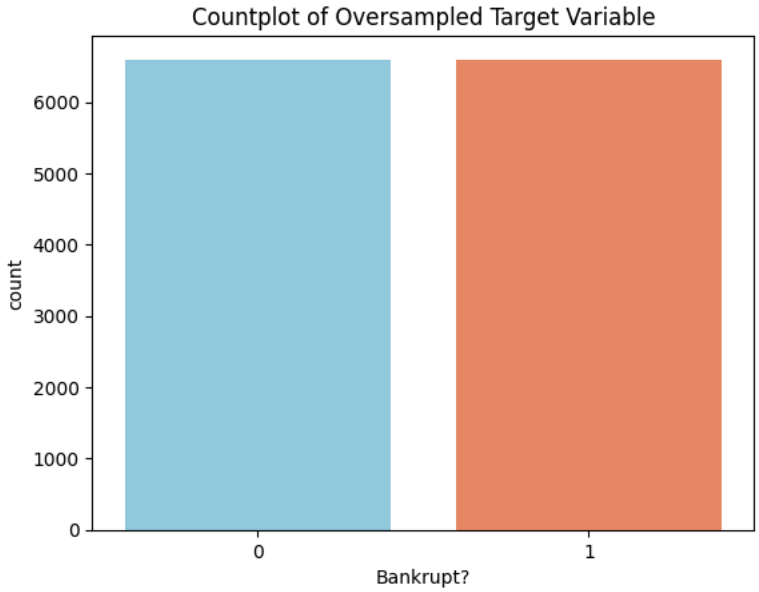
Oversampling the Target value.

```
from imblearn.over_sampling import SMOTE
X = df.drop('Bankrupt?', axis=1)
y = df['Bankrupt?']
```

Oversampling the Target Variable using SMOTE

```
oversample = SMOTE()
X,y=oversample.fit_resample(X,y)
# Create the countplot with a custom color palette
palette = ["skyblue", "coral"]
sns.countplot(x=y, palette=palette)  # Countplot based on the oversampled target var
plt.title("Countplot of Oversampled Target Variable")
plt.show()
```

```
<ipython-input-85-59fb5121f58a>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed i

  sns.countplot(x=y, palette=palette)  # Countplot based on the oversampled t
```



Observation:

- Assume 0.15 is the benchmark for correlation significance, so only 22 attributes have at least weak correlation with the target.
- The rest attributes have correlation with target less than the benchmark, which means they have too weak or even no relation with the target.

Problem:

The choice based on correlation significance with the target only might not be the best choice. The reason is that correlation assume linear relationship between features.

#Preperation for Analyzing and applying Tests and Modals.

```
from scipy import stats
from scipy.stats import ttest_ind
```

```
# Seperate the data into Bankrupt and Non Bankrupt.
bankrupt = df[df['Bankrupt?'] == 1]
non_bankrupt = df[df['Bankrupt?'] == 0]
```

Hypothesis Testing ->

1. **t-tests.**
2. **Chi Square Testing.**

t-tests.

```
# Perform t-tests
p_values = {}
for column in df.columns:
    if column != 'Bankrupt?':
        _, p_value = ttest_ind(bankrupt[column], non_bankrupt[column], equal_var=Fal
        p_values[column] = p_value
```

```
/usr/local/lib/python3.10/dist-packages/scipy/stats/_axis_nan_policy.py:523: RuntimeWarning: Precision loss occurred in
  res = hypotest_fun_out(*samples, **kwds)
```

Chi Square Testing.

```
# Create a contingency table for each feature
contingency_tables = {}
for column in df.columns:
    if column != 'Bankrupt?':
        contingency_tables[column] = pd.crosstab(df['Bankrupt?'], df[column])

# Perform chi-square tests
chi_square_results = {}
for column, table in contingency_tables.items():
    chi_square_results[column] = stats.chi2_contingency(table)

# Print the chi-square statistics and p-values
for column, result in chi_square_results.items():
    print(f"**Feature:** {column}")
    print(f"Chi-square statistic: {result[0]:.4f}")
    print(f"P-value: {result[1]:.4f}")
    print()
```

```
P-value: 1.0000

**Feature:**  Gross Profit to Sales
Chi-square statistic: 6451.5692
P-value: 0.6627

**Feature:**  Net Income to Stockholder's Equity
Chi-square statistic: 3925.2115
P-value: 1.0000

**Feature:**  Liability to Equity
Chi-square statistic: 4803.9360
P-value: 1.0000

**Feature:**  Degree of Financial Leverage (DFL)
Chi-square statistic: 3459.5523
P-value: 1.0000

**Feature:**  Interest Coverage Ratio (Interest expense to EBIT)
Chi-square statistic: 3113.3155
P-value: 1.0000

**Feature:**  Net Income Flag
Chi-square statistic: 0.0000
P-value: 1.0000

**Feature:**  Equity to Liability
Chi-square statistic: 6755.1758
P-value: 0.0000
```

Analyzing the results of the t-tests and chi-square tests.

```
# T-tests
significant_t_tests = [column for column, p_value in p_values.items() if p_value < 0
print(f"There are {len(significant_t_tests)} features with significant differences b
for column in significant_t_tests:
    print(f"\t- {column}")

# Chi-square tests
significant_chi_square_tests = [column for column, result in chi_square_results.item
print(f"\nThere are {len(significant_chi_square_tests)} features with significant di
for column in significant_chi_square_tests:
    print(f"\t- {column}")

# Compare the results of the two tests
common_features = set(significant_t_tests) & set(significant_chi_square_tests)
print(f"\nThere are {len(common_features)} features that are significant in both t-t
for column in common_features:
    print(f"\t- {column}")
```

- Persistent EPS in the Last Four Seasons
- ROA(A) before interest and % after tax
- Total Asset Growth Rate
- Cash Flow to Total Assets
- Net Value Per Share (C)
- Net Value Per Share (B)
- Total Asset Return Growth Rate Ratio
- Inventory/Current Liability
- Long-term Liability to Current Assets
- Cash/Total Assets
- Operating profit per person
- Net worth/Assets
- ROA(B) before interest and depreciation after tax
- Cash/Current Liability
- Realized Sales Gross Margin
- Quick Assets/Current Liability
- Operating Profit Per Share (Yuan ¥)
- Net Value Growth Rate
- Debt ratio %
- Quick Ratio
- Total Asset Turnover
- Non-industry income and expenditure/revenue
- Net profit before tax/Paid-in capital
- Contingent liabilities/Net worth
- Operating Profit Rate
- Current Ratio
- Operating profit/Paid-in capital
- Per Share Net profit before tax (Yuan ¥)
- Equity to Liability

Generating following for both t-tests and Chi Square Tests.

1. Test Type
2. Null Hypothesis
3. Alternative Hypothesis
4. Result
5. Conclusion