Problem Statement

Context

In the realm of modern finance, businesses encounter the perpetual challenge of managing debt obligations effectively to maintain a favorable credit standing and foster sustainable growth. Investors keenly scrutinize companies capable of navigating financial complexities while ensuring stability and profitability. A pivotal instrument in this evaluation process is the balance sheet, which provides a comprehensive overview of a company's assets, liabilities, and shareholder equity, offering insights into its financial health and operational efficiency. In this context, leveraging available financial data, particularly from preceding fiscal periods, becomes imperative for informed decision-making and strategic planning.

Objective

A renowned credit rating organization wants to develop a Financial Health Assessment Tool. With the help of the tool, it endeavors to empower businesses and investors with a robust mechanism for evaluating the financial well-being and creditworthiness of companies. By harnessing machine learning techniques, the organization aims to analyze historical financial statements and extract pertinent insights to facilitate informed decision-making via the tool. Specifically, the organization foresees facilitating the following with the help of the tool:

1. Debt Management Analysis: Identify patterns and trends in debt management practices to assess the ability of businesses to fulfill financial obligations promptly and efficiently, and identify potential cases of default.

2. Credit Risk Evaluation: Evaluate credit risk exposure by analyzing liquidity ratios, debt-to-equity ratios, and other key financial indicators to ascertain the likelihood of default and inform investment decisions.

As a part of the data science team in the organization, you have been provided with the financial metrics of different companies. The task is to analyze the data provided and develop a predictive model leveraging machine learning techniques to identify whether a given company will default on its debt repayments in the next two quarters. The predictive model will help the organization anticipate potential challenges with the financial performance of the companies and enable proactive risk mitigation strategies.

Data Dictionary

The data consists of financial metrics from the balance sheets of different companies. The detailed data dictionary is available in the data dictionary file (*FRA_DataDictionary.xlsx*).

Please read the instructions carefully before starting the project.

This is a commented Python Notebook file in which all the instructions and tasks to be performed are mentioned.

- Blanks '_______' are provided in the notebook that needs to be filled with an appropriate code to get the correct result. With every '_______' blank, there is a comment that briefly describes what needs to be filled in the blank space.
- Identify the task to be performed correctly, and only then proceed to write the required code.
- Fill the code wherever asked by the commented lines like "# write your code here" or "# complete the code". Running incomplete code may throw error.
- Please run the codes in a sequential manner from the beginning to avoid any unnecessary errors.
- Add the results/observations (wherever mentioned) derived from the analysis in the presentation and submit the same.

Importing necessary libraries

```
import numpy as np
import pandas as pd
import io

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
from scipy import stats
from statsmodels.stats.outliers_influence import
variance_inflation_factor
from sklearn.model_selection import train_test_split, GridSearchCV
# Train test Split and Grid Search
from sklearn.ensemble import RandomForestClassifier

import statsmodels.api as SM
from sklearn import metrics

from sklearn.metrics import (
    confusion_matrix,
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
```

```
    roc_curve,
    roc_auc_score
)

import warnings
warnings.filterwarnings('ignore')

from IPython.core.display import display, HTML
display(HTML('<style>.container { width:90% !important; }<\style>'))
```

```
<IPython.core.display.HTML object>
```

Loading the Data

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
df =
pd.read_csv('/content/drive/MyDrive/Finance_and_Risk_Analytics/project
_coded/CompData_1.csv')
```

Data Overview

```
df.head() ##  Complete the code to view top 5 rows of the data
```

{"type":"dataframe","variable_name":"df"}

```
df.tail() ##  Complete the code to view last 5 rows of the data
```

{"type":"dataframe"}

```
df.shape ##  Complete the code to view dimensions of the data
```

```
(2058, 58)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2058 entries, 0 to 2057
Data columns (total 58 columns):
 #   Column                                             Non-Null Count
Dtype
---  ------                                             --------------
-----
 0   Co_Code                                            2058 non-null
```

```
int64
 1   Co_Name                                             2058 non-null   object
 2   _Operating_Expense_Rate                             2058 non-null   float64
 3   _Research_and_development_expense_rate              2058 non-null   float64
 4   _Cash_flow_rate                                     2058 non-null   float64
 5   _Interest_bearing_debt_interest_rate                2058 non-null   float64
 6   _Tax_rate_A                                         2058 non-null   float64
 7   _Cash_Flow_Per_Share                                1891 non-null   float64
 8   _Per_Share_Net_profit_before_tax_Yuan_              2058 non-null   float64
 9   _Realized_Sales_Gross_Profit_Growth_Rate            2058 non-null   float64
 10  _Operating_Profit_Growth_Rate                       2058 non-null   float64
 11  _Continuous_Net_Profit_Growth_Rate                  2058 non-null   float64
 12  _Total_Asset_Growth_Rate                            2058 non-null   float64
 13  _Net_Value_Growth_Rate                              2058 non-null   float64
 14  _Total_Asset_Return_Growth_Rate_Ratio               2058 non-null   float64
 15  _Cash_Reinvestment_perc                             2058 non-null   float64
 16  _Current_Ratio                                      2058 non-null   float64
 17  _Quick_Ratio                                        2058 non-null   float64
 18  _Interest_Expense_Ratio                             2058 non-null   float64
 19  _Total_debt_to_Total_net_worth                      2037 non-null   float64
 20  _Long_term_fund_suitability_ratio_A                 2058 non-null   float64
 21  _Net_profit_before_tax_to_Paid_in_capital           2058 non-null   float64
 22  _Total_Asset_Turnover                               2058 non-null   float64
 23  _Accounts_Receivable_Turnover                       2058 non-null   float64
 24  _Average_Collection_Days                            2058 non-null   float64
```

```
 25  _Inventory_Turnover_Rate_times                    2058 non-null
float64
 26  _Fixed_Assets_Turnover_Frequency                  2058 non-null
float64
 27  _Net_Worth_Turnover_Rate_times                    2058 non-null
float64
 28  _Operating_profit_per_person                      2058 non-null
float64
 29  _Allocation_rate_per_person                       2058 non-null
float64
 30  _Quick_Assets_to_Total_Assets                     2058 non-null
float64
 31  _Cash_to_Total_Assets                             1962 non-null
float64
 32  _Quick_Assets_to_Current_Liability                2058 non-null
float64
 33  _Cash_to_Current_Liability                        2058 non-null
float64
 34  _Operating_Funds_to_Liability                     2058 non-null
float64
 35  _Inventory_to_Working_Capital                     2058 non-null
float64
 36  _Inventory_to_Current_Liability                   2058 non-null
float64
 37  _Long_term_Liability_to_Current_Assets            2058 non-null
float64
 38  _Retained_Earnings_to_Total_Assets                2058 non-null
float64
 39  _Total_income_to_Total_expense                    2058 non-null
float64
 40  _Total_expense_to_Assets                          2058 non-null
float64
 41  _Current_Asset_Turnover_Rate                      2058 non-null
float64
 42  _Quick_Asset_Turnover_Rate                        2058 non-null
float64
 43  _Cash_Turnover_Rate                               2058 non-null
float64
 44  _Fixed_Assets_to_Assets                           2058 non-null
float64
 45  _Cash_Flow_to_Total_Assets                        2058 non-null
float64
 46  _Cash_Flow_to_Liability                           2058 non-null
float64
 47  _CFO_to_Assets                                    2058 non-null
float64
 48  _Cash_Flow_to_Equity                              2058 non-null
float64
 49  _Current_Liability_to_Current_Assets              2044 non-null
```

```
float64
 50  _Liability_Assets_Flag                                  2058 non-null
int64
 51  _Total_assets_to_GNP_price                              2058 non-null
float64
 52  _No_credit_Interval                                     2058 non-null
float64
 53  _Degree_of_Financial_Leverage_DFL                       2058 non-null
float64
 54  _Interest_Coverage_Ratio_Interest_expense_to_EBIT       2058 non-null
float64
 55  _Net_Income_Flag                                        2058 non-null
int64
 56  _Equity_to_Liability                                    2058 non-null
float64
 57  Default                                                 2058 non-null
int64
dtypes: float64(53), int64(4), object(1)
memory usage: 932.7+ KB
```

```
# Remove '_' (startswith) from column headers where present
for col in df.columns:
    if col.startswith('_'):
        df.rename(columns={col: col[1:]}, inplace=True)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2058 entries, 0 to 2057
Data columns (total 58 columns):
 #   Column                                                 Non-Null Count
Dtype
---  ------                                                 --------------
-----
 0   Co_Code                                                2058 non-null
int64
 1   Co_Name                                                2058 non-null
object
 2   Operating_Expense_Rate                                 2058 non-null
float64
 3   Research_and_development_expense_rate                  2058 non-null
float64
 4   Cash_flow_rate                                         2058 non-null
float64
 5   Interest_bearing_debt_interest_rate                    2058 non-null
float64
 6   Tax_rate_A                                             2058 non-null
float64
 7   Cash_Flow_Per_Share                                    1891 non-null
float64
```

```
 8   Per_Share_Net_profit_before_tax_Yuan_                  2058 non-null
float64
 9   Realized_Sales_Gross_Profit_Growth_Rate                2058 non-null
float64
 10  Operating_Profit_Growth_Rate                           2058 non-null
float64
 11  Continuous_Net_Profit_Growth_Rate                      2058 non-null
float64
 12  Total_Asset_Growth_Rate                                2058 non-null
float64
 13  Net_Value_Growth_Rate                                  2058 non-null
float64
 14  Total_Asset_Return_Growth_Rate_Ratio                   2058 non-null
float64
 15  Cash_Reinvestment_perc                                 2058 non-null
float64
 16  Current_Ratio                                          2058 non-null
float64
 17  Quick_Ratio                                            2058 non-null
float64
 18  Interest_Expense_Ratio                                 2058 non-null
float64
 19  Total_debt_to_Total_net_worth                          2037 non-null
float64
 20  Long_term_fund_suitability_ratio_A                     2058 non-null
float64
 21  Net_profit_before_tax_to_Paid_in_capital               2058 non-null
float64
 22  Total_Asset_Turnover                                   2058 non-null
float64
 23  Accounts_Receivable_Turnover                           2058 non-null
float64
 24  Average_Collection_Days                                2058 non-null
float64
 25  Inventory_Turnover_Rate_times                          2058 non-null
float64
 26  Fixed_Assets_Turnover_Frequency                        2058 non-null
float64
 27  Net_Worth_Turnover_Rate_times                          2058 non-null
float64
 28  Operating_profit_per_person                            2058 non-null
float64
 29  Allocation_rate_per_person                             2058 non-null
float64
 30  Quick_Assets_to_Total_Assets                           2058 non-null
float64
 31  Cash_to_Total_Assets                                   1962 non-null
float64
 32  Quick_Assets_to_Current_Liability                      2058 non-null
```

```
float64
 33  Cash_to_Current_Liability                        2058 non-null
float64
 34  Operating_Funds_to_Liability                     2058 non-null
float64
 35  Inventory_to_Working_Capital                     2058 non-null
float64
 36  Inventory_to_Current_Liability                   2058 non-null
float64
 37  Long_term_Liability_to_Current_Assets            2058 non-null
float64
 38  Retained_Earnings_to_Total_Assets                2058 non-null
float64
 39  Total_income_to_Total_expense                    2058 non-null
float64
 40  Total_expense_to_Assets                          2058 non-null
float64
 41  Current_Asset_Turnover_Rate                      2058 non-null
float64
 42  Quick_Asset_Turnover_Rate                        2058 non-null
float64
 43  Cash_Turnover_Rate                               2058 non-null
float64
 44  Fixed_Assets_to_Assets                           2058 non-null
float64
 45  Cash_Flow_to_Total_Assets                        2058 non-null
float64
 46  Cash_Flow_to_Liability                           2058 non-null
float64
 47  CFO_to_Assets                                    2058 non-null
float64
 48  Cash_Flow_to_Equity                              2058 non-null
float64
 49  Current_Liability_to_Current_Assets              2044 non-null
float64
 50  Liability_Assets_Flag                            2058 non-null
int64
 51  Total_assets_to_GNP_price                        2058 non-null
float64
 52  No_credit_Interval                               2058 non-null
float64
 53  Degree_of_Financial_Leverage_DFL                 2058 non-null
float64
 54  Interest_Coverage_Ratio_Interest_expense_to_EBIT  2058 non-null
float64
 55  Net_Income_Flag                                  2058 non-null
int64
 56  Equity_to_Liability                              2058 non-null
float64
```

```
 57  Default                                           2058 non-null  
int64
dtypes: float64(53), int64(4), object(1)
memory usage: 932.7+ KB
```

```python
# checking for duplicate values
df.duplicated().sum() ##  Complete the code to check duplicate entries in the data
```

```
np.int64(0)
```

```python
df.nunique() ## complete the code to check unique entries in the data
```

```
Co_Code                                             2058
Co_Name                                             2058
Operating_Expense_Rate                              1495
Research_and_development_expense_rate                629
Cash_flow_rate                                      1888
Interest_bearing_debt_interest_rate                  813
Tax_rate_A                                           985
Cash_Flow_Per_Share                                  900
Per_Share_Net_profit_before_tax_Yuan_                876
Realized_Sales_Gross_Profit_Growth_Rate             1939
Operating_Profit_Growth_Rate                        2015
Continuous_Net_Profit_Growth_Rate                   2014
Total_Asset_Growth_Rate                              922
Net_Value_Growth_Rate                               1757
Total_Asset_Return_Growth_Rate_Ratio                1428
Cash_Reinvestment_perc                              1690
Current_Ratio                                       1972
Quick_Ratio                                         1970
Interest_Expense_Ratio                              1716
Total_debt_to_Total_net_worth                       1949
Long_term_fund_suitability_ratio_A                  2014
Net_profit_before_tax_to_Paid_in_capital            1798
Total_Asset_Turnover                                 283
Accounts_Receivable_Turnover                        1109
Average_Collection_Days                             1935
Inventory_Turnover_Rate_times                       1151
Fixed_Assets_Turnover_Frequency                     1079
Net_Worth_Turnover_Rate_times                        529
Operating_profit_per_person                         1484
Allocation_rate_per_person                          2051
Quick_Assets_to_Total_Assets                        2058
Cash_to_Total_Assets                                1962
Quick_Assets_to_Current_Liability                   2058
Cash_to_Current_Liability                           2056
Operating_Funds_to_Liability                        2058
Inventory_to_Working_Capital                        1931
Inventory_to_Current_Liability                      1932
```

```
Long_term_Liability_to_Current_Assets                1398
Retained_Earnings_to_Total_Assets                    2058
Total_income_to_Total_expense                        2056
Total_expense_to_Assets                              2058
Current_Asset_Turnover_Rate                          1973
Quick_Asset_Turnover_Rate                            1743
Cash_Turnover_Rate                                   1440
Fixed_Assets_to_Assets                               2054
Cash_Flow_to_Total_Assets                            2058
Cash_Flow_to_Liability                               2058
CFO_to_Assets                                        2058
Cash_Flow_to_Equity                                  2058
Current_Liability_to_Current_Assets                  2044
Liability_Assets_Flag                                   2
Total_assets_to_GNP_price                            2058
No_credit_Interval                                   2057
Degree_of_Financial_Leverage_DFL                     1940
Interest_Coverage_Ratio_Interest_expense_to_EBIT     1945
Net_Income_Flag                                         1
Equity_to_Liability                                  2058
Default                                                 2
dtype: int64
```

```
df.describe().T
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 57,\n \"fields\": [\n   {\n   \"column\": \"count\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 25.419352543077746,\n   \"min\": 1891.0,\n   \"max\": 2058.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   1891.0,\n   2044.0,\n   2037.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"mean\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 929954049.0062642,\n   \"min\": 0.0023579771039844515,\n   \"max\": 5287663257.045698,\n   \"num_unique_values\": 57,\n   \"samples\": [\n   17572.113216715257,\n   0.11477699404324587,\n   0.07993674991437308\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"std\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1005020809.4437544,\n   \"min\": 0.0,\n   \"max\": 3453544121.673858,\n   \"num_unique_values\": 57,\n   \"samples\": [\n   21892.886518349056,\n   0.15244565360109494,\n   0.09862259512732567\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"min\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.5534476428333828,\n   \"min\": 0.0,\n   \"max\": 4.0,\n   \"num_unique_values\": 22,\n   \"samples\": [\n   4.0,\n   0.026274053,\n   0.525126368\n   ],\n   \"semantic_type\": \"\",\n

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"25%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 571535703.3601657,\n        \"min\": 0.0,\n        \"max\":
4315000000.0,\n        \"num_unique_values\": 53,\n
\"samples\": [\n          0.16586230275,\n          0.09650577175,\n
0.0009124052499999999\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"50%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 851766436.0701509,\n        \"min\": 0.0,\n        \"max\":
6225000000.0,\n        \"num_unique_values\": 56,\n
\"samples\": [\n          6240.0,\n          0.0370988965,\n
0.345025692\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"75%\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1508198718.0011814,\n        \"min\": 0.0,\n        \"max\":
7220000000.0,\n        \"num_unique_values\": 56,\n
\"samples\": [\n          24280.75,\n          0.21619090975,\n
0.354140153\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"max\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 4560205596.464743,\n        \"min\": 0.01028413,\n
\"max\": 10000000000.0,\n        \"num_unique_values\": 37,\n
\"samples\": [\n          8800000000.0,\n          9940000000.0,\n
0.999696326\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

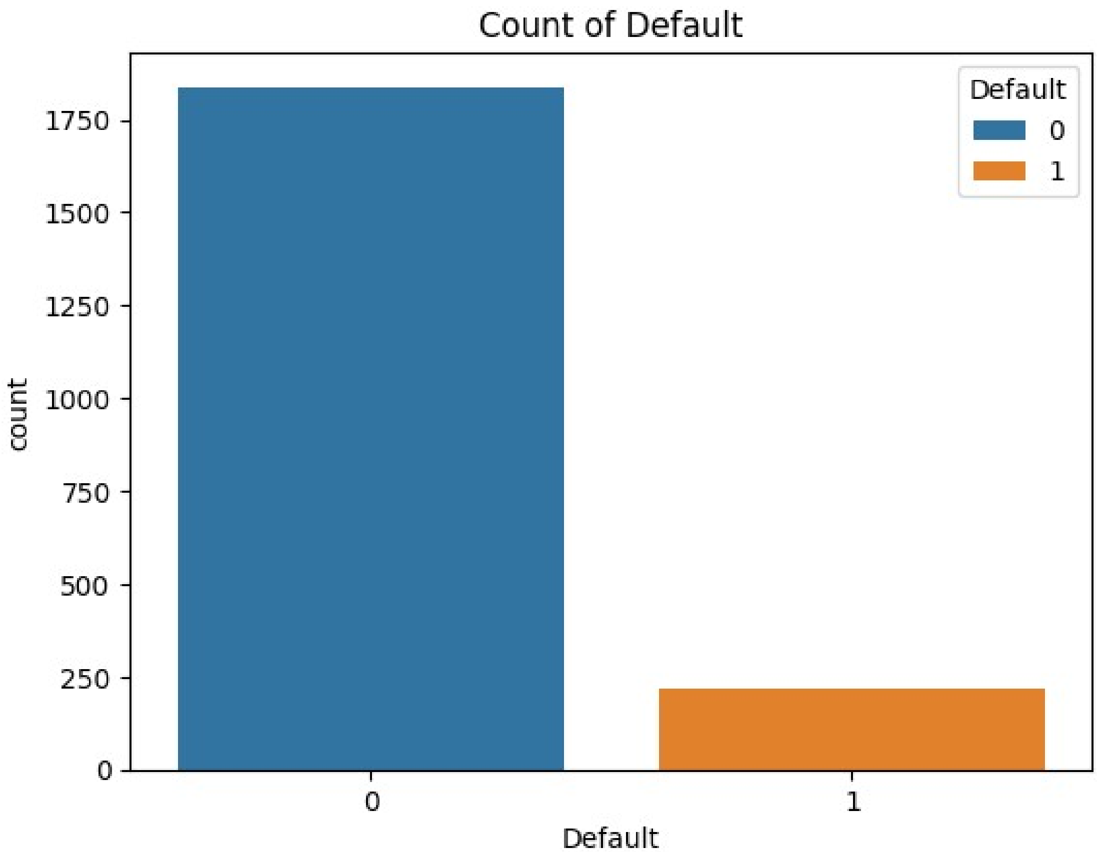

- We can see that `Co_Code` and `Co_Name` are not relevant for this exercise
- So we will drop these variables

```
df.drop(['Co_Code', 'Co_Name'], axis = 1, inplace = True)
```

#Exploratory Data Analysis

Univariate Analysis

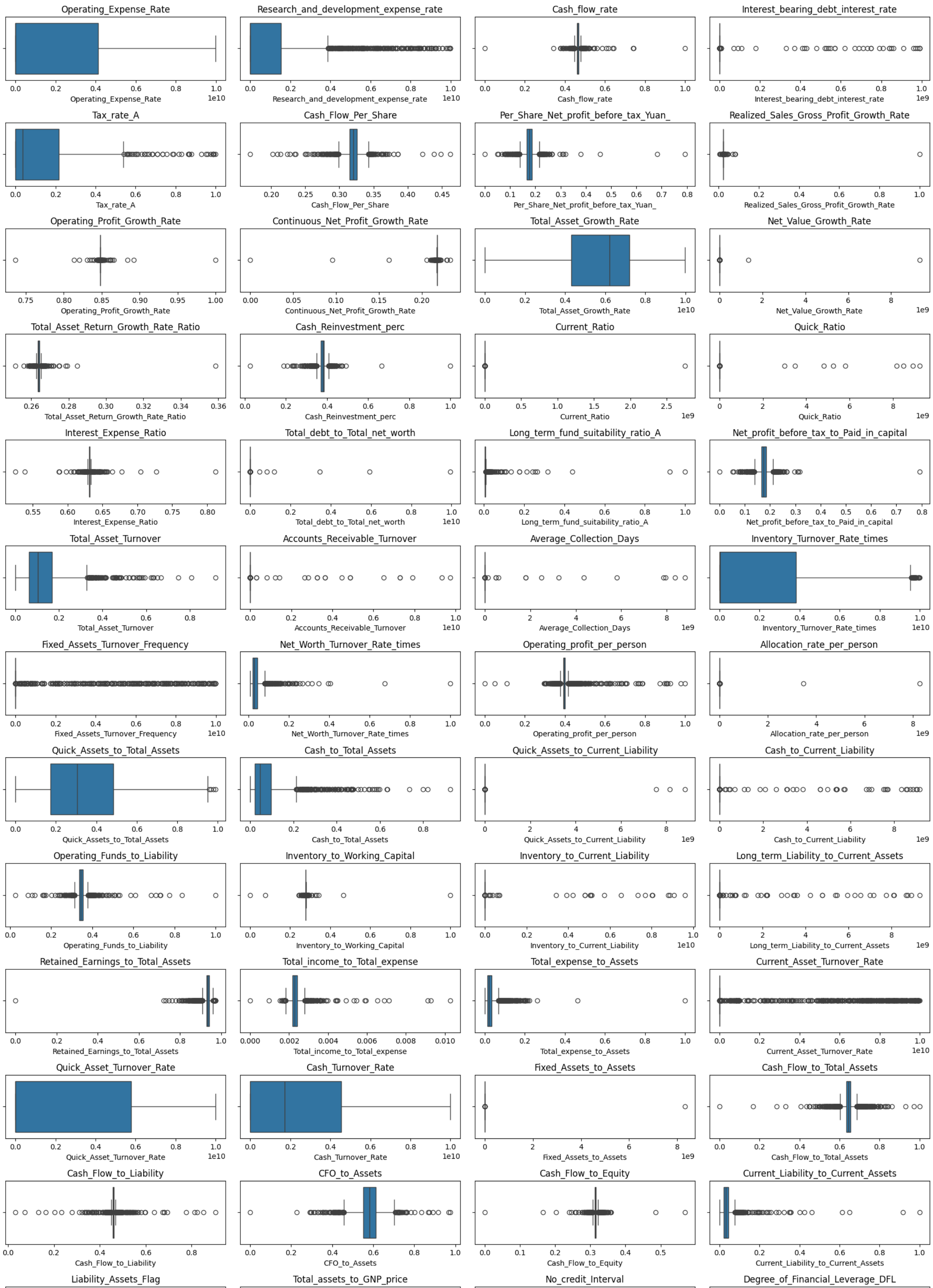
```
df["Default"].value_counts()    ## Complete the code to check unique
values in the mentioned column

Default
0    1838
1     220
Name: count, dtype: int64

sns.countplot(x = "Default", data = df, hue = 'Default') ## complete
the code to get a countplot of the mentioned column.
plt.title('Count of Default')
plt.show()
```




```
#Percentage of defaulters
(df.Default.sum()/len(df)) * 100
```

```
np.float64(10.689990281827017)
```

```
#Get boxplots for all the numerical columns
numeric_columns = df.select_dtypes(include=np.number).columns.tolist()

plt.figure(figsize=(18, 30))

for i, variable in enumerate(numeric_columns):
    plt.subplot(15, 4, i + 1)
    sns.boxplot(data=df, x=variable)  ## Complete the code to get
boxplots for all numerical columns
    plt.tight_layout()
    plt.title(variable)
```

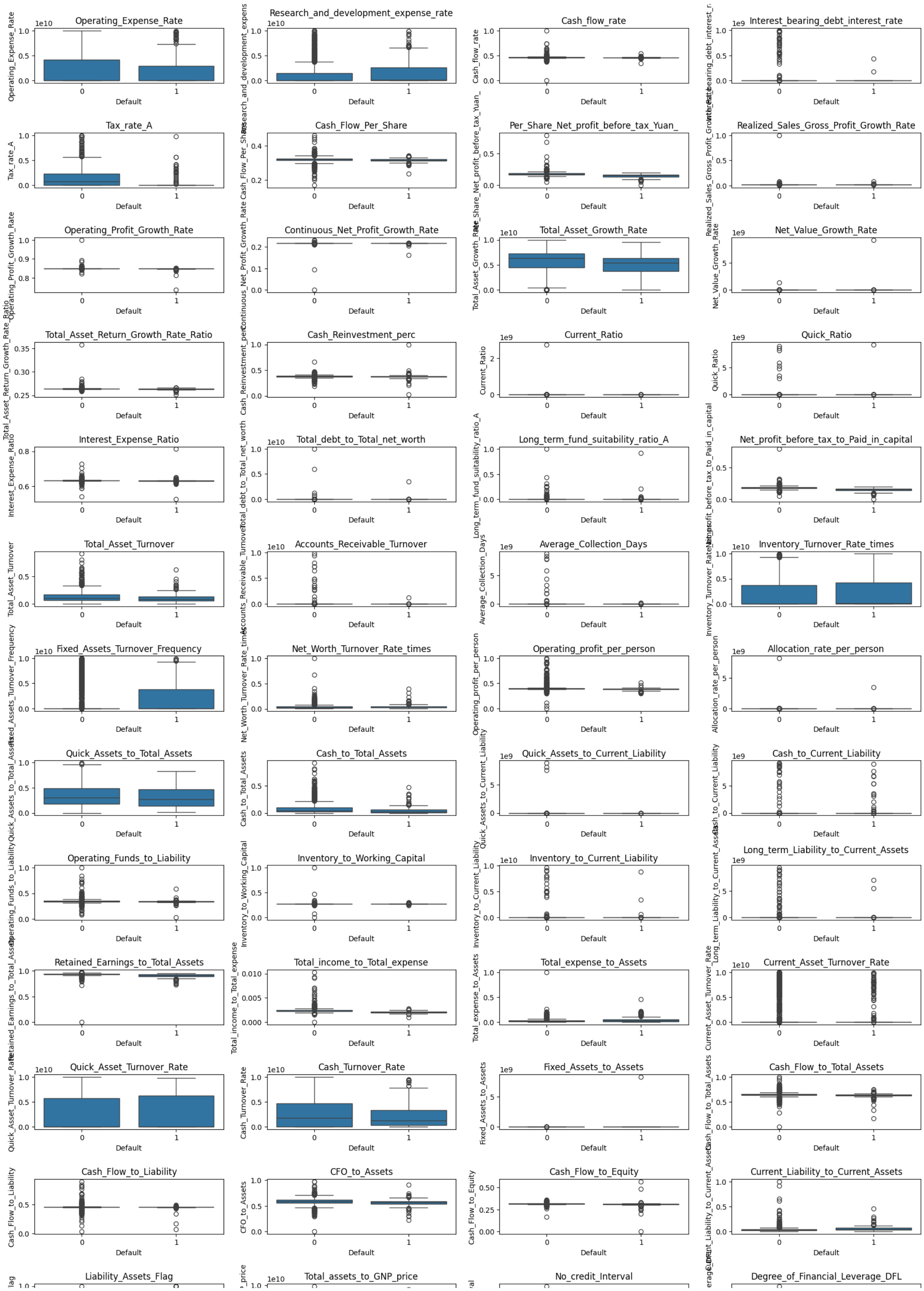

Operating_Expense_Rate
Research_and_development_expense_rate
Cash_flow_rate
Interest_bearing_debt_interest_rate
Tax_rate_A
Cash_Flow_Per_Share
Per_Share_Net_profit_before_tax_Yuan_
Realized_Sales_Gross_Profit_Growth_Rate
Operating_Profit_Growth_Rate
Continuous_Net_Profit_Growth_Rate
Total_Asset_Growth_Rate
Net_Value_Growth_Rate
Total_Asset_Return_Growth_Rate_Ratio
Cash_Reinvestment_perc
Current_Ratio
Quick_Ratio
Interest_Expense_Ratio
Total_debt_to_Total_net_worth
Long_term_fund_suitability_ratio_A
Net_profit_before_tax_to_Paid_in_capital
Total_Asset_Turnover
Accounts_Receivable_Turnover
Average_Collection_Days
Inventory_Turnover_Rate_times
Fixed_Assets_Turnover_Frequency
Net_Worth_Turnover_Rate_times
Operating_profit_per_person
Allocation_rate_per_person
Quick_Assets_to_Total_Assets
Cash_to_Total_Assets
Quick_Assets_to_Current_Liability
Cash_to_Current_Liability
Operating_Funds_to_Liability
Inventory_to_Working_Capital
Inventory_to_Current_Liability
Long_term_Liability_to_Current_Assets
Retained_Earnings_to_Total_Assets
Total_income_to_Total_expense
Total_expense_to_Assets
Current_Asset_Turnover_Rate
Quick_Asset_Turnover_Rate
Cash_Turnover_Rate
Fixed_Assets_to_Assets
Cash_Flow_to_Total_Assets
Cash_Flow_to_Liability
CFO_to_Assets
Cash_Flow_to_Equity
Current_Liability_to_Current_Assets
Liability_Assets_Flag
Total_assets_to_GNP_price
No_credit_Interval
Degree_of_Financial_Leverage_DFL

```python
#Get distplot for all the numerical columns
numeric_columns = df.select_dtypes(include=np.number).columns.tolist()

plt.figure(figsize=(18, 30))

for i, variable in enumerate(numeric_columns):
    plt.subplot(15, 4, i + 1)
    sns.histplot(data=df, x=variable)  ## Complete the code to get histplot for all numerical columns in the data
    plt.tight_layout()
    plt.title(variable)
```

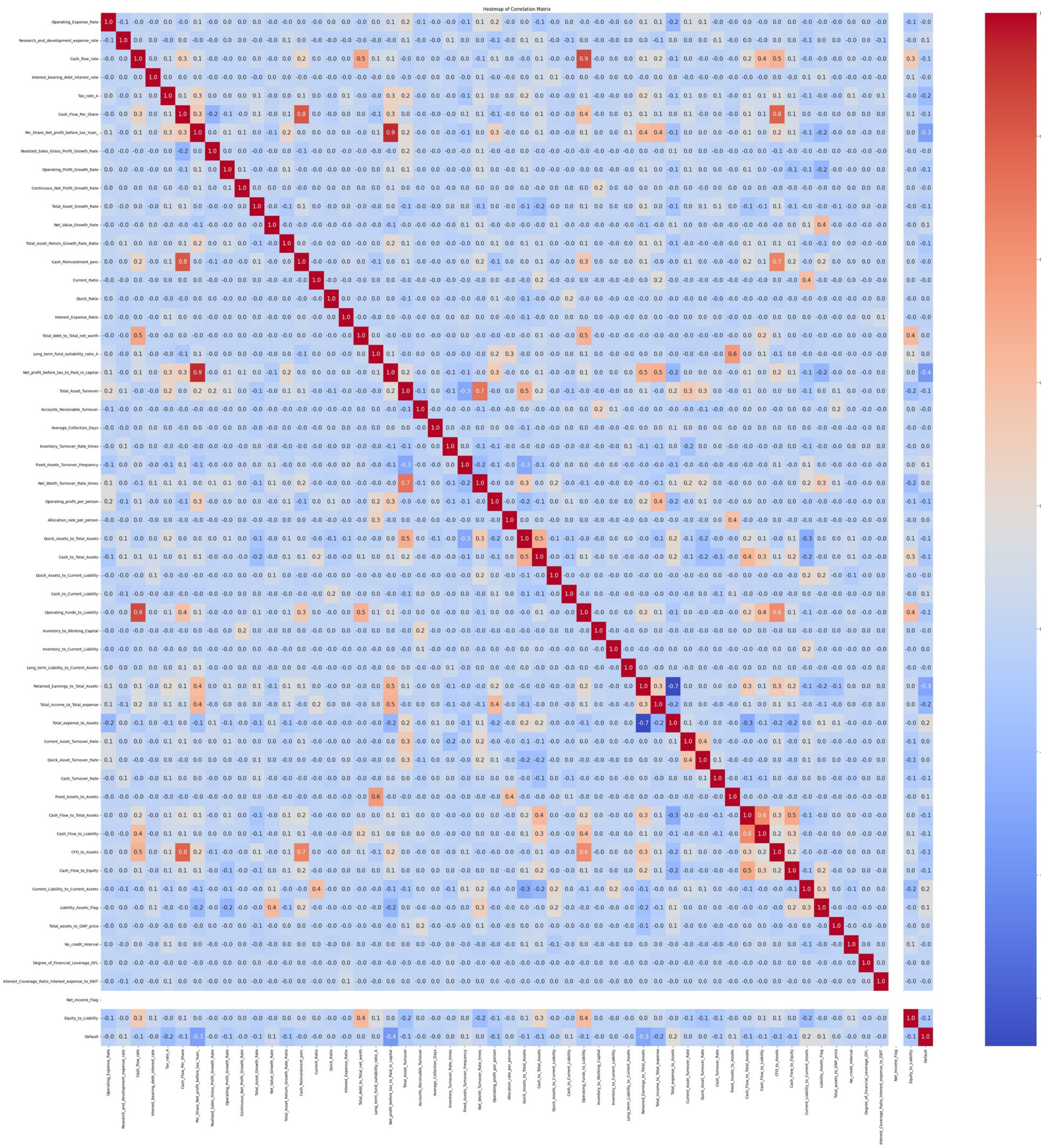

```python
#Get distplot for all the numerical columns
numeric_columns = df.select_dtypes(include=np.number).columns.tolist()

plt.figure(figsize=(18, 30))

for i, variable in enumerate(numeric_columns):
    plt.subplot(15, 4, i + 1)
    sns.histplot(data=df, x=variable)  ## Complete the code to get histplot for all numerical columns in the data
    plt.tight_layout()
    plt.title(variable)
```


Bivariate Analysis

```python
#Get boxplots for all the numerical columns
numeric_columns = df.select_dtypes(include=np.number).columns.tolist()

plt.figure(figsize=(18, 30))

for i, variable in enumerate(numeric_columns):
    plt.subplot(15, 4, i + 1)
    sns.boxplot(x="Default", y=variable, data=df)  ## Complete the code to get boxplot of all variables with Default column in the data
    plt.tight_layout()
    plt.title(variable)
```


Operating_Expense_Rate
Research_and_development_expense_rate
Cash_flow_rate
Interest_bearing_debt_interest_rate
Tax_rate_A
Cash_Flow_Per_Share
Per_Share_Net_profit_before_tax_Yuan_
Realized_Sales_Gross_Profit_Growth_Rate
Operating_Profit_Growth_Rate
Continuous_Net_Profit_Growth_Rate
Total_Asset_Growth_Rate
Net_Value_Growth_Rate
Total_Asset_Return_Growth_Rate_Ratio
Cash_Reinvestment_perc
Current_Ratio
Quick_Ratio
Interest_Expense_Ratio
Total_debt_to_Total_net_worth
Long_term_fund_suitability_ratio_A
Net_profit_before_tax_to_Paid_in_capital
Total_Asset_Turnover
Accounts_Receivable_Turnover
Average_Collection_Days
Inventory_Turnover_Rate_times
Fixed_Assets_Turnover_Frequency
Net_Worth_Turnover_Rate_times
Operating_profit_per_person
Allocation_rate_per_person
Quick_Assets_to_Total_Assets
Cash_to_Total_Assets
Quick_Assets_to_Current_Liability
Cash_to_Current_Liability
Operating_Funds_to_Liability
Inventory_to_Working_Capital
Inventory_to_Current_Liability
Long_term_Liability_to_Current_Assets
Retained_Earnings_to_Total_Assets
Total_income_to_Total_expense
Total_expense_to_Assets
Current_Asset_Turnover_Rate
Quick_Asset_Turnover_Rate
Cash_Turnover_Rate
Fixed_Assets_to_Assets
Cash_Flow_to_Total_Assets
Cash_Flow_to_Liability
CFO_to_Assets
Cash_Flow_to_Equity
Current_Liability_to_Current_Assets
Liability_Assets_Flag
Total_assets_to_GNP_price
No_credit_Interval
Degree_of_Financial_Leverage_DFL
Default

```python
# Calculate the correlation matrix
corr_matrix = df.corr(numeric_only=True)  ## Complete the code to get
the correlation matrix for the data

# Create a heatmap of the correlation matrix
plt.figure(figsize=(50, 50))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".1f",
annot_kws={"size": 15})
plt.title('Heatmap of Correlation Matrix')
plt.show()
```



Data Preprocessing

Dropping columns with few unique values

```
df.nunique()
```

Operating_Expense_Rate	1495
Research_and_development_expense_rate	629
Cash_flow_rate	1888
Interest_bearing_debt_interest_rate	813
Tax_rate_A	985
Cash_Flow_Per_Share	900
Per_Share_Net_profit_before_tax_Yuan_	876
Realized_Sales_Gross_Profit_Growth_Rate	1939
Operating_Profit_Growth_Rate	2015
Continuous_Net_Profit_Growth_Rate	2014
Total_Asset_Growth_Rate	922
Net_Value_Growth_Rate	1757
Total_Asset_Return_Growth_Rate_Ratio	1428
Cash_Reinvestment_perc	1690
Current_Ratio	1972
Quick_Ratio	1970
Interest_Expense_Ratio	1716
Total_debt_to_Total_net_worth	1949
Long_term_fund_suitability_ratio_A	2014
Net_profit_before_tax_to_Paid_in_capital	1798
Total_Asset_Turnover	283
Accounts_Receivable_Turnover	1109
Average_Collection_Days	1935
Inventory_Turnover_Rate_times	1151
Fixed_Assets_Turnover_Frequency	1079
Net_Worth_Turnover_Rate_times	529
Operating_profit_per_person	1484
Allocation_rate_per_person	2051
Quick_Assets_to_Total_Assets	2058
Cash_to_Total_Assets	1962
Quick_Assets_to_Current_Liability	2058
Cash_to_Current_Liability	2056
Operating_Funds_to_Liability	2058
Inventory_to_Working_Capital	1931
Inventory_to_Current_Liability	1932
Long_term_Liability_to_Current_Assets	1398
Retained_Earnings_to_Total_Assets	2058
Total_income_to_Total_expense	2056
Total_expense_to_Assets	2058
Current_Asset_Turnover_Rate	1973
Quick_Asset_Turnover_Rate	1743
Cash_Turnover_Rate	1440
Fixed_Assets_to_Assets	2054
Cash_Flow_to_Total_Assets	2058
Cash_Flow_to_Liability	2058
CFO_to_Assets	2058
Cash_Flow_to_Equity	2058
Current_Liability_to_Current_Assets	2044
Liability_Assets_Flag	2
Total_assets_to_GNP_price	2058

```
No_credit_Interval                                          2057
Degree_of_Financial_Leverage_DFL                            1940
Interest_Coverage_Ratio_Interest_expense_to_EBIT            1945
Net_Income_Flag                                                1
Equity_to_Liability                                         2058
Default                                                        2
dtype: int64
```

We can drop the columns `Net_Income_Flag` and `Liability_Assets_Flag` as they have very few unique values.

```
df.drop(['Net_Income_Flag', 'Liability_Assets_Flag'], axis = 1,
inplace = True)  ## Complete the code to drop the mentioned columns
from the dataset
df.nunique()
```

```
Operating_Expense_Rate                                      1495
Research_and_development_expense_rate                        629
Cash_flow_rate                                              1888
Interest_bearing_debt_interest_rate                          813
Tax_rate_A                                                   985
Cash_Flow_Per_Share                                          900
Per_Share_Net_profit_before_tax_Yuan_                        876
Realized_Sales_Gross_Profit_Growth_Rate                     1939
Operating_Profit_Growth_Rate                                2015
Continuous_Net_Profit_Growth_Rate                           2014
Total_Asset_Growth_Rate                                      922
Net_Value_Growth_Rate                                       1757
Total_Asset_Return_Growth_Rate_Ratio                        1428
Cash_Reinvestment_perc                                      1690
Current_Ratio                                               1972
Quick_Ratio                                                 1970
Interest_Expense_Ratio                                      1716
Total_debt_to_Total_net_worth                               1949
Long_term_fund_suitability_ratio_A                          2014
Net_profit_before_tax_to_Paid_in_capital                    1798
Total_Asset_Turnover                                         283
Accounts_Receivable_Turnover                                1109
Average_Collection_Days                                     1935
Inventory_Turnover_Rate_times                               1151
Fixed_Assets_Turnover_Frequency                             1079
Net_Worth_Turnover_Rate_times                                529
Operating_profit_per_person                                 1484
Allocation_rate_per_person                                  2051
Quick_Assets_to_Total_Assets                                2058
Cash_to_Total_Assets                                        1962
Quick_Assets_to_Current_Liability                           2058
Cash_to_Current_Liability                                   2056
Operating_Funds_to_Liability                                2058
```

```
Inventory_to_Working_Capital                            1931
Inventory_to_Current_Liability                          1932
Long_term_Liability_to_Current_Assets                   1398
Retained_Earnings_to_Total_Assets                       2058
Total_income_to_Total_expense                           2056
Total_expense_to_Assets                                 2058
Current_Asset_Turnover_Rate                             1973
Quick_Asset_Turnover_Rate                               1743
Cash_Turnover_Rate                                      1440
Fixed_Assets_to_Assets                                  2054
Cash_Flow_to_Total_Assets                               2058
Cash_Flow_to_Liability                                  2058
CFO_to_Assets                                           2058
Cash_Flow_to_Equity                                     2058
Current_Liability_to_Current_Assets                     2044
Total_assets_to_GNP_price                               2058
No_credit_Interval                                      2057
Degree_of_Financial_Leverage_DFL                        1940
Interest_Coverage_Ratio_Interest_expense_to_EBIT        1945
Equity_to_Liability                                     2058
Default                                                    2
dtype: int64
```

Outliers Check

```
outliers_count = {}

# Iterate over each column in the DataFrame
for column in df.columns:
    # Check if the column is numeric before processing
    if pd.api.types.is_numeric_dtype(df[column]):
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)

        IQR = Q3 - Q1

        lower_bound = Q1 - 1.5 * IQR  ## Fill the blank with correct
value for getting lower_bound
        upper_bound = Q3 + 1.5 * IQR  ## Fill the blank with correct
value for getting upper_bound

        outliers = df[(df[column] < lower_bound) | (df[column] >
upper_bound)]
        outliers_count[column] = len(outliers)

print("Number of outliers in each column:")
pd.DataFrame([{'Column': column, 'No. of outliers': outliers} for
column, outliers in outliers_count.items()])
```

```
Number of outliers in each column:
```

```
{"summary":"{\n  \"name\": \"pd\",\n  \"rows\": 54,\n  \"fields\": [\n
{\n      \"column\": \"Column\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 54,\n
\"samples\": [\n
\"Net_profit_before_tax_to_Paid_in_capital\",\n
\"No_credit_Interval\",\n          \"Total_assets_to_GNP_price\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"No. of outliers\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
125,\n        \"min\": 0,\n        \"max\": 501,\n
\"num_unique_values\": 48,\n        \"samples\": [\n          4,\n
407,\n          200\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


Data Preparation for Modeling

```
# Seperating target variable from the rest of the data
df_X = df.drop(['Default'], axis = 1)
df_y = df['Default']

#Splitting the data for training and testing
X_train, X_test, y_train, y_test = train_test_split(df_X, df_y,
test_size=0.25, random_state=42, stratify = df_y)  ## Complete the
code to split the data into train and test in the ratio 75:25
```

Missing Values Detection and Treatment

```
# Check missing values
X_train.isnull().sum()  ## Complete the code to get the number of null
or NaN values in each column
```

```
Operating_Expense_Rate                           0
Research_and_development_expense_rate            0
Cash_flow_rate                                   0
Interest_bearing_debt_interest_rate              0
Tax_rate_A                                       0
Cash_Flow_Per_Share                            126
Per_Share_Net_profit_before_tax_Yuan_            0
Realized_Sales_Gross_Profit_Growth_Rate          0
Operating_Profit_Growth_Rate                     0
Continuous_Net_Profit_Growth_Rate                0
Total_Asset_Growth_Rate                          0
Net_Value_Growth_Rate                            0
Total_Asset_Return_Growth_Rate_Ratio             0
Cash_Reinvestment_perc                           0
Current_Ratio                                    0
Quick_Ratio                                      0
Interest_Expense_Ratio                           0
Total_debt_to_Total_net_worth                   18
```

```
Long_term_fund_suitability_ratio_A                      0
Net_profit_before_tax_to_Paid_in_capital                0
Total_Asset_Turnover                                    0
Accounts_Receivable_Turnover                            0
Average_Collection_Days                                 0
Inventory_Turnover_Rate_times                           0
Fixed_Assets_Turnover_Frequency                         0
Net_Worth_Turnover_Rate_times                           0
Operating_profit_per_person                             0
Allocation_rate_per_person                              0
Quick_Assets_to_Total_Assets                            0
Cash_to_Total_Assets                                   71
Quick_Assets_to_Current_Liability                       0
Cash_to_Current_Liability                               0
Operating_Funds_to_Liability                            0
Inventory_to_Working_Capital                            0
Inventory_to_Current_Liability                          0
Long_term_Liability_to_Current_Assets                   0
Retained_Earnings_to_Total_Assets                       0
Total_income_to_Total_expense                           0
Total_expense_to_Assets                                 0
Current_Asset_Turnover_Rate                             0
Quick_Asset_Turnover_Rate                               0
Cash_Turnover_Rate                                      0
Fixed_Assets_to_Assets                                  0
Cash_Flow_to_Total_Assets                               0
Cash_Flow_to_Liability                                  0
CFO_to_Assets                                           0
Cash_Flow_to_Equity                                     0
Current_Liability_to_Current_Assets                    11
Total_assets_to_GNP_price                               0
No_credit_Interval                                      0
Degree_of_Financial_Leverage_DFL                        0
Interest_Coverage_Ratio_Interest_expense_to_EBIT        0
Equity_to_Liability                                     0
dtype: int64

# Check missing values
X_test.isnull().sum()

Operating_Expense_Rate                                 0
Research_and_development_expense_rate                  0
Cash_flow_rate                                         0
Interest_bearing_debt_interest_rate                    0
Tax_rate_A                                             0
Cash_Flow_Per_Share                                   41
Per_Share_Net_profit_before_tax_Yuan_                  0
Realized_Sales_Gross_Profit_Growth_Rate                0
Operating_Profit_Growth_Rate                           0
Continuous_Net_Profit_Growth_Rate                      0
```

```
Total_Asset_Growth_Rate                                      0
Net_Value_Growth_Rate                                        0
Total_Asset_Return_Growth_Rate_Ratio                         0
Cash_Reinvestment_perc                                       0
Current_Ratio                                                0
Quick_Ratio                                                  0
Interest_Expense_Ratio                                       0
Total_debt_to_Total_net_worth                                3
Long_term_fund_suitability_ratio_A                           0
Net_profit_before_tax_to_Paid_in_capital                     0
Total_Asset_Turnover                                         0
Accounts_Receivable_Turnover                                 0
Average_Collection_Days                                      0
Inventory_Turnover_Rate_times                                0
Fixed_Assets_Turnover_Frequency                              0
Net_Worth_Turnover_Rate_times                                0
Operating_profit_per_person                                  0
Allocation_rate_per_person                                   0
Quick_Assets_to_Total_Assets                                 0
Cash_to_Total_Assets                                        25
Quick_Assets_to_Current_Liability                            0
Cash_to_Current_Liability                                    0
Operating_Funds_to_Liability                                 0
Inventory_to_Working_Capital                                 0
Inventory_to_Current_Liability                               0
Long_term_Liability_to_Current_Assets                        0
Retained_Earnings_to_Total_Assets                            0
Total_income_to_Total_expense                                0
Total_expense_to_Assets                                      0
Current_Asset_Turnover_Rate                                  0
Quick_Asset_Turnover_Rate                                    0
Cash_Turnover_Rate                                           0
Fixed_Assets_to_Assets                                       0
Cash_Flow_to_Total_Assets                                    0
Cash_Flow_to_Liability                                       0
CFO_to_Assets                                                0
Cash_Flow_to_Equity                                          0
Current_Liability_to_Current_Assets                          3
Total_assets_to_GNP_price                                    0
No_credit_Interval                                           0
Degree_of_Financial_Leverage_DFL                             0
Interest_Coverage_Ratio_Interest_expense_to_EBIT             0
Equity_to_Liability                                          0
dtype: int64

# Drop the non-numeric 'Co_Name' column before imputation
X_train = X_train.drop('Co_Name', axis=1, errors='ignore')
X_test = X_test.drop('Co_Name', axis=1, errors='ignore')

#Replace the missing values in the data using KNN Imputer
```

```python
KNNimputerModel = KNNImputer(n_neighbors=5)  ## Complete the code to select 5 neighbors for KNN Imputer

X_train = pd.DataFrame(KNNimputerModel.fit_transform(X_train), columns = X_train.columns)
X_test = pd.DataFrame(KNNimputerModel.fit_transform(X_test), columns = X_test.columns)  ## Complete the code to replace missing values in X_test

print(X_train.isnull().sum().sum())
print(X_test.isnull().sum().sum())
```

```
0
0
```

Scaling the Data

```python
#Scaling of features is done to bring all the features to the same scale.
sc = StandardScaler()

X_train_scaled = pd.DataFrame(sc.fit_transform(X_train), columns=X_train.columns)
X_test_scaled = pd.DataFrame(sc.transform(X_test), columns=X_test.columns)  ## Complete the code to scale X_test to the same scale as X_train

X_train_scaled.head()
```

{"type":"dataframe","variable_name":"X_train_scaled"}

```python
X_test_scaled.head()
```

{"type":"dataframe","variable_name":"X_test_scaled"}

#Model Building

Model Evaluation Criterion

Metric of Choice

```python
# defining a function to compute different metrics to check performance of a classification model built using sklearn


def model_performance_classification(model, predictors, target, threshold = 0.5):
    """
    Function to compute different metrics to check classification model performance
```

```
    model: classifier
    predictors: independent variables
    target: dependent variable
    """

    # predicting using the independent variables
    y_pred = model.predict(predictors)

    if len(list(set(y_pred))) != 2:
        y_prob_pred = model.predict(predictors)

        y_pred=[]
        for i in range(0,len(y_prob_pred)):
            if np.array(y_prob_pred)[i] > threshold:
                a=1
            else:
                a=0
            y_pred.append(a)
    else:
        pass

    acc = accuracy_score(target, y_pred)  # to compute Accuracy
    recall = recall_score(target, y_pred)  # to compute Recall
    precision = precision_score(target, y_pred)  # to compute
Precision
    f1 = f1_score(target, y_pred)  # to compute F1-score

    # creating a dataframe of metrics
    df_perf = pd.DataFrame(
        {"Accuracy": acc, "Recall": recall, "Precision": precision,
"F1": f1,},
        index=[0],
    )

    return df_perf

def model_confusion_matrix(model, predictors, target, threshold =
0.5):
    """
    To plot the confusion_matrix with percentages

    model: classifier
    predictors: independent variables
    target: dependent variable
    """
    y_pred = model.predict(predictors)
    if len(list(set(y_pred))) != 2:
        y_prob_pred = model.predict(predictors)
```

```
        y_pred=[]
        for i in range(0,len(y_prob_pred)):
            if np.array(y_prob_pred)[i] > threshold:
                a=1
            else:
                a=0
            y_pred.append(a)
    else:
        pass

    cm = confusion_matrix(target, y_pred)
    labels = np.asarray(
        [
            ["{0:0.0f}".format(item) + "\n{0:.2%}".format(item /
cm.flatten().sum())]
            for item in cm.flatten()
        ]
    ).reshape(2, 2)

    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=labels, fmt="")
    plt.ylabel("True label")
    plt.xlabel("Predicted label")
```

Logistic Regression

```
# Adding constant to data for Logistic Regression
X_train_with_intercept = SM.add_constant(X_train_scaled)
X_test_with_intercept = SM.add_constant(X_test_scaled)

X_train_with_intercept.head()
```

{"type":"dataframe","variable_name":"X_train_with_intercept"}

```
y_train.reset_index(inplace = True, drop = True)

X_train_scaled = SM.add_constant(X_train_scaled)
X_test_scaled = SM.add_constant(X_test_scaled, has_constant='add')

LogisticReg = SM.Logit(y_train, X_train_scaled)   ## Complete the code
to define Logistic Regression Model
print(LogisticReg.fit().summary())
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.193946
         Iterations: 35
                           Logit Regression Results

======================================================================
========
Dep. Variable:                Default   No. Observations:
```

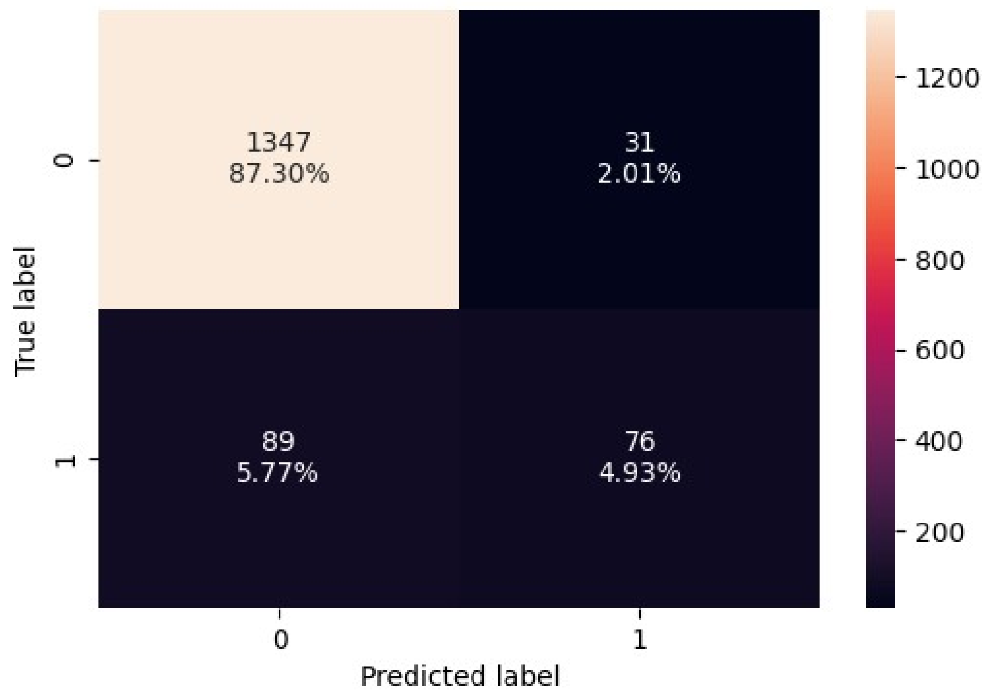
```
1543
Model:                          Logit   Df Residuals:
1489
Method:                           MLE   Df Model:
53
Date:                Sat, 05 Jul 2025   Pseudo R-squ.:
0.4297
Time:                        03:41:06   Log-Likelihood:
-299.26
converged:                      False   LL-Null:
-524.71
Covariance Type:            nonrobust   LLR p-value:
1.764e-64
======================================================================
==============================================
                                                       coef    std err
z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------
----------------------------------------------
const                                               -7.4685   2410.787
-0.003      0.998   -4732.525    4717.588
Operating_Expense_Rate                               0.2077      0.121
1.713      0.087      -0.030       0.445
Research_and_development_expense_rate                0.3556      0.104
3.433      0.001       0.153       0.559
Cash_flow_rate                                      -0.1837      1.016
-0.181      0.857      -2.175       1.808
Interest_bearing_debt_interest_rate                  0.1755      0.151
1.163      0.245      -0.120       0.471
Tax_rate_A                                          -0.2580      0.174
-1.481      0.139      -0.599       0.083
Cash_Flow_Per_Share                                 -0.3533      0.281
-1.260      0.208      -0.903       0.196
Per_Share_Net_profit_before_tax_Yuan_                0.2518      1.276
0.197      0.844      -2.249       2.752
Realized_Sales_Gross_Profit_Growth_Rate              0.1012      0.118
0.859      0.390      -0.130       0.332
Operating_Profit_Growth_Rate                        -0.1546      0.267
-0.579      0.563      -0.678       0.369
Continuous_Net_Profit_Growth_Rate                    0.1736      0.132
1.317      0.188      -0.085       0.432
Total_Asset_Growth_Rate                             -0.0640      0.131
-0.487      0.626      -0.321       0.193
Net_Value_Growth_Rate                                0.5177   3097.822
0.000      1.000   -6071.102    6072.138
Total_Asset_Return_Growth_Rate_Ratio                -0.3299      0.361
-0.915      0.360      -1.037       0.377
Cash_Reinvestment_perc                               0.1700      0.346
0.491      0.624      -0.509       0.849
```

```
Current_Ratio                                                  -1.6114      0.925
-1.742      0.081      -3.424       0.201
Quick_Ratio                                                    -2.7355   2.57e+04
-0.000      1.000   -5.05e+04    5.05e+04
Interest_Expense_Ratio                                          0.0197      0.065
0.303      0.762      -0.107       0.147
Total_debt_to_Total_net_worth                                   1.9035      0.623
3.058      0.002       0.683       3.124
Long_term_fund_suitability_ratio_A                              0.1675      0.223
0.751      0.452      -0.269       0.604
Net_profit_before_tax_to_Paid_in_capital                       -1.0834      1.179
-0.919      0.358      -3.394       1.227
Total_Asset_Turnover                                           -0.2122      0.319
-0.666      0.506      -0.837       0.413
Accounts_Receivable_Turnover                                   -1.0019      0.642
-1.560      0.119      -2.261       0.257
Average_Collection_Days                                       -15.1938   2.49e+04
-0.001      1.000   -4.89e+04    4.88e+04
Inventory_Turnover_Rate_times                                  -0.0490      0.117
-0.420      0.675      -0.278       0.180
Fixed_Assets_Turnover_Frequency                                 0.1775      0.106
1.678      0.093      -0.030       0.385
Net_Worth_Turnover_Rate_times                                  -0.2559      0.211
-1.212      0.225      -0.670       0.158
Operating_profit_per_person                                     0.0505      0.195
0.259      0.796      -0.331       0.432
Allocation_rate_per_person                                    -80.4893    153.634
-0.524      0.600    -381.606     220.627
Quick_Assets_to_Total_Assets                                    0.1935      0.189
1.024      0.306      -0.177       0.564
Cash_to_Total_Assets                                           -0.3059      0.222
-1.380      0.168      -0.740       0.129
Quick_Assets_to_Current_Liability                              -0.5860   1.49e+04
-3.92e-05      1.000   -2.93e+04    2.93e+04
Cash_to_Current_Liability                                       0.0684      0.076
0.905      0.365      -0.080       0.217
Operating_Funds_to_Liability                                    1.2409      0.783
1.584      0.113      -0.294       2.776
Inventory_to_Working_Capital                                   -0.1714      0.158
-1.088      0.276      -0.480       0.137
Inventory_to_Current_Liability                                  0.1022      0.117
0.870      0.384      -0.128       0.332
Long_term_Liability_to_Current_Assets                          -0.0208      0.107
-0.195      0.846      -0.230       0.188
Retained_Earnings_to_Total_Assets                              -0.2111      0.207
-1.019      0.308      -0.617       0.195
Total_income_to_Total_expense                                  -1.4219      0.437
-3.252      0.001      -2.279      -0.565
Total_expense_to_Assets                                         0.0849      0.253
```

```
0.335      0.738      -0.412       0.582
Current_Asset_Turnover_Rate                                   -0.0962      0.129
-0.746      0.456      -0.349       0.157
Quick_Asset_Turnover_Rate                                      0.0640      0.128
0.499      0.618      -0.188       0.316
Cash_Turnover_Rate                                            -0.4286      0.130
-3.307      0.001      -0.683      -0.175
Fixed_Assets_to_Assets                                        31.5359    195.727
0.161      0.872    -352.082     415.154
Cash_Flow_to_Total_Assets                                      0.9901      0.270
3.668      0.000       0.461       1.519
Cash_Flow_to_Liability                                        -2.7554      0.607
-4.542      0.000      -3.945      -1.566
CFO_to_Assets                                                 -0.3143      0.467
-0.673      0.501      -1.230       0.602
Cash_Flow_to_Equity                                           -0.0344      0.085
-0.404      0.686      -0.201       0.132
Current_Liability_to_Current_Assets                           -0.0863      0.121
-0.714      0.476      -0.323       0.151
Total_assets_to_GNP_price                                     -0.0290      0.076
-0.384      0.701      -0.177       0.119
No_credit_Interval                                             0.1051      0.079
1.326      0.185      -0.050       0.260
Degree_of_Financial_Leverage_DFL                               0.0729      0.056
1.303      0.193      -0.037       0.183
Interest_Coverage_Ratio_Interest_expense_to_EBIT      0.0677      0.087
0.778      0.437      -0.103       0.238
Equity_to_Liability                                           -3.0217      0.709
-4.260      0.000      -4.412      -1.632
======================================================================
==============================================
```

Logistic Regression Model - Training Performance

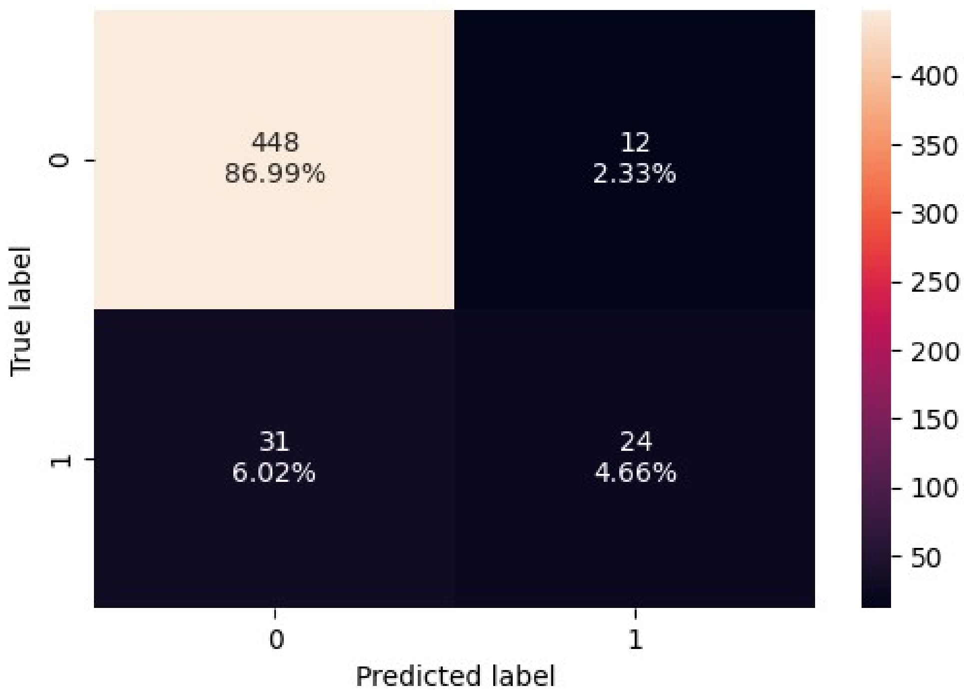
```
fitted_model = LogisticReg.fit()
model_confusion_matrix(fitted_model, X_train_with_intercept, y_train)

Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.193946
         Iterations: 35
```




```
# Fit the model
fitted_model = LogisticReg.fit()

# Use the fitted model, not the Logit class
logistic_regression_perf_train =
model_performance_classification(fitted_model, X_train_with_intercept,
y_train)
print(logistic_regression_perf_train)

Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.193946
         Iterations: 35
   Accuracy    Recall  Precision        F1
0  0.922229  0.460606    0.71028  0.558824
```

Logistic Regression Model - Test Performance

```
model_confusion_matrix(fitted_model, X_test_with_intercept, y_test)
## Complete the code to create confusion matrix for test data
```



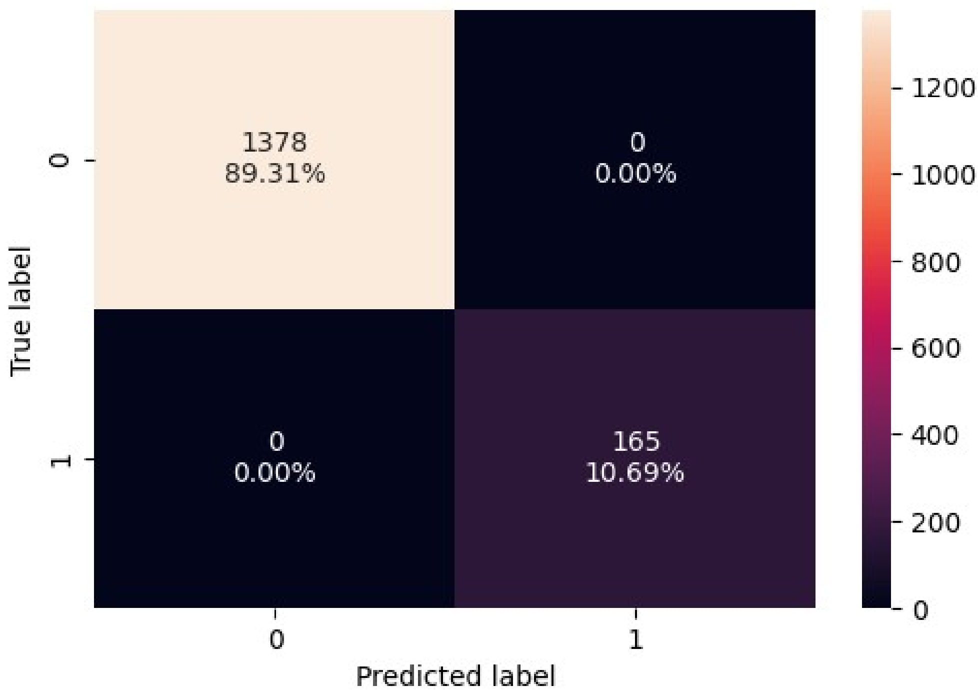

```
logistic_regression_perf_test =
model_performance_classification(fitted_model, X_test_with_intercept,
y_test)
print(logistic_regression_perf_test)

   Accuracy    Recall  Precision        F1
0  0.916505  0.436364   0.666667  0.527473
```

Random Forest

```
rf_classifier = RandomForestClassifier(random_state=42)  ## Complete
the code to define random forest with random state = 42
rf_model = rf_classifier.fit(X_train_scaled, y_train)    ## Complete
the code to fit random forest on the train data
```

Random Forest Model - Training Performance

```
rf_conf_matrix_train = model_confusion_matrix(rf_model,
X_train_scaled, y_train)
print(rf_conf_matrix_train)

None
```

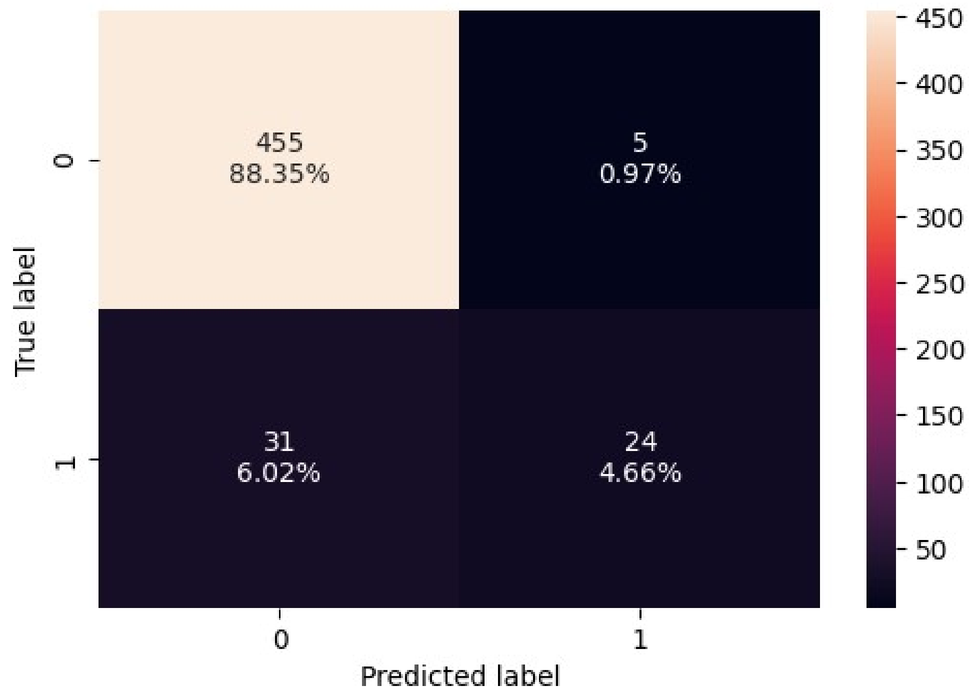



```
random_forest_perf_train = model_performance_classification(rf_model,
X_train_scaled, y_train)
random_forest_perf_train
```

{"summary":"{\n \"name\": \"random_forest_perf_train\",\n \"rows\": 1,\n \"fields\": [\n   {\n   \"column\": \"Accuracy\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 1.0,\n   \"max\": 1.0,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   1.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Recall\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 1.0,\n   \"max\": 1.0,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   1.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Precision\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 1.0,\n   \"max\": 1.0,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   1.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"F1\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 1.0,\n   \"max\": 1.0,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   1.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"random_forest_perf_train"}

Random Forest Model - Test Performance

```
model_confusion_matrix(rf_model, X_test_scaled, y_test) ## Complete
the code to create confusion matrix for test data
```




```
random_forest_perf_test = model_performance_classification(rf_model,
X_test_scaled, y_test)
print(random_forest_perf_test)

   Accuracy    Recall  Precision        F1
0  0.930097  0.436364   0.827586  0.571429
```

Model Performance Improvement

Model Performance Improvement - Logistic Regression

```
from statsmodels.stats.outliers_influence import
variance_inflation_factor
import pandas as pd

def calculate_vif(X):
    vif_data = pd.DataFrame()
    vif_data["feature"] = X.columns
    vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                       for i in range(X.shape[1])]
```

```
    return vif_data

vif_result = calculate_vif(X_train_scaled)
print(vif_result.sort_values("VIF", ascending=False))
```

```
                                     feature        VIF
33               Operating_Funds_to_Liability  12.536226
3                              Cash_flow_rate  12.259184
46                              CFO_to_Assets  10.987676
7       Per_Share_Net_profit_before_tax_Yuan_   8.756612
20     Net_profit_before_tax_to_Paid_in_capital   8.637685
14                      Cash_Reinvestment_perc   7.340538
21                        Total_Asset_Turnover   5.467530
15                               Current_Ratio   4.945713
53                         Equity_to_Liability   4.779776
6                          Cash_Flow_Per_Share   4.564430
26               Net_Worth_Turnover_Rate_times   3.945259
18              Total_debt_to_Total_net_worth   3.776391
39                     Total_expense_to_Assets   3.366230
37             Retained_Earnings_to_Total_Assets   3.365775
44                    Cash_Flow_to_Total_Assets   3.309496
45                      Cash_Flow_to_Liability   2.813638
29                Quick_Assets_to_Total_Assets   2.397607
30                        Cash_to_Total_Assets   2.183010
19           Long_term_fund_suitability_ratio_A   1.839945
43                      Fixed_Assets_to_Assets   1.815190
38              Total_income_to_Total_expense   1.676735
27                 Operating_profit_per_person   1.568575
10            Continuous_Net_Profit_Growth_Rate   1.467948
48          Current_Liability_to_Current_Assets   1.464422
34               Inventory_to_Working_Capital   1.459350
47                         Cash_Flow_to_Equity   1.425404
40                  Current_Asset_Turnover_Rate   1.416203
41                    Quick_Asset_Turnover_Rate   1.377544
1                       Operating_Expense_Rate   1.259611
5                                   Tax_rate_A   1.248346
25             Fixed_Assets_Turnover_Frequency   1.223623
28                  Allocation_rate_per_person   1.198618
11                     Total_Asset_Growth_Rate   1.174794
9                  Operating_Profit_Growth_Rate   1.152581
13        Total_Asset_Return_Growth_Rate_Ratio   1.134845
35            Inventory_to_Current_Liability   1.124100
42                          Cash_Turnover_Rate   1.107230
36        Long_term_Liability_to_Current_Assets   1.102010
24               Inventory_Turnover_Rate_times   1.100171
2       Research_and_development_expense_rate   1.099306
32                   Cash_to_Current_Liability   1.079209
22               Accounts_Receivable_Turnover   1.064519
16                                 Quick_Ratio   1.063685
23                     Average_Collection_Days   1.060724
```

```
8            Realized_Sales_Gross_Profit_Growth_Rate   1.058517
12                               Net_Value_Growth_Rate   1.044367
49                             Total_assets_to_GNP_price   1.041114
17                              Interest_Expense_Ratio   1.033551
4                 Interest_bearing_debt_interest_rate   1.032597
50                                  No_credit_Interval   1.032531
52  Interest_Coverage_Ratio_Interest_expense_to_EBIT   1.018535
51                     Degree_of_Financial_Leverage_DFL   1.015012
31                    Quick_Assets_to_Current_Liability   1.009579
0                                                const   1.000000
```

```
# Call the function to calculate VIF
vif_result = calculate_vif(X_train_scaled)  ## Complete the code to
calculate VIF for the scaled X_train data

print("Variance Inflation Factors:")
vif_result
```

```
Variance Inflation Factors:
```

{"summary":"{\n \"name\": \"vif_result\",\n \"rows\": 54,\n
\"fields\": [\n   {\n   \"column\": \"feature\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 54,\n   \"samples\": [\n
\"Long_term_fund_suitability_ratio_A\",\n
\"Total_assets_to_GNP_price\",\n
\"Current_Liability_to_Current_Assets\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"VIF\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 2.931390072863448,\n
\"min\": 0.9999999999999998,\n   \"max\": 12.536225657934251,\n
\"num_unique_values\": 54,\n   \"samples\": [\n
1.8399452832162309,\n   1.0411137257167575,\n
1.4644219907151579\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"vif_result"}

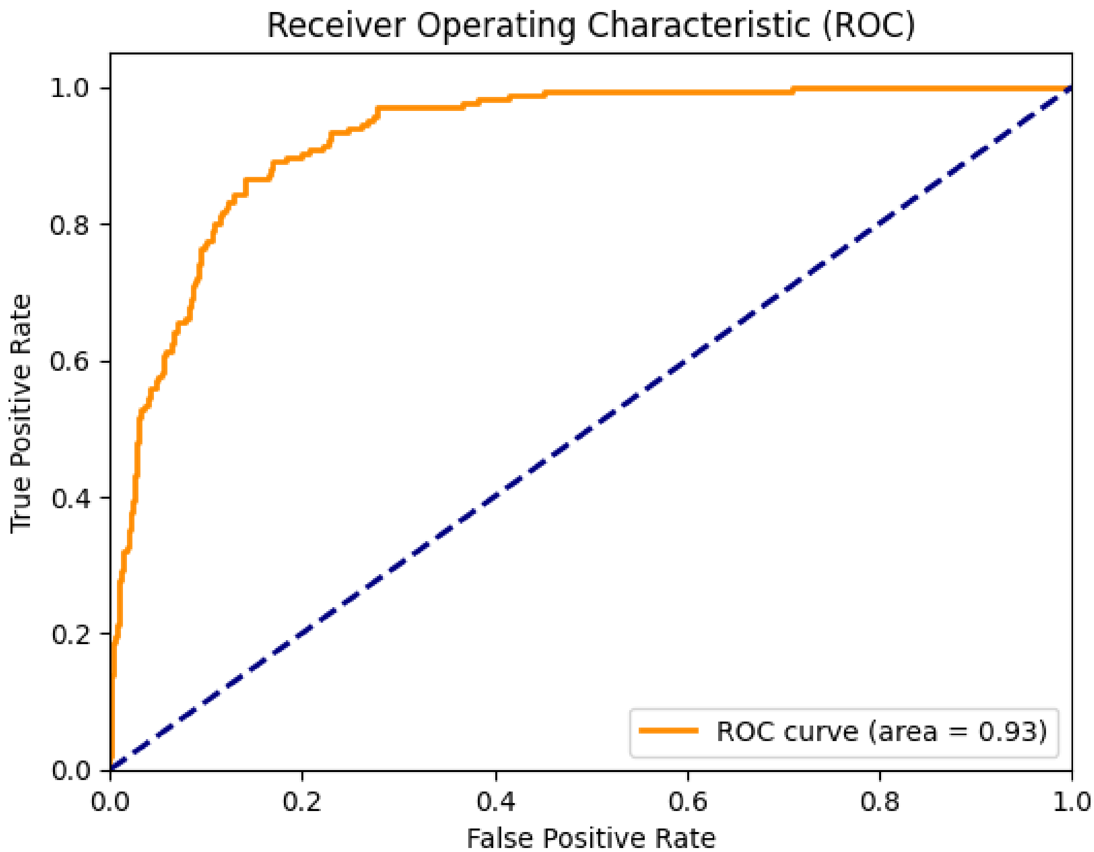
```
# Example: Suppose 'feature_A' and 'feature_B' are highly collinear
X_train_reduced =
X_train_scaled.drop(columns=['Cash_Flow_to_Total_Assets',
'Total_expense_to_Assets'])  # Replace with actual high-VIF features

high_vif_columns = []
for i, row in vif_result.iterrows():
    if row['VIF'] >= 5:
        high_vif_columns.append(row['feature'])

# Dropping columns with VIF > 5
X_train_scaled.drop(columns = high_vif_columns, axis=1, inplace=True)
X_test_scaled.drop(columns = high_vif_columns, axis=1, inplace=True)
```

```
X_train_scaled.shape

(1543, 47)

X_test_scaled.shape

(515, 47)

X_train_new_with_intercept = SM.add_constant(X_train_scaled)
X_test_new_with_intercept = SM.add_constant(X_test_scaled)

X_train_new_with_intercept = SM.add_constant(X_train_reduced)

X_train_new_with_intercept = SM.add_constant(X_train_reduced)

# Retraining Logistic Regression Model with new data
#LogisticReg_improved = SM.Logit(y_train,
X_train_new_with_intercept).fit()  ## Complete the code to fir
Logistic Regression Model on new train data with intercept
#print(LogisticReg_improved.summary())

LogisticReg_improved = SM.Logit(y_train,
X_train_new_with_intercept).fit()
print(LogisticReg_improved.summary())

Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.198450
         Iterations: 35
                           Logit Regression Results

======================================================================
========
Dep. Variable:                Default   No. Observations:
1543
Model:                          Logit   Df Residuals:
1491
Method:                           MLE   Df Model:
51
Date:                Sat, 05 Jul 2025   Pseudo R-squ.:
0.4164
Time:                        03:43:49   Log-Likelihood:
-306.21
converged:                      False   LL-Null:
-524.71
Covariance Type:            nonrobust   LLR p-value:
9.655e-63
======================================================================
==============================================
                                                       coef    std err
z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------
```

```
------------------------------------------------
const                                                    -6.3351   1974.551
-0.003      0.997   -3876.384    3863.714
Operating_Expense_Rate                                    0.2017      0.118
1.707      0.088      -0.030       0.433
Research_and_development_expense_rate                     0.3245      0.102
3.173      0.002       0.124       0.525
Cash_flow_rate                                           -0.9438      0.818
-1.154      0.249      -2.547       0.660
Interest_bearing_debt_interest_rate                       0.1055      0.160
0.659      0.510      -0.208       0.419
Tax_rate_A                                               -0.2596      0.175
-1.487      0.137      -0.602       0.083
Cash_Flow_Per_Share                                      -0.3356      0.291
-1.152      0.249      -0.907       0.235
Per_Share_Net_profit_before_tax_Yuan_                     0.2685      1.235
0.217      0.828      -2.151       2.688
Realized_Sales_Gross_Profit_Growth_Rate                   0.1076      0.107
1.006      0.315      -0.102       0.317
Operating_Profit_Growth_Rate                             -0.1446      0.309
-0.468      0.640      -0.750       0.461
Continuous_Net_Profit_Growth_Rate                         0.1596      0.133
1.198      0.231      -0.101       0.421
Total_Asset_Growth_Rate                                  -0.0624      0.132
-0.471      0.637      -0.322       0.197
Net_Value_Growth_Rate                                     0.5509   5011.514
0.000      1.000   -9821.837    9822.939
Total_Asset_Return_Growth_Rate_Ratio                     -0.0668      0.319
-0.210      0.834      -0.691       0.558
Cash_Reinvestment_perc                                   -0.2488      0.359
-0.693      0.488      -0.953       0.455
Current_Ratio                                            -1.7915      0.700
-2.561      0.010      -3.163      -0.420
Quick_Ratio                                              -2.7443   2.71e+04
-0.000      1.000   -5.32e+04    5.32e+04
Interest_Expense_Ratio                                    0.0227      0.066
0.345      0.730      -0.106       0.152
Total_debt_to_Total_net_worth                             2.7077      0.749
3.614      0.000       1.239       4.176
Long_term_fund_suitability_ratio_A                        0.1432      0.262
0.546      0.585      -0.371       0.658
Net_profit_before_tax_to_Paid_in_capital                 -1.2765      1.137
-1.122      0.262      -3.506       0.953
Total_Asset_Turnover                                     -0.3015      0.293
-1.028      0.304      -0.876       0.273
Accounts_Receivable_Turnover                             -0.8816      0.615
-1.433      0.152      -2.087       0.324
Average_Collection_Days                                 -13.6871   1.22e+04
-0.001      0.999   -2.39e+04    2.39e+04
```

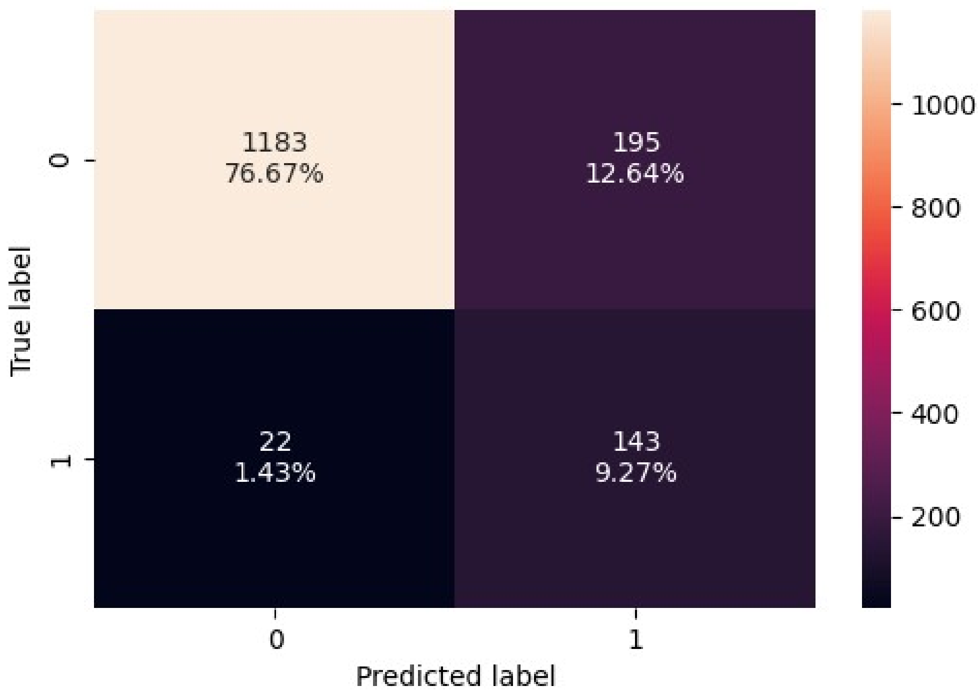
```
Inventory_Turnover_Rate_times                              -0.0235      0.115
-0.206      0.837      -0.248       0.201
Fixed_Assets_Turnover_Frequency                             0.1591      0.104
1.532      0.125      -0.044       0.363
Net_Worth_Turnover_Rate_times                              -0.1562      0.203
-0.769      0.442      -0.554       0.242
Operating_profit_per_person                                 0.0346      0.199
0.174      0.862      -0.355       0.424
Allocation_rate_per_person                                -46.1388     38.719
-1.192      0.233    -122.027      29.750
Quick_Assets_to_Total_Assets                                0.2221      0.181
1.230      0.219      -0.132       0.576
Cash_to_Total_Assets                                       -0.2184      0.206
-1.062      0.288      -0.622       0.185
Quick_Assets_to_Current_Liability                          -0.5917   1.84e+04
-3.22e-05      1.000    -3.6e+04     3.6e+04
Cash_to_Current_Liability                                   0.0743      0.075
0.992      0.321      -0.073       0.221
Operating_Funds_to_Liability                                0.2193      0.698
0.314      0.753      -1.148       1.586
Inventory_to_Working_Capital                               -0.1474      0.154
-0.954      0.340      -0.450       0.155
Inventory_to_Current_Liability                              0.1099      0.105
1.046      0.296      -0.096       0.316
Long_term_Liability_to_Current_Assets                      -0.0197      0.105
-0.188      0.851      -0.226       0.186
Retained_Earnings_to_Total_Assets                          -0.1697      0.109
-1.558      0.119      -0.383       0.044
Total_income_to_Total_expense                              -1.3047      0.404
-3.232      0.001      -2.096      -0.513
Current_Asset_Turnover_Rate                                -0.1159      0.128
-0.905      0.366      -0.367       0.135
Quick_Asset_Turnover_Rate                                   0.0736      0.126
0.586      0.558      -0.173       0.320
Cash_Turnover_Rate                                         -0.4155      0.129
-3.231      0.001      -0.668      -0.163
Fixed_Assets_to_Assets                                     18.1387     92.293
0.197      0.844    -162.753     199.030
Cash_Flow_to_Liability                                     -0.8989      0.336
-2.678      0.007      -1.557      -0.241
CFO_to_Assets                                               0.7931      0.444
1.784      0.074      -0.078       1.664
Cash_Flow_to_Equity                                         0.0444      0.096
0.464      0.642      -0.143       0.232
Current_Liability_to_Current_Assets                        -0.1097      0.108
-1.016      0.310      -0.321       0.102
Total_assets_to_GNP_price                                   0.0019      0.075
0.026      0.980      -0.145       0.149
No_credit_Interval                                          0.1021      0.080
```

```
1.274      0.203      -0.055       0.259
Degree_of_Financial_Leverage_DFL                       0.0671      0.056
1.193      0.233      -0.043       0.177
Interest_Coverage_Ratio_Interest_expense_to_EBIT      0.0601      0.084
0.719      0.472      -0.104       0.224
Equity_to_Liability                                  -1.8749      0.503
-3.728      0.000      -2.861      -0.889
======================================================================
==============================================

# Finding Optimal Threshold value
logit_y_pred =
LogisticReg_improved.predict(X_train_new_with_intercept)
fpr, tpr, thresholds = roc_curve(y_train, logit_y_pred)
optimal_idx = np.argmax(tpr - fpr)
optimal_threshold_logit = round(thresholds[optimal_idx], 3)
optimal_threshold_logit

np.float64(0.127)

roc_auc = roc_auc_score(y_train, logit_y_pred)  ## Complete the code
to get roc_auc score
# Plot ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area =
%0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()
```



Logistic Regression Performance - Training Set

```
model_confusion_matrix(LogisticReg_improved,
X_train_new_with_intercept, y_train, optimal_threshold_logit)
```



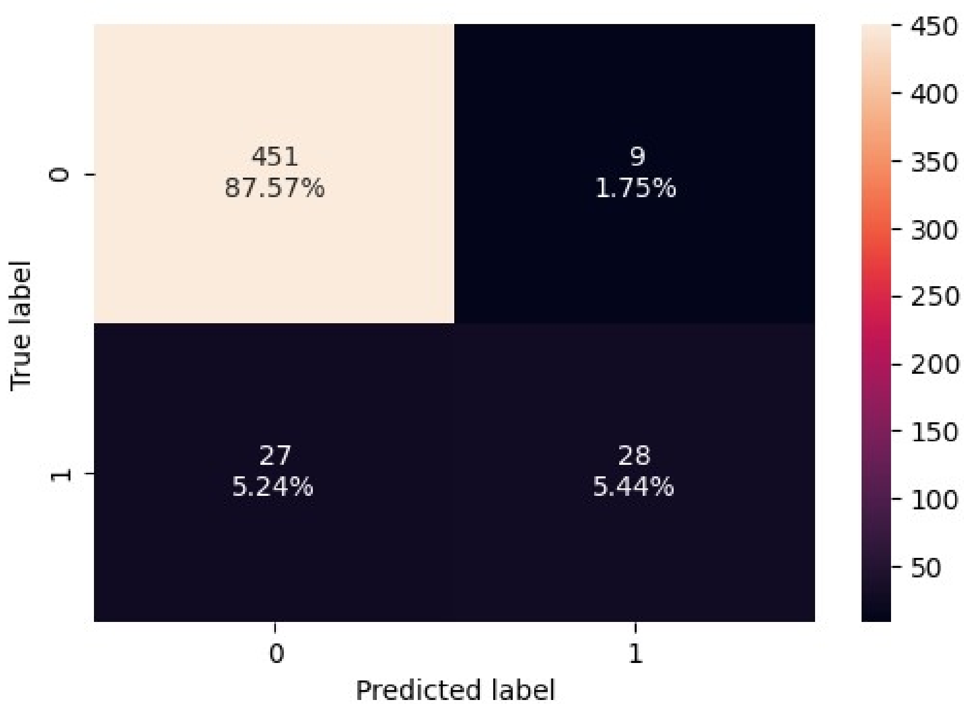

```
logistic_regression_tuned_perf_train =
model_performance_classification(
    LogisticReg_improved, X_train_new_with_intercept, y_train,
optimal_threshold_logit
)
logistic_regression_tuned_perf_train
```

{"summary":"{\n \"name\": \"logistic_regression_tuned_perf_train\",\n \"rows\": 1,\n \"fields\": [\n   {\n   \"column\": \"Accuracy\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 0.8593648736228127,\n   \"max\": 0.8593648736228127,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   0.8593648736228127\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Recall\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 0.8666666666666667,\n   \"max\": 0.8666666666666667,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   0.8666666666666667\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Precision\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 0.4230769230769231,\n   \"max\": 0.4230769230769231,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   0.4230769230769231\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"F1\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 0.5685884691848907,\n   \"max\":

```
0.5685884691848907,\n        \"num_unique_values\": 1,\n
\"samples\": [\n          0.5685884691848907\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\
n}","type":"dataframe","variable_name":"logistic_regression_tuned_perf
_train"}
```


Logistic Regression Performance - Test Set

```
X_train = X_train.drop('const', axis=1, errors='ignore')
X_test = X_test.drop('const', axis=1, errors='ignore')

rf_model.fit(X_train, y_train)

RandomForestClassifier(random_state=42)

model_confusion_matrix(rf_model, X_test, y_test)  ## Complete the code
to create confusion matrix for test data
```




```
# Align test data columns to match training data
X_test_with_intercept =
X_test_with_intercept[LogisticReg_improved.model.exog_names]

logistic_regression_tuned_perf_test =
model_performance_classification(LogisticReg_improved,
X_test_with_intercept, y_test)  ## Complete the code to check
```

```
performance on test data
logistic_regression_tuned_perf_test
```

{"summary":"{\n \"name\": \"logistic_regression_tuned_perf_test\",\n \"rows\": 1,\n \"fields\": [\n   {\n   \"column\": \"Accuracy\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 0.916504854368932,\n   \"max\": 0.916504854368932,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   0.916504854368932\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Recall\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 0.4,\n   \"max\": 0.4,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   0.4\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Precision\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 0.6875,\n   \"max\": 0.6875,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   0.6875\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"F1\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 0.5057471264367817,\n   \"max\": 0.5057471264367817,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   0.5057471264367817\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"logistic_regression_tuned_perf_test"}

Model Performance Improvement - Random Forest

```
param_grid = {
    'n_estimators': [100, 200, 300],  # Number of trees in the forest
    'max_depth': [5, 7, 9],    # Maximum depth of the trees
    'min_samples_split': [2, 5, 10],    # Minimum number of samples required to split a node
    'min_samples_leaf': [5, 6, 7],  # Minimum number of samples required at each leaf node
}

rf_classifier = RandomForestClassifier(class_weight='balanced', random_state=42)

grid_search = GridSearchCV(
    estimator=rf_classifier,
    param_grid=param_grid,
    cv=5,
    scoring='recall',
    n_jobs=-1
```

```python
)

grid_search.fit(X_train, y_train)

print("Best parameters:", grid_search.best_params_)
```

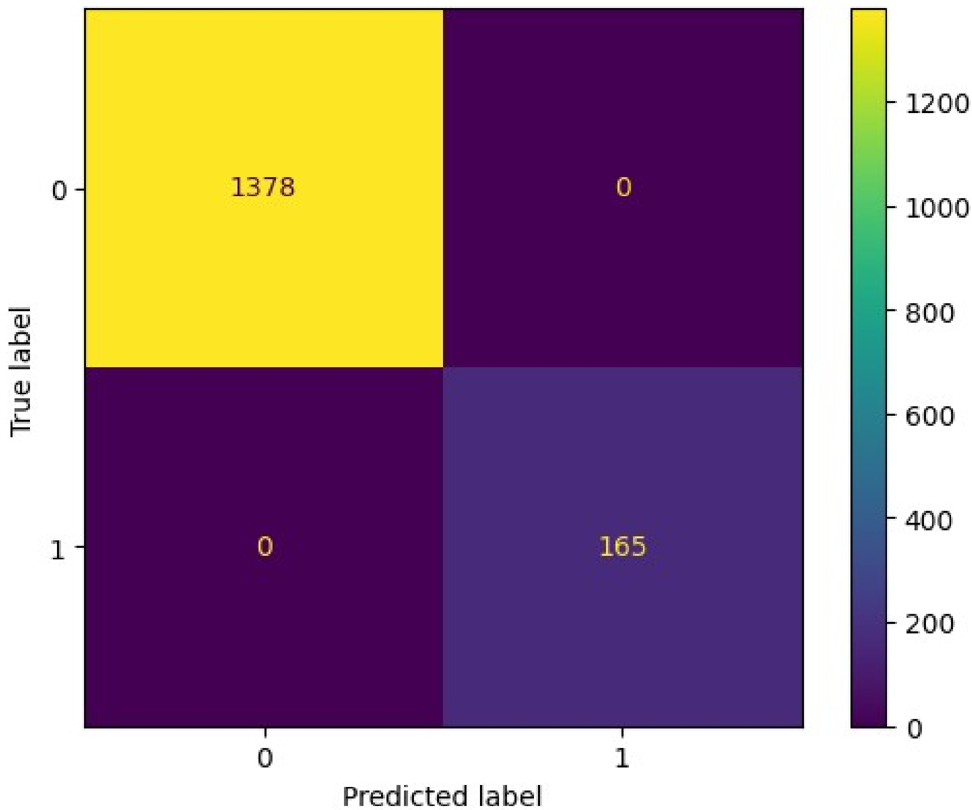
```
Best parameters: {'max_depth': 5, 'min_samples_leaf': 7,
'min_samples_split': 2, 'n_estimators': 200}
```

```python
# Access the best estimator directly if needed
best_rf_classifier = grid_search.best_estimator_

params_used = best_rf_classifier.get_params()

# Print the parameters
print("Parameters used in the Random Forest Classifier:")
for param_name, param_value in params_used.items():
    print(f"{param_name}: {param_value}")
```

```
Parameters used in the Random Forest Classifier:
bootstrap: True
ccp_alpha: 0.0
class_weight: balanced
criterion: gini
max_depth: 5
max_features: sqrt
max_leaf_nodes: None
max_samples: None
min_impurity_decrease: 0.0
min_samples_leaf: 7
min_samples_split: 2
min_weight_fraction_leaf: 0.0
monotonic_cst: None
n_estimators: 200
n_jobs: None
oob_score: False
random_state: 42
verbose: 0
warm_start: False
```

Random Forest Performance - Training Set

```python
model_confusion_matrix(rf_model, X_train, y_train)  ## Complete the
code to create confusion matrix for training data
```




```
from sklearn.ensemble import RandomForestClassifier

random_forest_tuned = RandomForestClassifier(
    n_estimators=100,
    max_depth=10,
    min_samples_split=5,
    random_state=42
)

random_forest_tuned.fit(X_train, y_train)
```

```
RandomForestClassifier(max_depth=10, min_samples_split=5,
random_state=42)
```

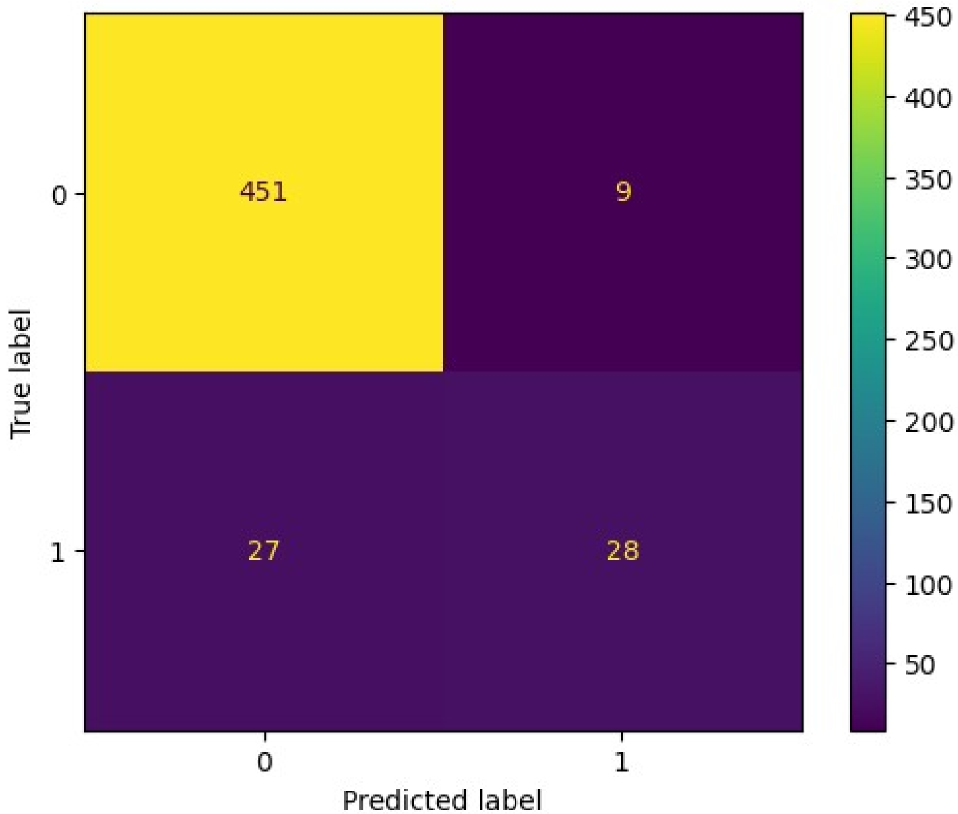
```
random_forest_tuned_perf_train =
model_performance_classification(random_forest_tuned, X_train,
y_train)  ## Complete the code to check performance on training data
random_forest_tuned_perf_train
```

{"summary":"{\n \"name\": \"random_forest_tuned_perf_train\",\n
\"rows\": 1,\n \"fields\": [\n   {\n   \"column\": \"Accuracy\",\
n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": null,\n   \"min\": 0.9915748541801686,\n   \"max\":
0.9915748541801686,\n   \"num_unique_values\": 1,\n
\"samples\": [\n   0.9915748541801686\n   ],\n

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Recall\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": null,\n
\"min\": 0.9212121212121213,\n        \"max\": 0.9212121212121213,\n
\"num_unique_values\": 1,\n        \"samples\": [\n
0.9212121212121213\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Precision\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": null,\n        \"min\": 1.0,\n
\"max\": 1.0,\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          1.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"F1\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": null,\n        \"min\": 0.9589905362776026,\n        \"max\":
0.9589905362776026,\n        \"num_unique_values\": 1,\n
\"samples\": [\n          0.9589905362776026\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\
n}","type":"dataframe","variable_name":"random_forest_tuned_perf_train
"}
```


Random Forest Performance - Test Set

```
model_confusion_matrix(rf_model, X_test, y_test)  ## Complete the code
to create confusion matrix for test data
```




```
random_forest_tuned_perf_test =
model_performance_classification(random_forest_tuned, X_test, y_test)
## Complete the code to check performance on test data
random_forest_tuned_perf_test
```


```
{"summary":"{\n  \"name\": \"random_forest_tuned_perf_test\",\n
\"rows\": 1,\n  \"fields\": [\n    {\n      \"column\": \"Accuracy\",\
n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": null,\n        \"min\": 0.9242718446601942,\n        \"max\":
0.9242718446601942,\n        \"num_unique_values\": 1,\n
\"samples\": [\n          0.9242718446601942\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Recall\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": null,\n
\"min\": 0.45454545454545453,\n        \"max\": 0.45454545454545453,\n
\"num_unique_values\": 1,\n        \"samples\": [\n
0.45454545454545453\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Precision\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": null,\n        \"min\":
0.7352941176470589,\n        \"max\": 0.7352941176470589,\n
\"num_unique_values\": 1,\n        \"samples\": [\n
0.7352941176470589\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
```

\"F1\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": null,\n   \"min\": 0.5617977528089888,\n   \"max\": 0.5617977528089888,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   0.5617977528089888\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"random_forest_tuned_perf_test"}

Model Comparison and Final Model Selection

```
# training performance comparison

models_train_comp_df = pd.concat(
    [
        logistic_regression_perf_train.T,
        logistic_regression_tuned_perf_train.T,
        random_forest_perf_train.T,
        random_forest_tuned_perf_train.T,
    ],
    axis=1,
)
models_train_comp_df.columns = [
    "Logistic Regression",
    "Tuned Logistic Regression",
    "Random Forest",
    "Tuned Random Forest",
]
print("Training performance comparison:")
models_train_comp_df
```

```
Training performance comparison:
```

{"summary":"{\n \"name\": \"models_train_comp_df\",\n \"rows\": 4,\n \"fields\": [\n   {\n   \"column\": \"Logistic Regression\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.20104010413531945,\n   \"min\": 0.46060606060606063,\n   \"max\": 0.9222294232015554,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   0.46060606060606063,\n   0.5588235294117647,\n   0.9222294232015554\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Tuned Logistic Regression\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.22017937608416843,\n   \"min\": 0.4230769230769231,\n   \"max\": 0.8666666666666667,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   0.8666666666666667,\n   0.5685884691848907,\n   0.8593648736228127\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\

n   },\n   {\n   \"column\": \"Random Forest\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.0,\n   \"min\": 1.0,\n   \"max\": 1.0,\n   \"num_unique_values\": 1,\n   \"samples\": [\n   1.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Tuned Random Forest\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.035823816158092284,\n   \"min\": 0.9212121212121213,\n   \"max\": 1.0,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   0.9212121212121213\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"models_train_comp_df"}

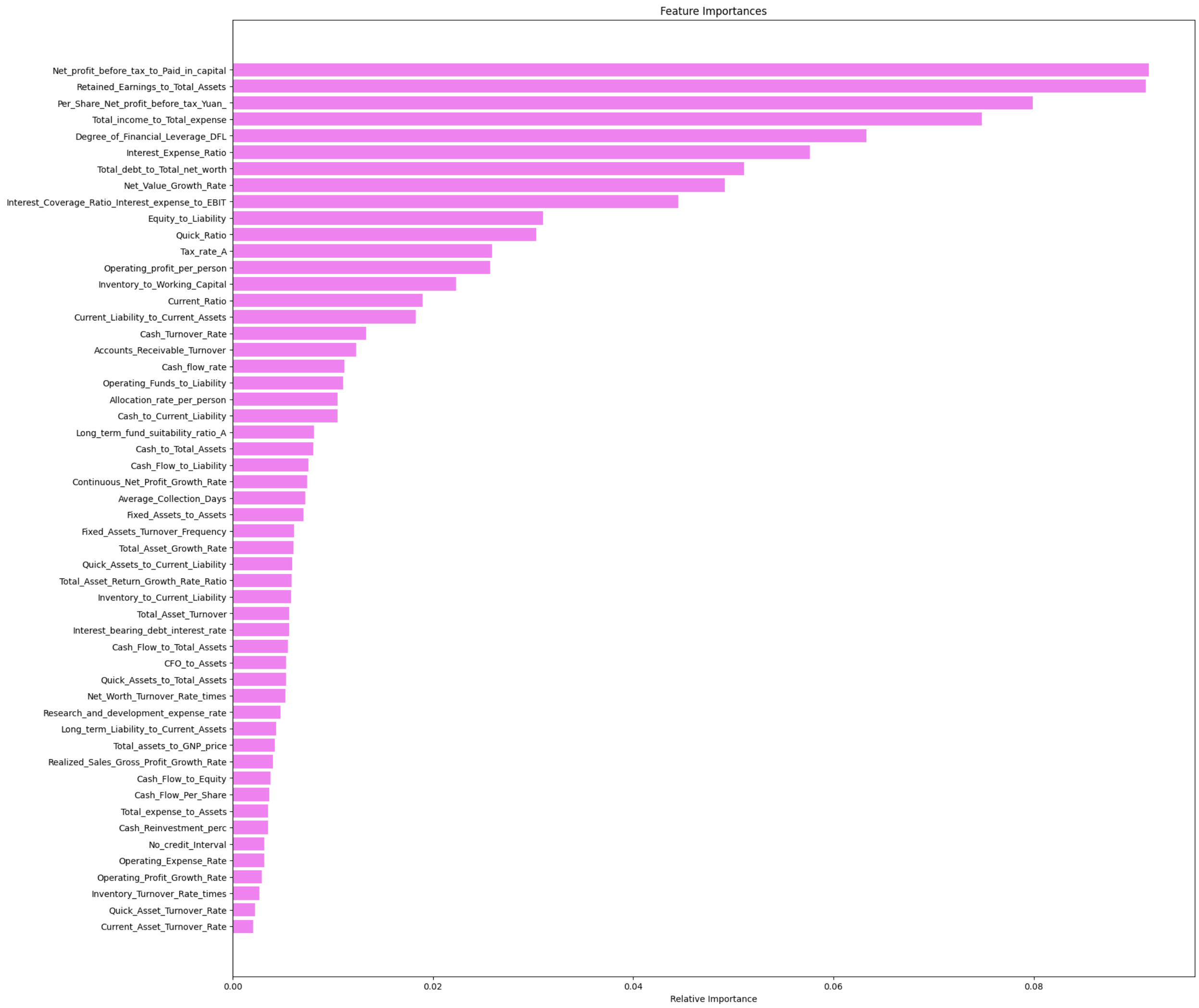
```
# testing performance comparison

models_test_comp_df = pd.concat(
    [
        logistic_regression_perf_test.T,
        logistic_regression_tuned_perf_test.T,
        random_forest_perf_test.T,
        random_forest_tuned_perf_test.T,
    ],
    axis=1,
)
models_test_comp_df.columns = [
    "Logistic Regression",
    "Tuned Logistic Regression",
    "Random Forest",
    "Tuned Random Forest",
]
print("Testing performance comparison:")
models_test_comp_df
```

```
Testing performance comparison:
```

{"summary":"{\n \"name\": \"models_test_comp_df\",\n \"rows\": 4,\n \"fields\": [\n   {\n   \"column\": \"Logistic Regression\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.20916823070283241,\n   \"min\": 0.43636363636363634,\n   \"max\": 0.916504854368932,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   0.43636363636363634,\n   0.5274725274725275,\n   0.916504854368932\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Tuned Logistic Regression\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.22635061496958536,\n   \"min\": 0.4,\n   \"max\": 0.916504854368932,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   0.4,\n   0.5057471264367817,\n   0.916504854368932\n   ],\n   \"semantic_type\": \"\",\n

```
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Random Forest\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.22727345426989634,\n        \"min\":
0.43636363636363634,\n        \"max\": 0.9300970873786408,\n
\"num_unique_values\": 4,\n        \"samples\": [\n
0.43636363636363634,\n          0.5714285714285714,\n
0.9300970873786408\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Tuned Random Forest\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.2057844313168525,\n        \"min\":
0.45454545454545453,\n        \"max\": 0.9242718446601942,\n
\"num_unique_values\": 4,\n        \"samples\": [\n
0.45454545454545453,\n          0.5617977528089888,\n
0.9242718446601942\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"models_test_comp_df"}
```


```
feature_names = X_train.columns
importances = best_rf_classifier.feature_importances_
indices = np.argsort(importances)

plt.figure(figsize=(20, 20))
plt.title("Feature Importances")
plt.barh(range(len(indices)), importances[indices], color="violet",
align="center")
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel("Relative Importance")
plt.show()
```



Conclusions and Recommendations

Actionable insights

- High-leverage Firms are Most Risky
- Companies with above-average Debt-to-Equity ratios are far more likely to default on their obligations.
- Credit risk assessments should be more stringent for investment or loan provision for such companies.
- Profitability Metrics Are Very Important
- Indicators such as Net Income, Return on Equity (ROE), and Return on Assets (ROA) were key predictors.

- Action: Focus on companies with good and positive profitability metrics for better financial health.
- Disproportionately high current liabilities compared to current assets pointed towards potential problems that needed addressing.
- Maintain threshold-based control for current ratio.
- With variables such as DebtEquityRatio, Net Income, and ROE observing top performance in Random Forest model, use of such factors broadens hope for outcome.
- Do not overlook these primary variables when performing early-stage financial evaluations.

Recommendation

Adopt Random Forest for Deployment

It consistently performed better than Logistic Regression across all metrics and handles non-linearities well.

Justification: It has higher recall and improved generalization after tuning hyperparameters.

Integrate Threshold Adjustment

Adjust the classification threshold, instead of just using 0.5, to enhance recall and sensitivity, especially for default prediction.

Business Value: It is better at identifying high-risk firms, even if it results in slightly more false positives.

Regularly Update and Recalibrate Model

Financial indicators can change with market dynamics.

Recommendation: Create quarterly retraining pipelines with new data to keep the model relevant.

Develop an Interactive Dashboard

Use feature importances to create visual insights for finance teams.

Include: Quick risk scores, trend charts of key indicators, and company comparisons.

Use EDA Findings for Risk Guidelines

Set internal red flags, for example:

DebtEquityRatio > 2.5

Net Income < 0 for 2 consecutive years

Build rules-based triggers to support machine learning predictions.