Forecasting Company Insolvency

Project: Milestone 2 Project EDA and Feature Engineering & Selection

Student 1: Anjali Haryani

Student 2: Payal Chavan

Student 1: +1 716 277 6868

Student 2: +1 206 290 1057

Student 1: haryani.a@northeastern.edu

Student 2: chavan.pay@northeastern.edu

Percentage of Effort Contributed by Student 1: 50%

Percentage of Effort Contributed by Student 2: 50%

Signature of Student 1: Anjali Haryani

Signature of Student 2: Payal Chavan

Submission Date: 02/22/2024

INDEX

Introduction:

Ensuring profitability is vital for business expansion. The financial health of a company hinges on various factors, with even minor adjustments capable of yielding significant outcomes. Therefore, it is imperative to analyze diverse information about a company to assess its risk of bankruptcy. This proactive approach enables organizations to identify crucial warning signs early on, allowing for strategic adjustments to prevent financial insolvency. However, navigating this terrain is complex; each industry employs distinct data tracking methods, necessitating tailored approaches. Consequently, applying standardized methodologies across industries may yield inaccurate results. Leveraging data mining techniques offers a promising avenue for predicting corporate bankruptcy. By analyzing intricate datasets encompassing various financial indicators, companies can gain insights into potential financial vulnerabilities and preemptively safeguard against insolvency risks. Thus, integrating data mining into bankruptcy prediction strategies holds immense value in steering companies towards sustainable growth and financial stability.

Problem Definition:

The central aim of this project is to utilize various data mining methodologies to ascertain whether companies, based on distinct accounting indicators, face the prospect of bankruptcy (a classification predicament).
The project's primary objective is to compare factors like accuracy, precision etc. - across different data mining methodologies to pinpoint the most efficient one.

The project will encompass the following stages:

1. Identify the pivotal features associated with company insolvency.
2. Investigate correlations among the provided categories.
3. Tackle data imbalances.
4. Utilize range to data mining algorithms to construct models and assess their performance using metrics like accuracy, precision, AUC, recall, F1 score to ascertain the optimal approach for prediction.

Dataset Selection:

Data is selected from UCI machine learning repository https://archive.ics.uci.edu

Data Source:

https://archive.ics.uci.edu/dataset/572/taiwanese+bankruptcy+prediction

Data Description:

The information was gathered from the Economic Journal. The Stock Exchange's corporate directive served as the basis for the definition of company bankruptcy. A set of data with various corporate attributes can be found in the file. With a file size of roughly 11.46 MB, this is sufficient for a credible estimate. There are around 100 columns and 6000 rows in the data set. The output characteristic designated as Y, Bankrupt or not indicates whether the company is bankrupt or not. Boolean values are the output feature; 0 indicates bankruptcy and 1 indicates not. The input features, which consist of different indications and background data about the company, are the remaining 95 columns. Examples of these input variables include Continuous interest rate, Operating Expense Rate, Operating Gross Margin, Debt ratio %, Liability to Equity, and CFO to Assets. The data set comprises 2 integer variables, 93 decimal variables, and 1 Boolean variable. For simplicity, columns are labeled sequentially as X1, X2, X3, and so forth. The response variable 'Bankrupt or not' holds values of either 0 (indicating insolvency) or 1 (indicating non- insolvency).

Data Dictionary:
There are 96 columns in the dataset. Our target variable is Bankrupt or not. Below are some of the important columns from the Dataset:

Attribute	Description
Y (Target Variable)	Bankrupt OR NOT
X1	ROA_(C) (value and depreciation before interest)
X2	ROA_(A) (value and % before interest & after tax)
X3	ROA_(B) (value and depreciation before & after tax)
X4	Operating_Gross_Margin(OGM)
X5	Realized_Sales_Gross_Margin(RSGM)
X6	Operating_Profit_Rate(OPR)
X7	Pre-tax_net_Interest_Rate(PNIR)
X8	After-tax_net_Interest_Rate(ANIR)
X9	non-industry_income_and_expenditure(NIIE)
X10	Continuous_interest_rate_(after_tax)(CIEA)
X11	Operating_Expense_Rate(OER)
X12	Research_and_development_expense_rate(RDER)
X13	Cash_flow_rate(CFR)
X14	Interest-bearing_debt_interest_rate(IDIR)
X15	Tax_rate(A)(TRA)
X16	Net_Value_Per_Share(B)(NVPS_B)
X17	Net_Value_Per_Share(A)(NVPS_A)
X18	Net_Value_Per_Share(C)(NVPS_C)
X19	Persistent_EPS_in_the_Last_Four_Seasons(PEPSL)
X20	Cash_Flow_Per_Share(CFPS)
X21	Revenue_Per_Share(RPS)

Descriptive Analysis:

We will start with looking at shape of data and information about data:

```
In [3]: # Take a look at the data structure
        CBP_df.shape

Out[3]: (6819, 96)

In [4]: #getting info about the dataset
        CBP_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6819 entries, 0 to 6818
Data columns (total 96 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Bankrupt?                                               6819 non-null   int64
 1    ROA(C) before interest and depreciation before interest 6819 non-null  float64
 2    ROA(A) before interest and % after tax                 6819 non-null   float64
 3    ROA(B) before interest and depreciation after tax      6819 non-null   float64
 4    Operating Gross Margin                                 6819 non-null   float64
 5    Realized Sales Gross Margin                            6819 non-null   float64
 6    Operating Profit Rate                                  6819 non-null   float64
 7    Pre-tax net Interest Rate                              6819 non-null   float64
 8    After-tax net Interest Rate                            6819 non-null   float64
 9    Non-industry income and expenditure/revenue            6819 non-null   float64
 10   Continuous interest rate (after tax)                   6819 non-null   float64
 11   Operating Expense Rate                                 6819 non-null   float64
 12   Research and development expense rate                  6819 non-null   float64
 13   Cash flow rate                                         6819 non-null   float64
 14   Interest-bearing debt interest rate                    6819 non-null   float64
 15   Tax rate (A)                                           6819 non-null   float64
 16   Net Value Per Share (B)                                6819 non-null   float64
 17   Net Value Per Share (A)                                6819 non-null   float64
 18   Net Value Per Share (C)                                6819 non-null   float64
 19   Persistent EPS in the Last Four Seasons                6819 non-null   float64
 20   Cash Flow Per Share                                    6819 non-null   float64
 21   Revenue Per Share (Yuan ¥)                             6819 non-null   float64
 22   Operating Profit Per Share (Yuan ¥)                    6819 non-null   float64
 23   Per Share Net profit before tax (Yuan ¥)               6819 non-null   float64
 24   Realized Sales Gross Profit Growth Rate                6819 non-null   float64
 25   Operating Profit Growth Rate                           6819 non-null   float64
 26   After-tax Net Profit Growth Rate                       6819 non-null   float64
 27   Regular Net Profit Growth Rate                         6819 non-null   float64
 28   Continuous Net Profit Growth Rate                      6819 non-null   float64
 29   Total Asset Growth Rate                                6819 non-null   float64
 30   Net Value Growth Rate                                  6819 non-null   float64
 31   Total Asset Return Growth Rate Ratio                   6819 non-null   float64
 32   Cash Reinvestment %                                    6819 non-null   float64
 33   Current Ratio                                          6819 non-null   float64
```

We can see that there are around 6819 rows in dataset and 96 columns. ‘Bankrupt?’ Is our target variable and all other 95 variables are features which are related to target variable.

Data Preprocessing:

Given below are the steps undertaken for data cleaning and preprocessing:

Step 1: Checking null/missing values in the dataset.

To ensure that the data is clean and ready for analysis, it is important to check for null or missing values in the dataset. However, there were no missing values found in our dataset.

```
# Check for any missing values in the dataset
CBP_df.isnull().sum()
```

```
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
```

Fig 1.1- Checking null/missing values.

Step 2: Checking duplicate values in the dataset.

While cleaning data, it's necessary to ensure that there are no duplicate values present in the dataset. They can lead to incorrect analysis and biased results. Fortunately, there were no duplicate records found in our dataset.

```
# Check for duplicate records
CBP_df.duplicated().sum()
```

```
0
```

Fig 2.1- Checking duplicate values.

Step 3: Removing extra spaces from the column names.

While cleaning the data, we found out that the columns had extra spacing at the beginning. Also, to make it easier to read, we have replaced spaces with underscore.

```
# Removing extra spaces and replacing with '_'
CBP_df.columns = [c.strip().replace(' ', '_') for c in CBP_df.columns]
CBP_df.head()
```

	Bankrupt?	ROA(C)_before_interest_and_depreciation_before_interest	ROA(A)_before_interest_and_%_after_tax	ROA(B)_before_interest_and_depreciation_after_ta
0	1	0.370594	0.424389	0.40575
1	1	0.464291	0.538214	0.51673
2	1	0.426071	0.499019	0.47229
3	1	0.399844	0.451265	0.45773
4	1	0.465022	0.538432	0.52229

5 rows × 96 columns

Fig 3.1- Removing extra spaces from the column names.

Step 4: Checking unique values in the dataset.

We checked the unique values in our dataset, so that we can drop the features having just one unique value.

```
# Checking unique values in dataset
CBP_df.nunique()
```

```
Bankrupt?                                                    2
ROA(C)_before_interest_and_depreciation_before_interest    3333
ROA(A)_before_interest_and_%_after_tax                     3151
ROA(B)_before_interest_and_depreciation_after_tax          3160
Operating_Gross_Margin                                     3781
                                                           ...
Liability_to_Equity                                        6819
Degree_of_Financial_Leverage_(DFL)                         6240
Interest_Coverage_Ratio_(Interest_expense_to_EBIT)         6240
Net_Income_Flag                                               1
Equity_to_Liability                                        6819
Length: 96, dtype: int64
```

Fig 4.1- Checking unique values.

We can see that 'Net_Income_Flag' column has only 1 value which is displayed below.

```
In [10]: CBP_df['Net_Income_Flag'].value_counts()

Out[10]: 1    6819
         Name: Net_Income_Flag, dtype: int64

         It appears that the 'Net_Income_Flag' variable contains only one value, indicating no variability in the target variable. So removing that variable from analysis

In [11]: #dropping column Net_Income_Flag
         CBP_df = CBP_df.drop('Net_Income_Flag', axis=1)
```

Fig 4.2- Removing column 'Net_Income_Flag'.

With only 1 value, indicating it has no variability in target variable. So, removing it from the analysis

Step 5: Checking the summary statistics of the dataset.

The summary statistics returns a new data frame containing statistics such as count, mean, standard deviation, minimum, and maximum values for each feature.

```
# Checking statistics of the dataset
CBP_df.describe()
```

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	...	Net I t /
count	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	6819.000000	...	6819.0
mean	0.032263	0.505180	0.558625	0.553589	0.607948	0.607929	0.998755	0.797190	0.809084	0.303623	...	0.8
std	0.176710	0.060686	0.065620	0.061595	0.016934	0.016916	0.013010	0.012869	0.013601	0.011163	...	0.0
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.0
25%	0.000000	0.476527	0.535543	0.527277	0.600445	0.600434	0.998969	0.797386	0.809312	0.303466	...	0.7
50%	0.000000	0.502706	0.559802	0.552278	0.605997	0.605976	0.999022	0.797464	0.809375	0.303525	...	0.8
75%	0.000000	0.535563	0.589157	0.584105	0.613914	0.613842	0.999095	0.797579	0.809469	0.303585	...	0.8
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.0

8 rows × 96 columns

Fig 5.1- Checking summary statistics.

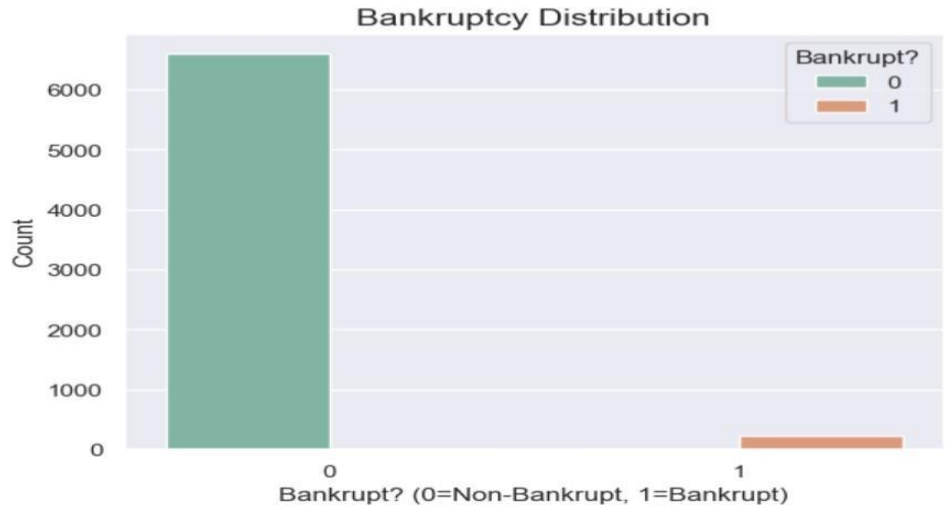
Exploratory Data Analysis:

Data Visualization:

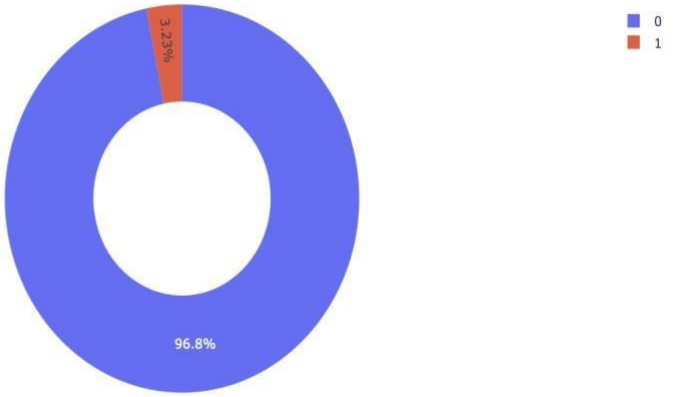
In EDA, we have used the univariate and bivariate plots to analyze the correlation between various features with the target variable i.e., Bankrupt?

1. **Distribution of Target Variable: Bankruptcy:**

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

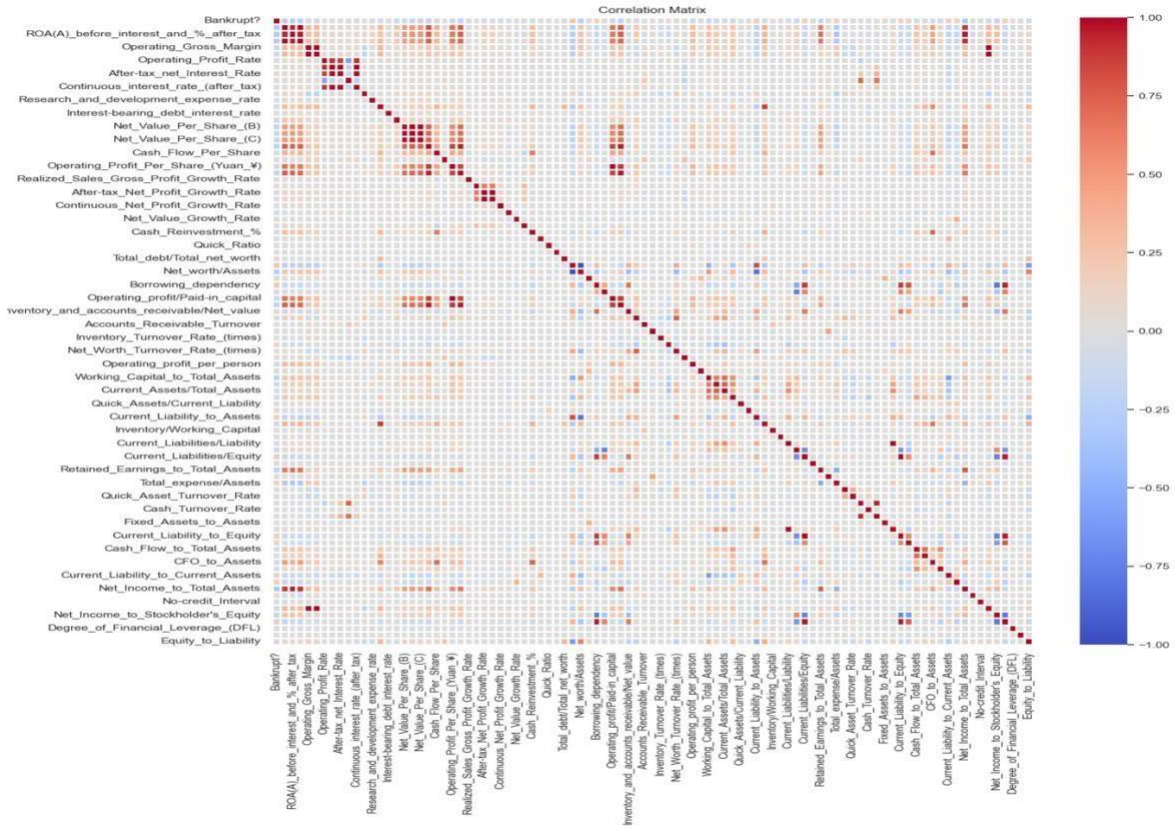


Bankrupt Distribution



The above count plot and pie chart shows the distribution of the outcome variable i.e., Bankrupt? where 0 denotes “non-bankrupt” and 1 denotes “bankrupt”. Only 220 out of 6819 cases are bankrupt, indicating that bankruptcy is an extremely uncommon occurrence. This indicates that most of the data points are not bankrupt, with only roughly 3.2% of them being so. Additionally, this suggests that there is a lot of imbalances in the dataset, which could make modeling or analysis difficult. We will further address this imbalanced data with “SMOTE” analysis.

2. Correlated Features with the target variable:



As we can see from the above heatmap that there are several features which are correlated to each other. There are around 95 features in the dataset. From that 95 we will further try to explore which features are highly correlated to the target variable. This will indicate which features show a significant contribution towards predicting the company's bankruptcy.

We first started with checking correlation of variables with respect to target variable:

```
# Checking correlation between Target variable and other variables

correlation_with_target = correlation['Bankrupt?'].abs().sort_values(ascending=False)
correlation_with_target
```

```
Bankrupt?                                                 1.000000
Net_Income_to_Total_Assets                                0.315457
ROA(A)_before_interest_and_%_after_tax                    0.282941
ROA(B)_before_interest_and_depreciation_after_tax         0.273051
ROA(C)_before_interest_and_depreciation_before_interest   0.260807
                                                            ...
Inventory/Current_Liability                               0.000822
Long-term_Liability_to_Current_Assets                     0.000778
Cash_Flow_to_Sales                                        0.000479
Realized_Sales_Gross_Profit_Growth_Rate                   0.000458
Operating_Profit_Rate                                     0.000230
Name: Bankrupt?, Length: 95, dtype: float64
```

Going ahead, we will choose the features whose correlation with the target variable, Bankrupt? is higher than 0.01. This resulted in 72 features that were highly correlated to the target variable. Moving further, we again performed the similar task to check the features having correlation greater than 0.1 with the target variable. Finally, we acquired 33 features showing a high correlation with the outcome variable.

```
# taking features which have correlation > 0.01 based on target variable bankrupt

features_with_correlation = correlation_with_target[correlation_with_target.values > 0.01].index
len(features_with_correlation)
```

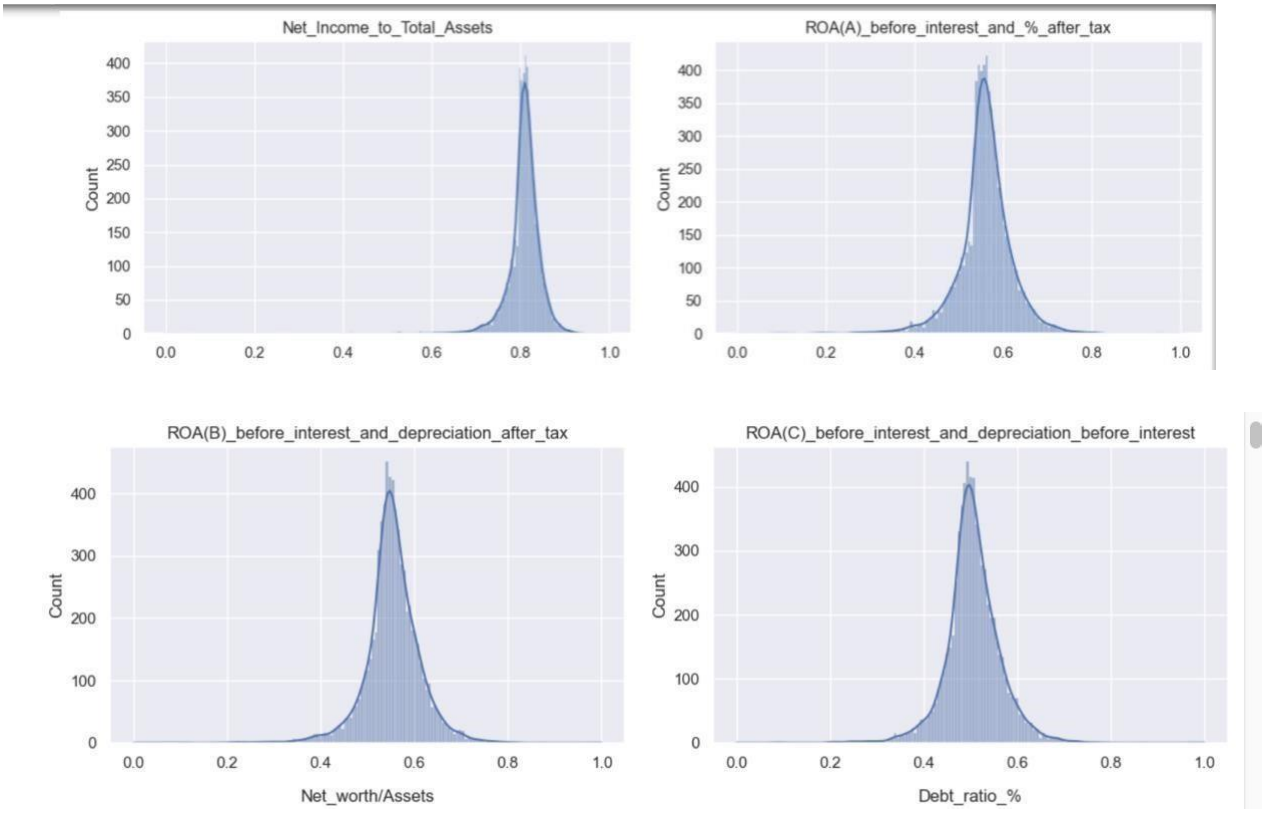
```
72
```

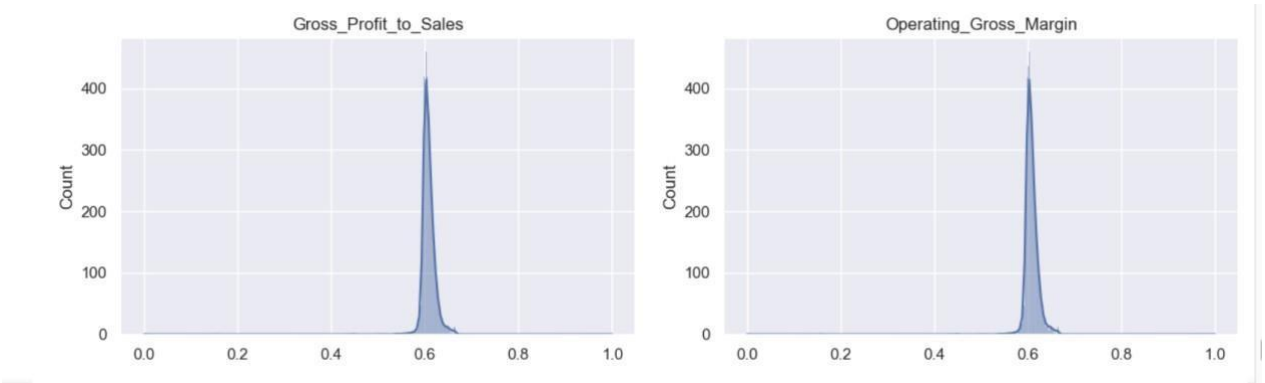
```
# as there are 72 features again taking features which have correlation > 0.1
features_with_correlation = correlation_with_target[correlation_with_target.values > 0.1].index
print(len(features_with_correlation))
features_with_correlation = features_with_correlation[1:] #removing element bankrupt from analysis
```

```
33
```

3. Univariate plots of features correlated with target variable:

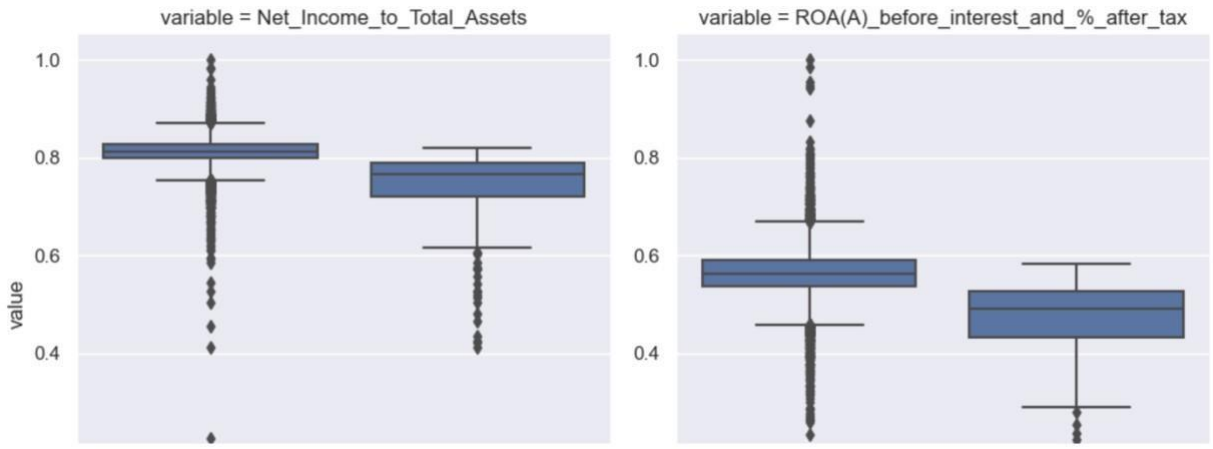
There are around 33 plots which are present in python notebook. Here we have attached screenshots of some of the plots.

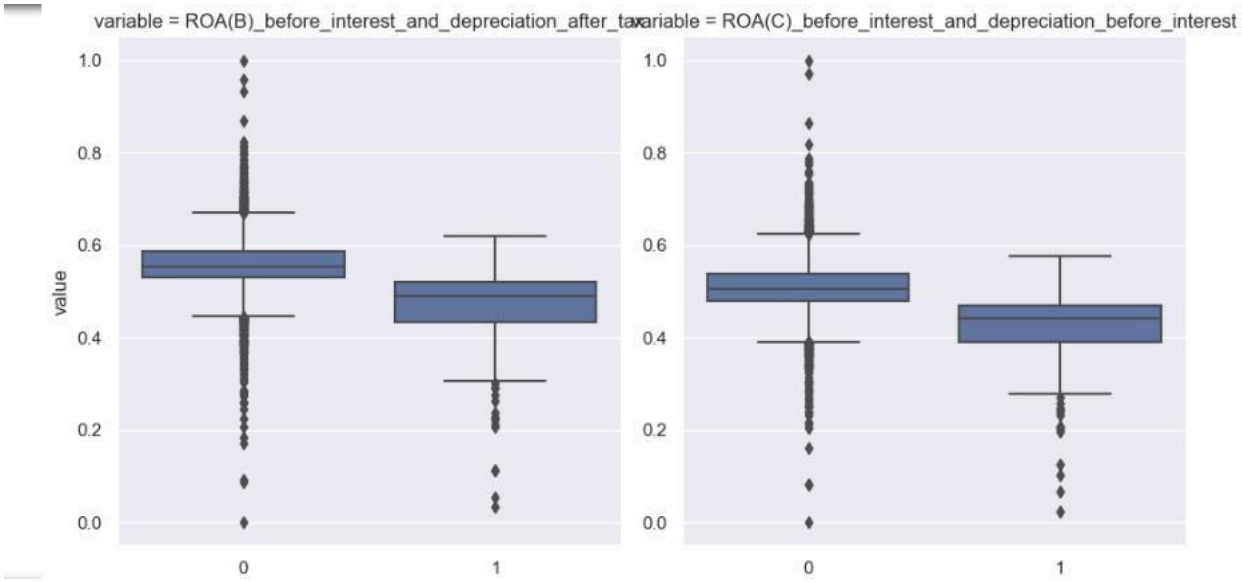




We conducted a comprehensive analysis by plotting histograms for 33 individual features, aiming to evaluate their distributions and their potential correlations with the target variable.
Among our findings, it was evident that the Net Income of total asset column, ROA(B), ROA(C), and the ROA(A) before interest columns displayed a discernible adherence to a normal distribution pattern. This comprehensive analysis sheds light on the underlying relationships between these features and the target variable, providing valuable insights for further investigation.

4. **Bivariate plots of features correlated with the target variable:** There are around 33 plots which are present in python notebook. Here we have attached screenshots of some of the plots.



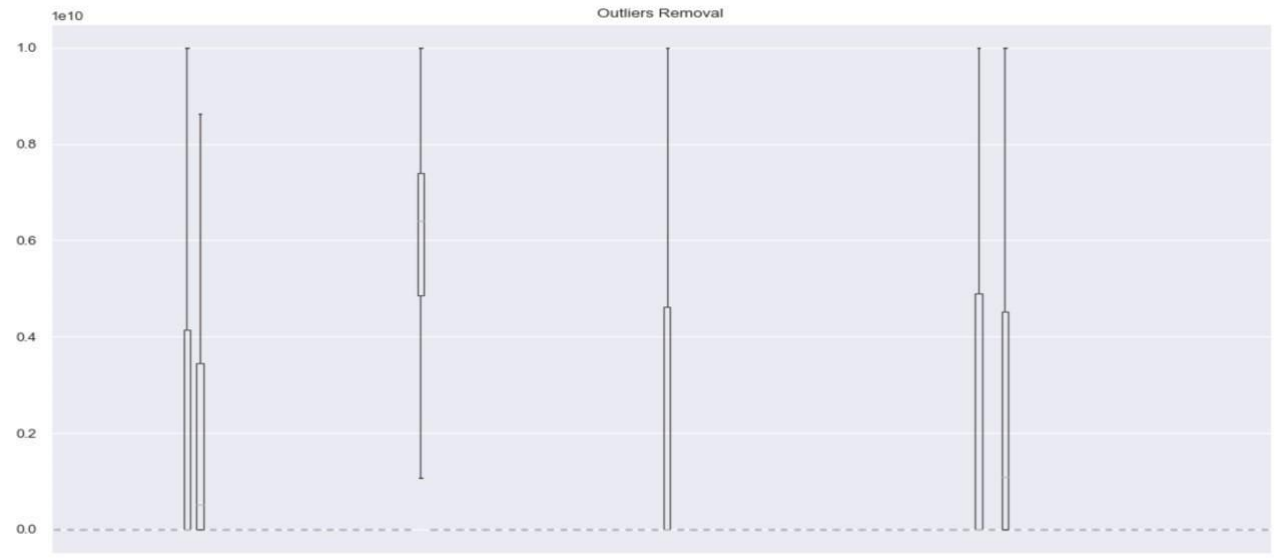


The provided box plot illustrates the distribution of each of the 33 highly correlated features concerning the target variable, which takes on values of either 0 or 1. Utilizing the box plot, we were able to effectively compare the outcome values for each variable and identify any outliers present in the dataset. This visualization technique not only facilitated the examination of the relationship between features and the target variable but also provided a clear depiction of any data points that deviate significantly from the overall pattern. The box plot's ability to present the central tendency, spread, and outliers in a concise manner makes it an excellent visual tool for conducting such analyses.

5. Outlier Analysis:

In our dataset, we have identified the outliers using the “Interquartile Range Method”. We noticed that all the outliers were highly dispersed over the interquartile range. They can have a big impact on our statistical analyses and skew the results. It’s important to carefully identify potential outliers in our dataset and deal with them in an appropriate manner for accurate results. So, we processed the outliers with certain criteria. Outliers larger than the upper quartile were replaced with the upper quartile values. And outliers smaller than the lower quartiles were replaced with the lower quartile values.

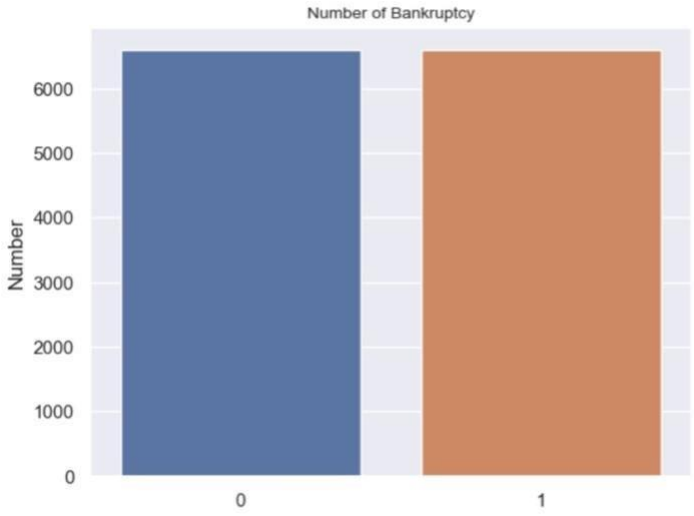
To identify outliers in data, the interquartile range is frequently utilized. Observations that are above Q3 + 1.5 IQR or below Q1 − 1.5 IQR are considered outliers in this context. A statistic that represents the spread of the middle half or middle 50% of the data is called the interquartile range. Outliers are defined as data points that are more than 1.5 times the IQR from Q1 or Q3. We are considering all of the data points rather than designating them as outliers because we can see that they are highly dispersed over the interquartile range.




```
All outliers have been fully handled.
Processing rules:
Outliers larger than the upper quartile are replaced with upper quartile values.
Outliers smaller than the lower quartile are replaced with lower quartile values.
```

6. Class Imbalance - SMOTE Technique:

As we saw, in our previous analysis of “Distribution of Bankruptcy” that there was an imbalance of the outcome variable in the dataset. To address this issue, oversampling method was performed using the SMOTE algorithm. The objective is to increase the samples of the minority class so that the observations of both major and minor classes become equal. With the help of SMOTE, we can increase the number of observations of the minority class in a balanced way, helping the model to become more effective. From the fig below, we can now see that the dataset is balanced for both the classes after the application of SMOTE algorithm.



Feature Engineering:

Feature engineering is the process of selecting and transforming variables when creating a predictive model using machine learning. In this method, we will be selecting the relevant features, create new features from the existing ones, and transform the features to improve the model accuracy.

Dimension Reduction:

As we have lots of features in our dataset, we will be using PCA Technique to reduce the dimensions of the dataset.

Principal Component Analysis (PCA) :

PCA, or Principal Component Analysis, is a statistical technique used for dimensionality reduction in data analysis. It aims to transform a dataset containing possibly correlated variables into a set of linearly uncorrelated variables called principal components. These components capture the maximum variance in the data, allowing for a simplified representation while retaining the essential information present in the original dataset.

1. Before implementing PCA, the dataset is normalized to scale the features.

Scaled Dataset:

	ROA(C)_before_interest_and_depreciation_before_interest	ROA(A)_before_interest_and_%_after_tax	ROA(B)_before_interest_and_depreciation_after_tax
0	-1.534245	-1.369698	-1.479384
1	-0.152736	0.181864	-0.099997
2	-0.844594	-0.550020	-0.920560
3	-1.319363	-1.369698	-1.189468
4	-0.139499	0.185935	0.002821

5 rows × 95 columns

2. In the 2nd step we calculated explained Variance and Cumulative Variance from sklearn.decomposition import PCA library.

Compared to all other components, the first principal component (PC1) accounts for a significantly larger portion of the data variance—roughly 33%. This indicates that PC1 finds the most significant pattern or piece of information in the data. Most of the information is included in the first few components, as evidenced by the explained variance rapidly decreasing as additional components are added. The cumulative percentage of variance shows that 99% of the volatility in the data can be explained by 57 components. This

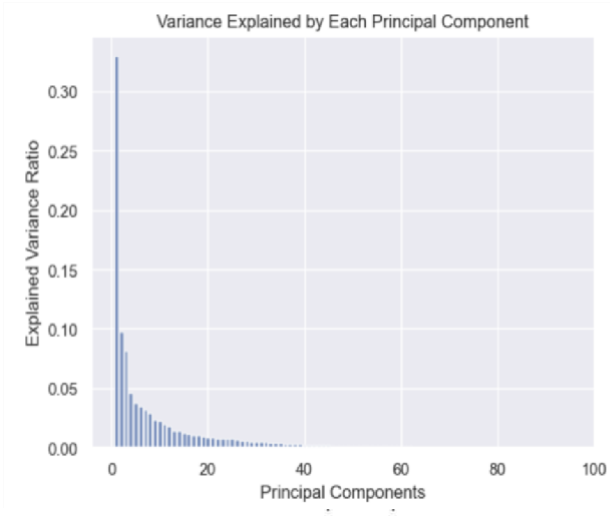
indicates that practically all the data in the original dataset can be represented by 57 components.

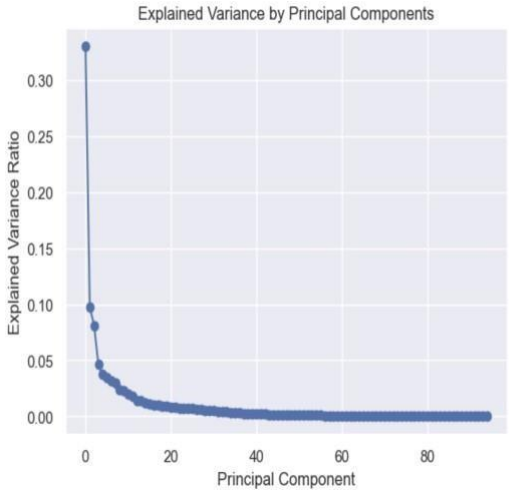
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	...	PC86	PC87	PC88	PC89
Explained Variance	0.329978	0.097623	0.081345	0.046354	0.037518	0.034485	0.032007	0.029605	0.024029	0.022536	...	0.000020	0.000010	0.000009	0.000004
Cumulative proportion	0.329978	0.427601	0.508945	0.555299	0.592817	0.627303	0.659309	0.688914	0.712943	0.735479	...	0.999976	0.999986	0.999994	0.999999

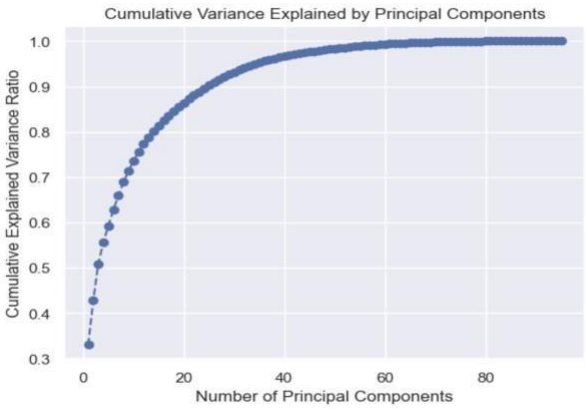
2 rows × 95 columns

```
Cumulative Explained Variance:
[0.32997777 0.42760058 0.50894514 0.55529911 0.59281719 0.62730254
 0.65930928 0.68891427 0.712943   0.73547851 0.75539601 0.773529
 0.78763355 0.80145745 0.81346642 0.82485356 0.83524031 0.84550428
 0.85504625 0.86398851 0.87227275 0.88026987 0.88784535 0.89514504
 0.90235393 0.90914879 0.91537267 0.92138531 0.9266519  0.93170943
 0.93654501 0.94097647 0.94521724 0.94922423 0.95280638 0.95612055
 0.95900746 0.96159475 0.96413366 0.96639838 0.96852609 0.97054835
 0.97250321 0.97433213 0.97599785 0.97755792 0.97909126 0.98055583
 0.98195421 0.98326539 0.98448715 0.98568249 0.98684988 0.98791888
 0.98894584 0.98988725 0.9907431  0.99156417 0.99235303 0.99307465
 0.99376908 0.99441907 0.99499696 0.99552765 0.99603316 0.99648407
 0.99688517 0.99724213 0.9975775  0.99789915 0.99819792 0.99848199
 0.99873029 0.99896555 0.9991675  0.99934529 0.99946707 0.99958006
 0.99967643 0.99975281 0.99980854 0.99985983 0.99989514 0.9999285
 0.99995594 0.99997597 0.99998558 0.99999425 0.99999854 1.
 1.         1.         1.         1.         1.        ]

Number of Principal Components to Retain (99% Variance): 57
```

3. We also plotted the explained variance and Cumulative variance below:





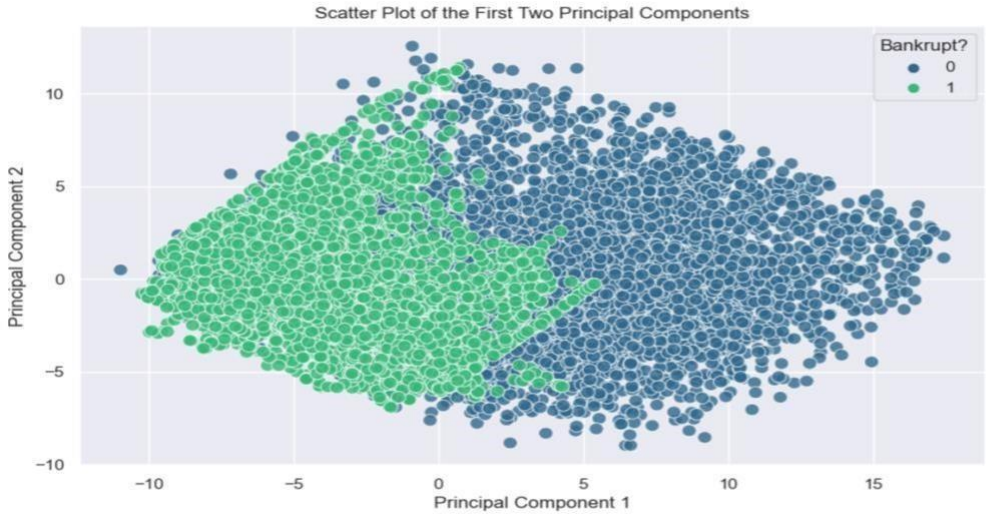


A considerable portion of the variance in the data may be explained by the first few principal components, but for higher-order components, this quickly diminishes. This indicates that most of the information is present in the first several components, and there would be little benefit to adding more.

The plot displays an elbow around the tenth principal component, which is where the curve's slope drastically varies. This shows that, to balance the trade-off between complexity and information loss, 10 components are a suitable number to preserve.

Additionally, the graphic demonstrates that about 40 primary components are required to account for 90% of the variance in the data. This indicates that nearly all the information in the can be represented by 40 components.

4. Below, we've plotted the first two principal components to assess the potential clustering of the target variable.



The figure displays two alternative color-coded clusters of data points that indicate whether a company is bankrupt (1) or not (0). This suggests that the companies can be efficiently divided by the first two main components according to their bankruptcy state, which may be a helpful predictor or indication of financial performance. Companies that are not in bankruptcy are represented by the green dots, and those that are represented by the blue dots. Although there is considerable overlap, the two groups usually form separate clusters. This implies that each group of companies shares certain traits or patterns, although the groups also differ or vary from one another.

About 50% of the variance in the data is explained by the first two principal components, which is a rather significant percentage when compared to other components. This indicates that the most significant information or patterns in the data are captured by these two components.

A few outliers are also depicted in the plot for both groups, particularly for the bankrupt group. These are entities whose principal component values deviate significantly from those of most of their group. These might be the result of extraordinary work, fraud, mistakes in accounting, or other causes.

Feature Selection:

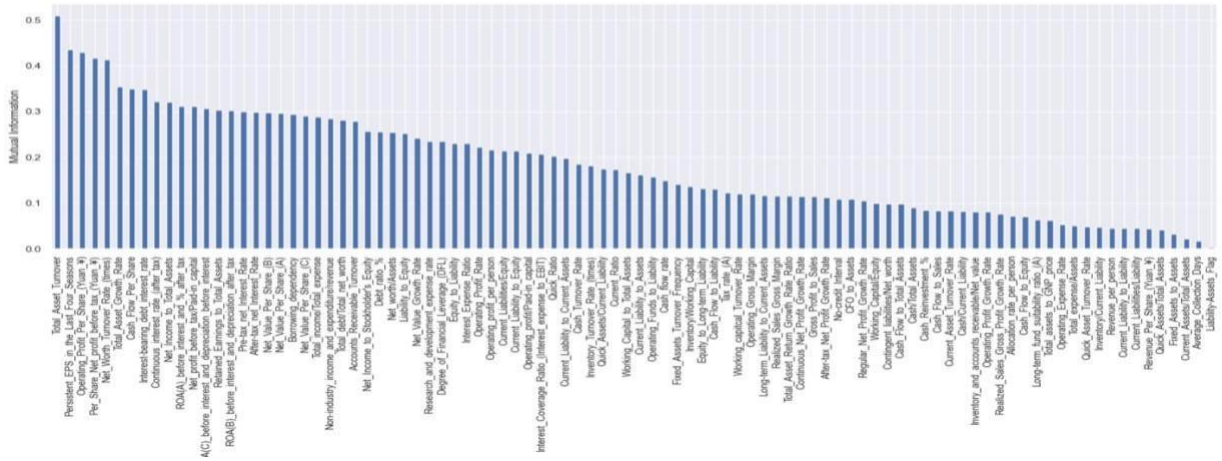
Feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model construction**.**

We have used 2 methods for feature selection:

1. **mutual_info_classIf:**

This is a scikit-learn function that is used to choose features for classification tasks. In order to determine which characteristics are the most informative, it computes the mutual information between each feature and the target variable. Greater relevance to predicting the goal is indicated by higher scores. It helps lessen overfitting and enhance model performance.

```
ROA(C)_before_interest_and_depreciation_before_interest    0.306667
ROA(A)_before_interest_and_%_after_tax                     0.311776
ROA(B)_before_interest_and_depreciation_after_tax          0.302143
Operating_Gross_Margin                                     0.120062
Realized_Sales_Gross_Margin                                0.116089
                                                              ...
Net_Income_to_Stockholder's_Equity                         0.257302
Liability_to_Equity                                        0.252666
Degree_of_Financial_Leverage_(DFL)                         0.235128
Interest_Coverage_Ratio_(Interest_expense_to_EBIT)         0.206933
Equity_to_Liability                                        0.230833
Length: 94, dtype: float64
```



Mutual Information values between all the features and the target variable is calculated. Suppose if we have smaller mutual information value, then we can get less information about the target from the feature.

The above histogram illustrates that there are few features which have higher mutual values and few others have almost zero mutual information values. The features are sorted on the basis of mutual information value.

```
Total_Asset_Turnover                                          0.507582
Operating_Profit_Per_Share_(Yuan_¥)                           0.442241
Persistent_EPS_in_the_Last_Four_Seasons                       0.439793
Per_Share_Net_profit_before_tax_(Yuan_¥)                      0.422140
Net_Worth_Turnover_Rate_(times)                               0.413793
Total_Asset_Growth_Rate                                       0.361964
Cash_Flow_Per_Share                                           0.353012
Interest-bearing_debt_interest_rate                           0.349873
Net_Income_to_Total_Assets                                    0.328633
Continuous_interest_rate_(after_tax)                          0.320707
Net_profit_before_tax/Paid-in_capital                         0.319105
ROA(A)_before_interest_and_%_after_tax                        0.313249
ROA(C)_before_interest_and_depreciation_before_interest       0.312195
ROA(B)_before_interest_and_depreciation_after_tax             0.310894
Net_Value_Per_Share_(B)                                       0.308777
Net_Value_Per_Share_(A)                                       0.306581
Retained_Earnings_to_Total_Assets                             0.306500
Net_Value_Per_Share_(C)                                       0.301838
After-tax_net_Interest_Rate                                   0.300899
Pre-tax_net_Interest_Rate                                     0.300785
Borrowing_dependency                                          0.297051
Total_income/Total_expense                                    0.291313
Non-industry_income_and_expenditure/revenue                   0.280258
Total_debt/Total_net_worth                                    0.274782
Accounts_Receivable_Turnover                                  0.274012
Net_Income_to_Stockholder's_Equity                            0.269404
Net_worth/Assets                                              0.258060
Debt_ratio_%                                                  0.257876
Net_Value_Growth_Rate                                         0.252162
Research_and_development_expense_rate                         0.248187
Liability_to_Equity                                           0.246498
Degree_of_Financial_Leverage_(DFL)                            0.236376
Interest_Expense_Ratio                                        0.231655
Equity_to_Liability                                           0.231339
Operating_Profit_Rate                                         0.227043
Operating_profit/Paid-in_capital                              0.218336
Operating_profit_per_person                                   0.215518
Interest_Coverage_Ratio_(Interest_expense_to_EBIT)            0.215116
Current_Liability_to_Equity                                   0.211643
Current_Liabilities/Equity                                    0.209742
dtype: float64
```

In the provided list of the top 40 columns based on mutual information scores, it's notable that the "Total Asset Turnover" column stands out with a score of 0.5, indicating its high relevance to the target variable. Following closely are "Operating Profit per Share" and "Persistent EPS" columns, each with mutual information scores around 0.44, suggesting their significant association with the target variable.

Considering a threshold of 0.2, we'll prioritize selecting columns with scores exceeding this value. This means that only features with mutual information scores above 0.2 will be considered important for our analysis, allowing us to focus on the most relevant variables for predicting the target variable.

This approach ensures that we concentrate on features with stronger associations with the target variable, streamlining our analysis and potentially improving the performance of our predictive model.

2. **SelectKBest**:

SelectKBest is particularly useful when dealing with high-dimensional datasets, where selecting a subset of relevant features can enhance model efficiency and prevent overfitting. By scoring each feature individually and selecting the top k features with the highest scores, SelectKBest allows practitioners to streamline their analysis and focus on the most impactful predictors. It's important to experiment with different values of k and scoring functions to optimize model performance for a given task. Overall, SelectKBest serves as a valuable tool in feature selection, aiding in the creation of more accurate and interpretable machine learning models.

As our data is high dimensional, we have implemented SelectKBest method.

```
from sklearn.feature_selection import SelectKBest
sel_ = SelectKBest(mutual_info_classif, k=40).fit(train_X, train_y)
print('\n Important features in dataset')
display(train_X.columns[sel_.get_support()])
```

```
 Important features in dataset

Index(['ROA(C)_before_interest_and_depreciation_before_interest',
       'ROA(A)_before_interest_and_%_after_tax',
       'ROA(B)_before_interest_and_depreciation_after_tax',
       'Operating_Profit_Rate', 'Pre-tax_net_Interest_Rate',
       'After-tax_net_Interest_Rate',
       'Non-industry_income_and_expenditure/revenue',
       'Continuous_interest_rate_(after_tax)',
       'Research_and_development_expense_rate',
       'Interest-bearing_debt_interest_rate', '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',
       'Operating_Profit_Per_Share_(Yuan_¥)',
       'Per_Share_Net_profit_before_tax_(Yuan_¥)', 'Total_Asset_Growth_Rate',
       'Net_Value_Growth_Rate', 'Interest_Expense_Ratio',
       'Total_debt/Total_net_worth', 'Debt_ratio_%', 'Net_worth/Assets',
       'Borrowing_dependency', 'Operating_profit/Paid-in_capital',
       'Net_profit_before_tax/Paid-in_capital', 'Total_Asset_Turnover',
       'Accounts_Receivable_Turnover', 'Net_Worth_Turnover_Rate_(times)',
       'Operating_profit_per_person', 'Current_Liabilities/Equity',
       'Retained_Earnings_to_Total_Assets', 'Total_income/Total_expense',
       'Current_Liability_to_Equity', 'Net_Income_to_Total_Assets',
       'Net_Income_to_Stockholder's_Equity', 'Liability_to_Equity',
       'Degree_of_Financial_Leverage_(DFL)',
       'Interest_Coverage_Ratio_(Interest_expense_to_EBIT)',
       'Equity_to_Liability'],
      dtype='object')
```

The highlighted features represent the subset deemed crucial for our dataset. Remarkably, many of these features align with those identified using the mutual_info_classif selection method. In employing the SelectKBest technique, we opted for a value of k equal to 40, suggesting a strategic choice aimed at capturing a comprehensive yet manageable set of predictors. This approach ensures that our model focuses on the most informative features, enhancing its predictive accuracy and interpretability.

Conclusion of Feature Engineering (Dimension Reduction & Feature Selection):

In the process of feature engineering and selection, we explored multiple techniques to enhance the predictive power and efficiency of our model.

1. Principal Component Analysis (PCA) was utilized to tackle the dimensionality of our dataset, comprising 95 feature variables and 1 target variable. Remarkably, employing PCA revealed that a subset of 57 principal components captures 99% of the variance in our data. Additionally, we visualized the cumulative and explained variance ratios via an elbow plot, providing insights into the optimal number of principal components required for effective dimensionality reduction.

2. We employed the mutual_info_classif feature selection technique, which computes the mutual information between each feature and the target variable. Setting a threshold of 0.2, we identified approximately 40 columns demonstrating high mutual information with the target. This approach allowed us to focus on the most relevant features for predictive modeling.

3. The SelectKBest feature selection method was also applied, with k set to 40. Intriguingly, a significant overlap was observed between the features selected by mutual_info_classif and SelectKBest, reinforcing the importance of the identified features.

In deliberating between these techniques, mutual_info_classif emerged as the preferred method. Not only did it effectively reduce the dimensionality of our feature space, but it also exhibited considerable consistency with the SelectKBest approach. Furthermore, the flexibility to adjust the threshold in mutual_info_classif based on performance metrics in subsequent classification tasks adds an additional layer of adaptability to our feature selection process.

By leveraging mutual_info_classif as our primary feature selection method, we can confidently streamline our model's input space while ensuring that only the most informative features are retained, ultimately contributing to improved model performance and interpretability.

Recommended list of features:
Below are the columns or features selected using the mutual_info_classif feature selection method that most accurately describe our target variable 'Bankrupt?'.

```
Total_Asset_Turnover                                        0.507582
Operating_Profit_Per_Share_(Yuan_¥)                         0.442241
Persistent_EPS_in_the_Last_Four_Seasons                     0.439793
Per_Share_Net_profit_before_tax_(Yuan_¥)                    0.422140
Net_Worth_Turnover_Rate_(times)                             0.413793
Total_Asset_Growth_Rate                                     0.361964
Cash_Flow_Per_Share                                         0.353012
Interest-bearing_debt_interest_rate                         0.349873
Net_Income_to_Total_Assets                                  0.328633
Continuous_interest_rate_(after_tax)                        0.320707
Net_profit_before_tax/Paid-in_capital                       0.319105
ROA(A)_before_interest_and_%_after_tax                      0.313249
ROA(C)_before_interest_and_depreciation_before_interest     0.312195
ROA(B)_before_interest_and_depreciation_after_tax           0.310894
Net_Value_Per_Share_(B)                                     0.308777
Net_Value_Per_Share_(A)                                     0.306581
Retained_Earnings_to_Total_Assets                           0.306500
Net_Value_Per_Share_(C)                                     0.301838
After-tax_net_Interest_Rate                                 0.300899
Pre-tax_net_Interest_Rate                                   0.300785
Borrowing_dependency                                        0.297051
Total_income/Total_expense                                  0.291313
Non-industry_income_and_expenditure/revenue                 0.280258
Total_debt/Total_net_worth                                  0.274782
Accounts_Receivable_Turnover                                0.274012
Net_Income_to_Stockholder's_Equity                          0.269404
Net_worth/Assets                                            0.258060
Debt_ratio_%                                                0.257876
Net_Value_Growth_Rate                                       0.252162
Research_and_development_expense_rate                       0.248187
Liability_to_Equity                                         0.246498
Degree_of_Financial_Leverage_(DFL)                          0.236376
Interest_Expense_Ratio                                      0.231655
Equity_to_Liability                                         0.231339
Operating_Profit_Rate                                       0.227043
Operating_profit/Paid-in_capital                            0.218336
Operating_profit_per_person                                 0.215518
Interest_Coverage_Ratio_(Interest_expense_to_EBIT)          0.215116
Current_Liability_to_Equity                                 0.211643
Current_Liabilities/Equity                                  0.209742
dtype: float64
```

As we keep working, we know it's important to change the threshold values based on how well our classification models are doing. This helps us fine-tune our feature selection method over time, so we find the right balance between including features and making sure our model works well. By checking how different threshold values affect our model's performance, we can adjust our approach to make sure we're capturing the most important patterns in the data and making our model more reliable.

Data Mining Models

In our predictive analysis, we employed a total of **seven** models: Logistic Regression, KNN, Decision Tree, Random Forest, Naive Bayes, SVM, and XGBoost. The predictor variables were collectively denoted as **'Variable X'**, while the target variable **'bankrupt'** was represented by **'Variable y'**. We partitioned the entire dataset into a training dataset (used for training the classification models) and a validation dataset (used to evaluate the classification performance of these models). Specifically, **80%** of the data was allocated for training, and the remaining **20%** served as validation data.

The dataset has too many attributes, so we used mutual_info_classIf method to narrow it down to the top 10 representative variables based on mutual information between the variables and the target before building the model.

Furthermore, we also performed **'Cross Validation'** and **'Hyperparameter Tuning'** techniques on all the 7 models. A key method for evaluating a machine learning model's performance and generalization capacity is cross-validation. The primary objective is to evaluate the model's performance on training data that has not yet been seen. The popular CV techniques are –

‘Kfold’ and ‘**Stratified K-Fold’.** The dataset is divided into K folds of equal size in K-Fold crossvalidation. One-fold is tested during each iteration, while the remaining K-1 folds are used for training. To enhance the performance of the model, hyperparameter tuning entails determining the ideal values for these parameters. We can determine the optimal combination by testing various hyperparameter settings on validation sets. Techniques like **grid search** and **random search** explore different hyperparameter values systematically. GridSearchCV Uses crossvalidation to assess model performance for every combination of parameters as it thoroughly searches a predetermined grid of hyperparameters. Although computationally costly for larger parameter spaces, it’s also suitable for smaller ones. RandomizedSearchCV investigates a subset of hyperparameters by sampling hyperparameter settings at random from predefined distributions. Larger search spaces benefit from its increased efficiency, which delivers decent results at a lower computational cost but does not ensure that the ideal hyperparameters are found.

Model Implementation

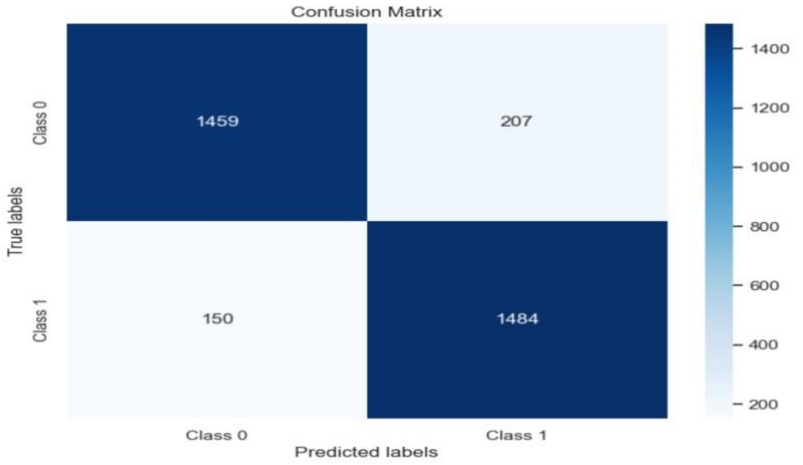
1. **Logistic Regression:**

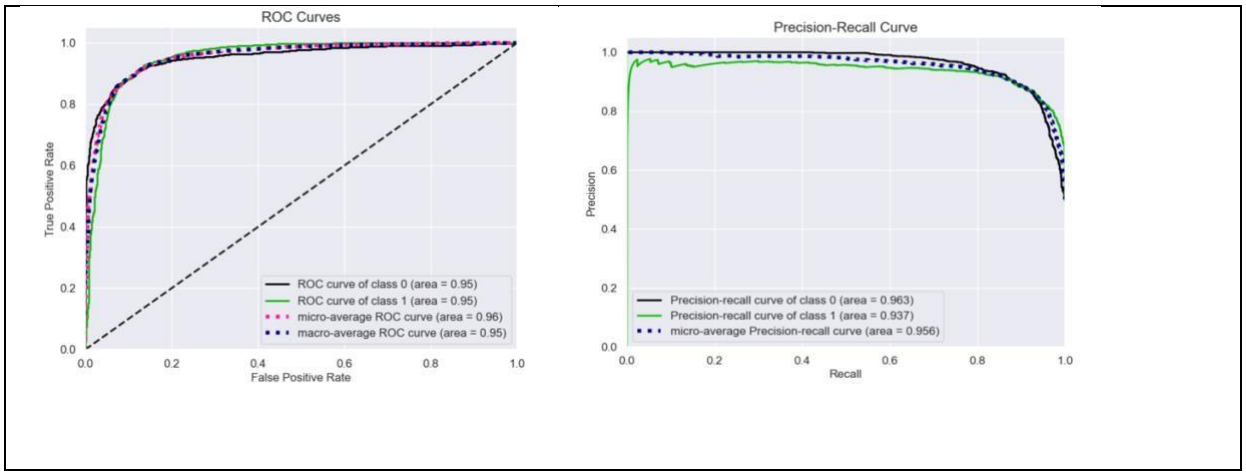
   A popular statistical technique in data mining and machine learning is logistic regression. It is especially well-suited for problems involving binary classification, in which the objective is to forecast one of two likely outcomes (e.g., true/false, bankrupt/not bankrupt, yes/no). The logistic function, also called the sigmoid function, is used by the logistic regression model to convert a linear combination of predictor variables into a probability. The probability of the positive class is represented by the function's output, which ranges from 0 to 1. Labeled data—where the target variable is known—are used to train logistic regression. Metrics like recall, accuracy, and precision are then used to assess the model, as is the area under the Receiver Operating Characteristic (ROC) curve.

```
Use Logistic Regression to evaluate on the validation set
auc: 0.9546179548930955
accuracy score: 0.8918181818181818
Classification Report:
              precision    recall  f1-score   support

           0      0.907     0.876     0.891      1666
           1      0.878     0.908     0.893      1634

    accuracy                          0.892      3300
   macro avg      0.892     0.892     0.892      3300
weighted avg      0.892     0.892     0.892      3300
```

```
K-fold Cross Validation

Logistic Regression: Mean Accuracy = 0.898363166786613, Std Deviation = 0.003435451396991009


Stratified K-fold Cross Validation

Logistic Regression: Mean Accuracy = 0.8980603406475058, Std Deviation = 0.005952541878567572
```

```
Hyperparameter tuning with grid serach

Best parameters for Logistic Regression: {'C': 0.1}
Best cross-validation score for Logistic Regression: 0.899475349758321


Hyperparameter tuning with random serach

Best parameters for Logistic Regression: {'C': 0.1}
Best cross-validation score for Logistic Regression: 0.899475349758321
```

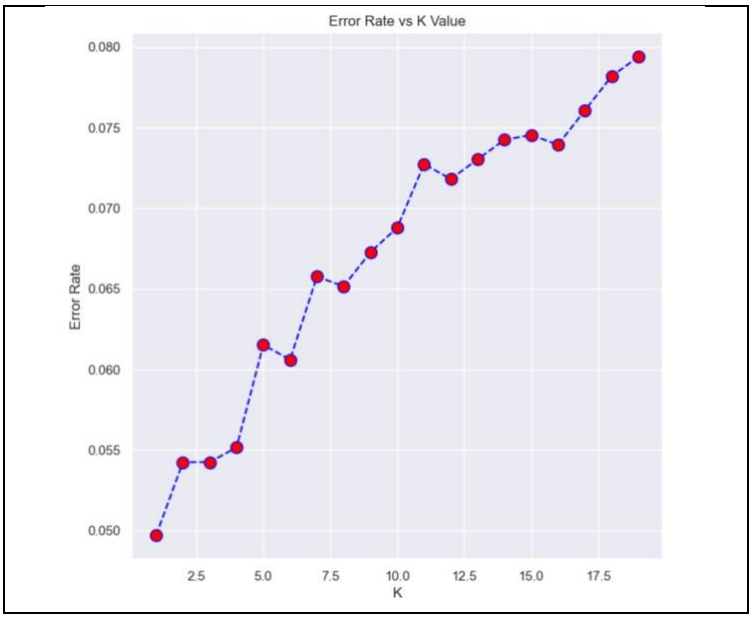




- The logistic regression model's f1-score is 0.891 and its accuracy is 89%, both of which are rather high. Furthermore, with an AUC value of 0.95, the ROC curve is at the upperleft corner, indicating that the model performs admirably. Furthermore, the PR curve is entirely concentrated on positive samples, and class 0, or the line of interest, is near the upper-right corner, with a value of 0.963.
- The average accuracy scores obtained by K-fold Cross Validation and Stratified K-fold Cross Validation for Logistic Regression are similar. The average accuracy score of the Stratified K-fold approach is somewhat higher. However, it also reveals a little bit greater variance in the accuracy ratings obtained from other testing.
- For the Logistic Regression model, the optimal parameter value (C=1) was found using both the grid search and random search approaches. The model performed well with varying parameter choices, as indicated by the best cross-validation score of about 0.899.

2. KNN Classifier:

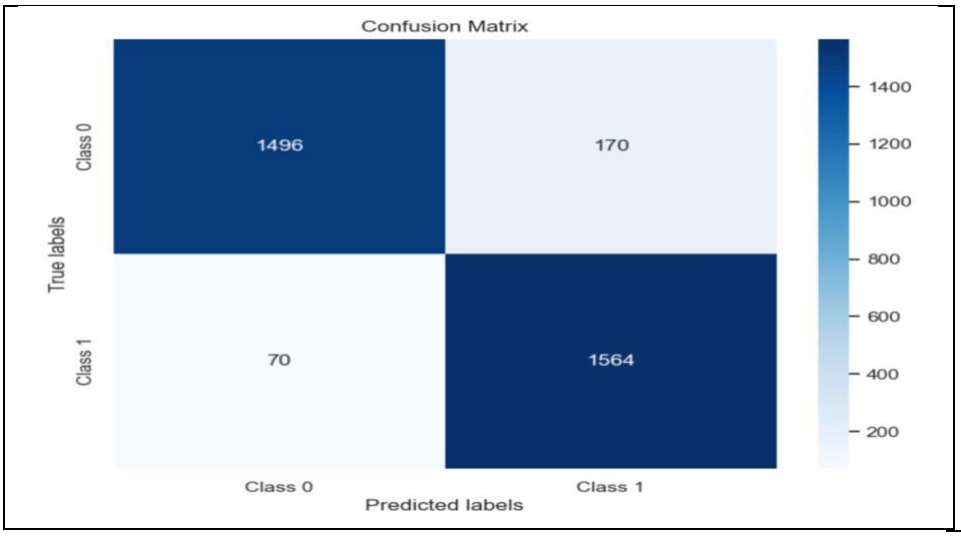
KNN is a robust and intuitive algorithm which builds its predictions on how close together the data points are. KNN attempts to categorize new, unclassified data points given a collection of training data, sometimes called previous data, where each data point has attributes (features) and relates to a certain class label. To be more precise, we figure out how far the unclassified point is from every other point in the training set (such as the Euclidean distance). The class label applied to the unclassified point is determined by the majority class among these K neighbors.

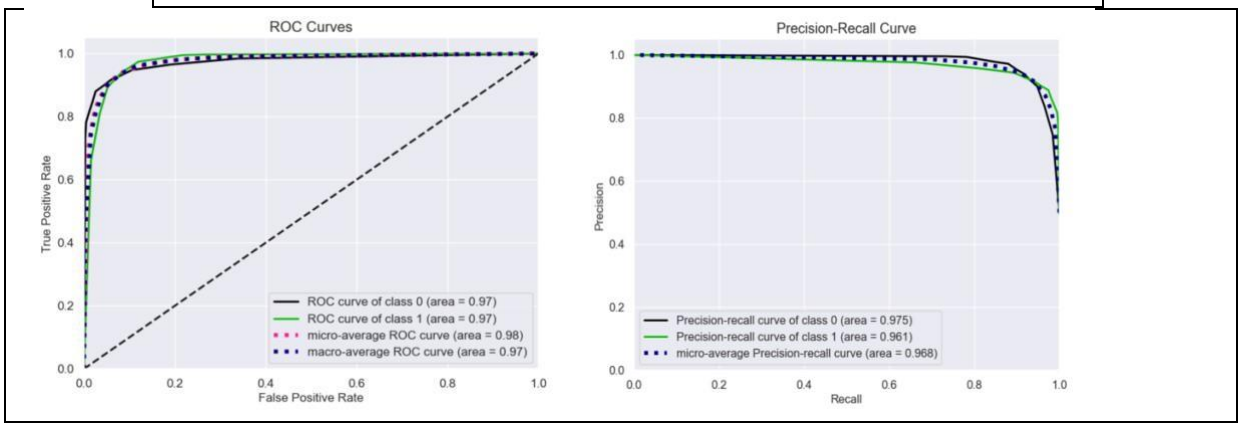



```
Use KNN to evaluate on the validation set
auc: 0.9749015885423937
accuracy score: 0.9272727272727272
Classification Report:
              precision    recall  f1-score   support

           0      0.955     0.898     0.926      1666
           1      0.902     0.957     0.929      1634

    accuracy                          0.927      3300
   macro avg      0.929     0.928     0.927      3300
weighted avg      0.929     0.927     0.927      3300
```

```
K-fold Cross Validation

KNN: Mean Accuracy = 0.9334208175744305, Std Deviation = 0.0016132804795578007


Stratified K-fold Cross Validation

KNN: Mean Accuracy = 0.9334203582056032, Std Deviation = 0.00403035005667781
```

```
Hyperparameter tuning with grid serach

Best parameters for KNN: {'n_neighbors': 3}
Best cross-validation score for KNN: 0.93796785439029


Hyperparameter tuning with random serach

Best parameters for KNN: {'n_neighbors': 3}
Best cross-validation score for KNN: 0.93796785439029
```





- We selected the optimal k=11 in the KNN model. The error rate appears to be minimized at an ideal K value. To achieve accurate forecasts, this ideal K value is essential. With a f1-score of 0.931 and an accuracy of 93%, the KNN model outperforms the logistic regression model. With an AUC value of 0.97, the ROC curve is in the top-left corner, indicating that the model performs better than the logistic regression model. Additionally, the PR curve is entirely focused on positive samples, and class 0, or the line of interest, has a value of 0.975.
- With a high mean accuracy of 0.933 and a comparatively small standard deviation, both cross-validation approaches show that the KNN model performs consistently well across various data splits, indicating its dependability in classification tasks.
- The best parameter value of 3 neighbors was consistently discovered by the K-Nearest Neighbors (KNN) model following hyperparameter tuning using both grid search and random search strategies. The KNN model's best cross-validation score, which was roughly 0.937, showed that it performed well while adjusting across a range of parameter settings.

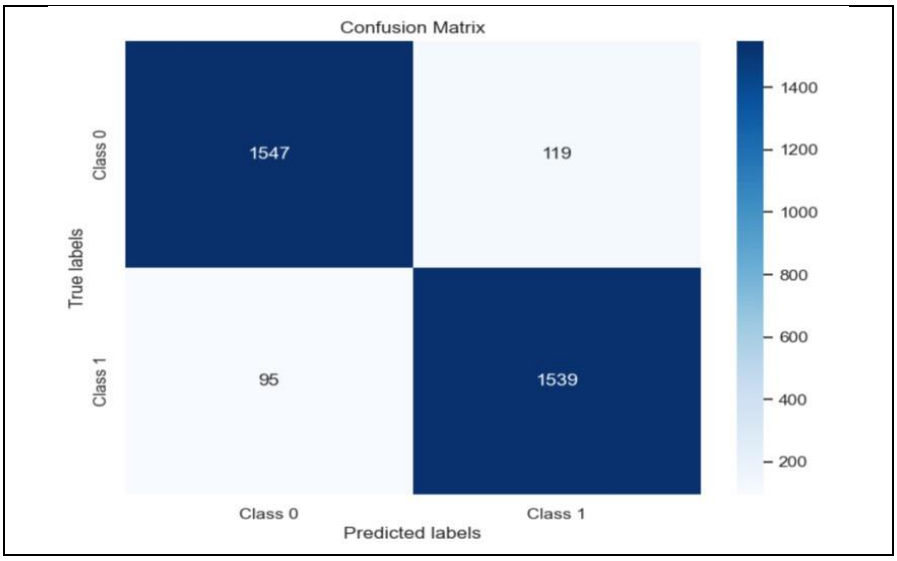
3. Decision Tree:

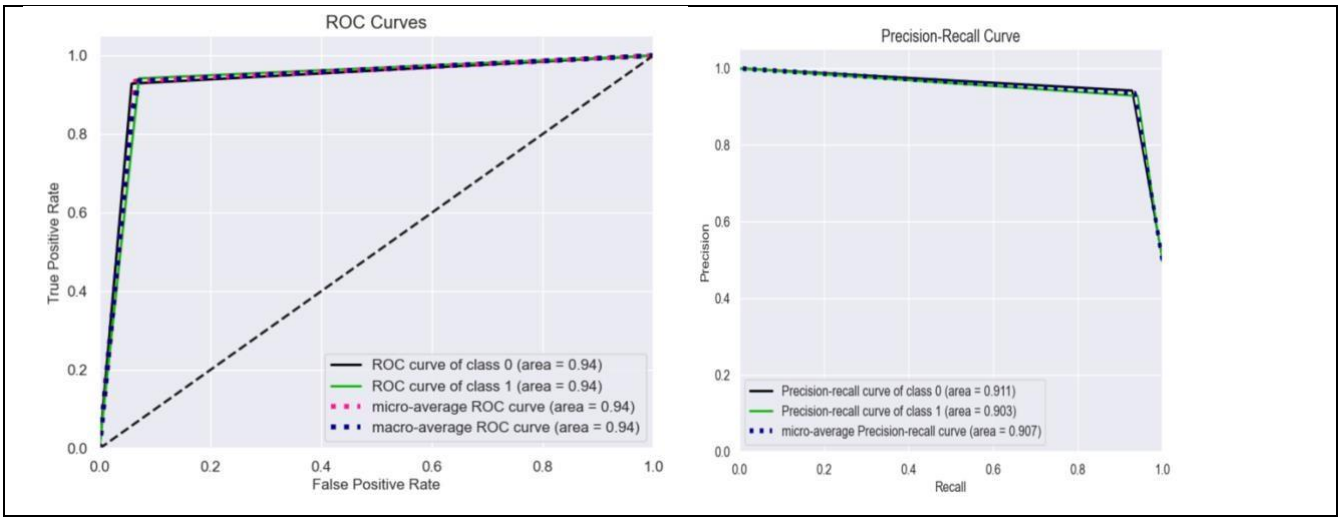
In supervised learning algorithms, decision trees are an effective tool for both regression and classification problems. It builds a tree structure that resembles a flowchart, with each internal node denoting a test on an attribute, each branch denoting a test result, and each leaf node (terminal node) holding a class label or a numerical value. The training data is divided recursively into subsets according to attribute values to construct a decision tree. They are essential for comprehending difficult decision boundaries and producing accurate forecasts.

```
Use DecisionTree to evaluate on the validation set
auc: 0.9352159468438538
accuracy score: 0.9351515151515152
Classification Report:
              precision    recall  f1-score   support

           0      0.942     0.929     0.935      1666
           1      0.928     0.942     0.935      1634

    accuracy                          0.935      3300
   macro avg      0.935     0.935     0.935      3300
weighted avg      0.935     0.935     0.935      3300
```

```
K-fold Cross Validation

Decision Tree: Mean Accuracy = 0.9305914118445701, Std Deviation = 0.004678088125389497


Stratified K-fold Cross Validation

Decision Tree: Mean Accuracy = 0.9336233992272396, Std Deviation = 0.0039006070534627125
```

```
Hyperparameter tuning with grid serach

Best parameters for Decision Tree: {'max_depth': 20, 'min_samples_split': 2}
Best cross-validation score for Decision Tree: 0.9303911780768779


Hyperparameter tuning with random serach

Best parameters for Decision Tree: {'min_samples_split': 2, 'max_depth': 20}
Best cross-validation score for Decision Tree: 0.9306943104618698
```





- The decision tree model has a greater accuracy (93%), and a f1-score (0.935) than the logistic regression model. Additionally, with an AUC value of 0.94, the ROC curve is near the diagonal, indicating that the model performs as expected. Furthermore, the PR curve is entirely concentrated on positive samples, and class 0, or the line of interest, has a value of 0.911.
- With a high mean accuracy of 0.93 and a comparatively low standard deviation, both cross-validation procedures show that the Decision Tree model performs consistently well across various data splits, indicating its reliability in classification tasks.
- Following the application of grid search and random search strategies to hyperparameter tuning, the Decision Tree model consistently found the following optimal parameter values: 'max_depth': 20 and 'min_samples_split': 2. During the tuning phase, the Decision Tree model demonstrated good performance across

various parameter values, as seen by its best cross-validation score of roughly 0.930.

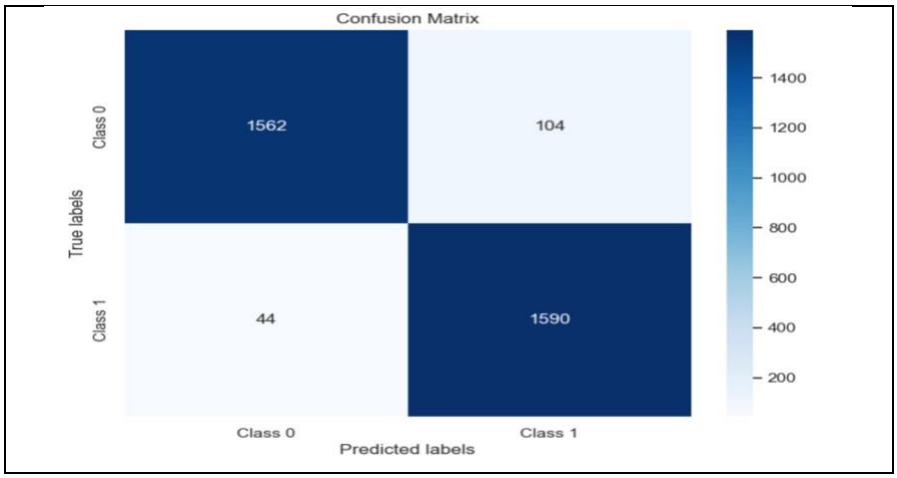
4. Random Forest:

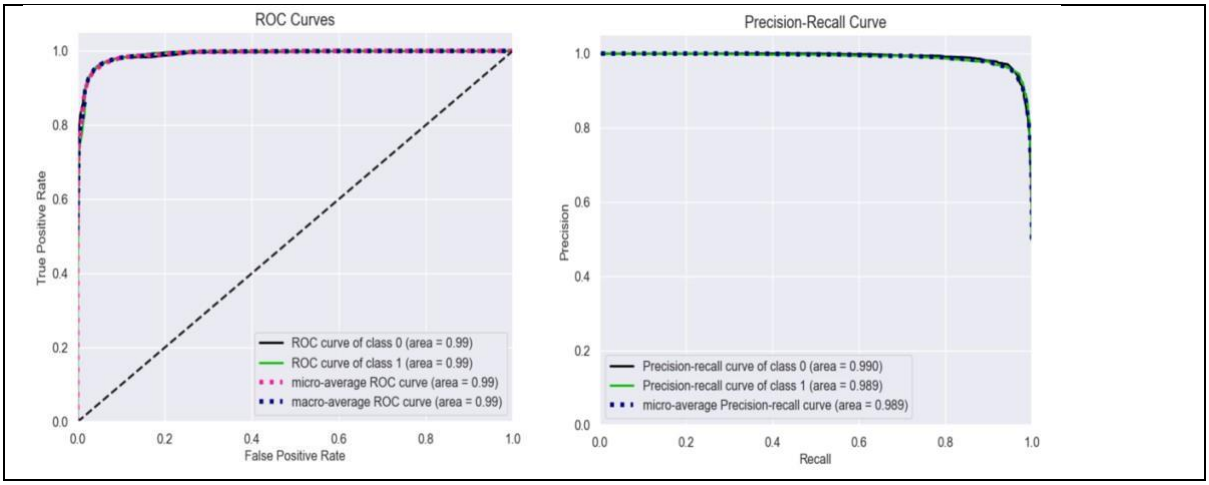
A collection of decision trees is called Random Forest. Every decision tree is trained using a bootstrap sample, and the final forecast considers the forecasts of every single tree. Instead of looking for the greatest predictor among all predictors, a random forest may look for the best predictor to divide among a random sample of predictors, adding another degree of randomization. To grow the trees, extremely randomized trees may select a random split point as opposed to looking for the best splitting point for a predictor.

```
Use RandomForest to evaluate on the validation set
auc: 0.9898772850633522
accuracy score: 0.9551515151515152
Classification Report:
              precision    recall  f1-score   support

           0      0.973     0.938     0.955      1666
           1      0.939     0.973     0.956      1634

    accuracy                          0.955      3300
   macro avg      0.956     0.955     0.955      3300
weighted avg      0.956     0.955     0.955      3300
```

```
K-fold Cross Validation

Random Forest: Mean Accuracy = 0.953425105016818, Std Deviation = 0.0027713672751812535


Stratified K-fold Cross Validation

Random Forest: Mean Accuracy = 0.9516060555019624, Std Deviation = 0.004988710784571448
```

```
Hyperparameter tuning with grid serach

Best parameters for Random Forest: {'max_depth': 20, 'n_estimators': 300}
Best cross-validation score for Random Forest: 0.9530214218996432


Hyperparameter tuning with random serach

Best parameters for Random Forest: {'n_estimators': 100, 'max_depth': 20}
Best cross-validation score for Random Forest: 0.9524153102526018
```





- In comparison to the logistic regression model, the random forest model has a 95% accuracy rate and a f1-score of 0.955, both of which are rather high. Furthermore, with an AUC value of 0.99, the ROC curve is at the upper-left corner, indicating that the model performs admirably. Furthermore, the PR curve is entirely concentrated on positive samples, and class 0, or the line of interest, is near the upper-right corner, with a value of 0.990.
- Both cross-validation techniques demonstrate that the RandomForest model performs consistently well across different data splits, with high mean accuracy of around 0.95 and relatively low standard deviation, suggesting its reliability in classification tasks.
- The optimal values for grid search were determined to be 'max_depth': 20 and 'n_estimators': 300, resulting in a cross-validation score of roughly 0.952. Conversely, random search yielded similar cross-validation scores of about 0.953 and determined that the ideal parameters were 'max_depth': 20 and 'n_estimators': 100. These findings imply that a moderate tree depth and a larger number of estimators are optimal for the RandomForest model's performance.

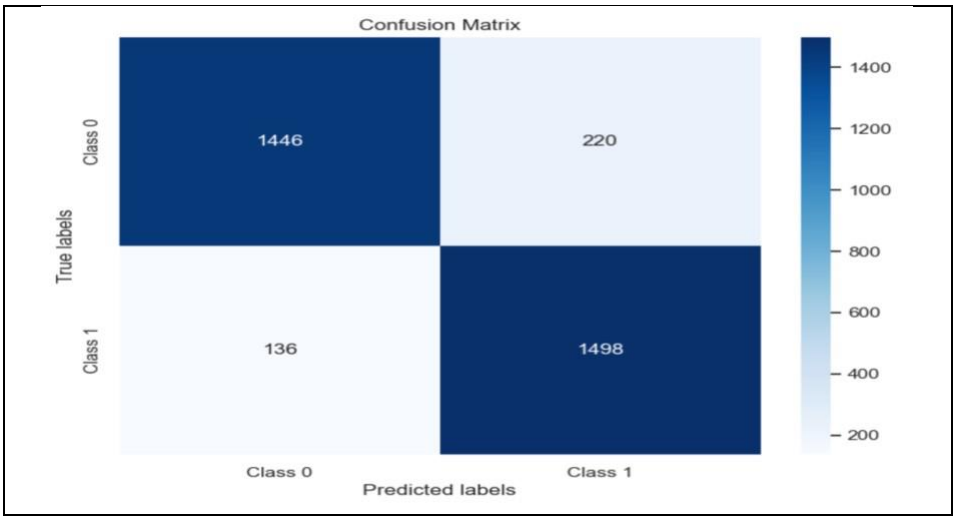
5. **SVM:**

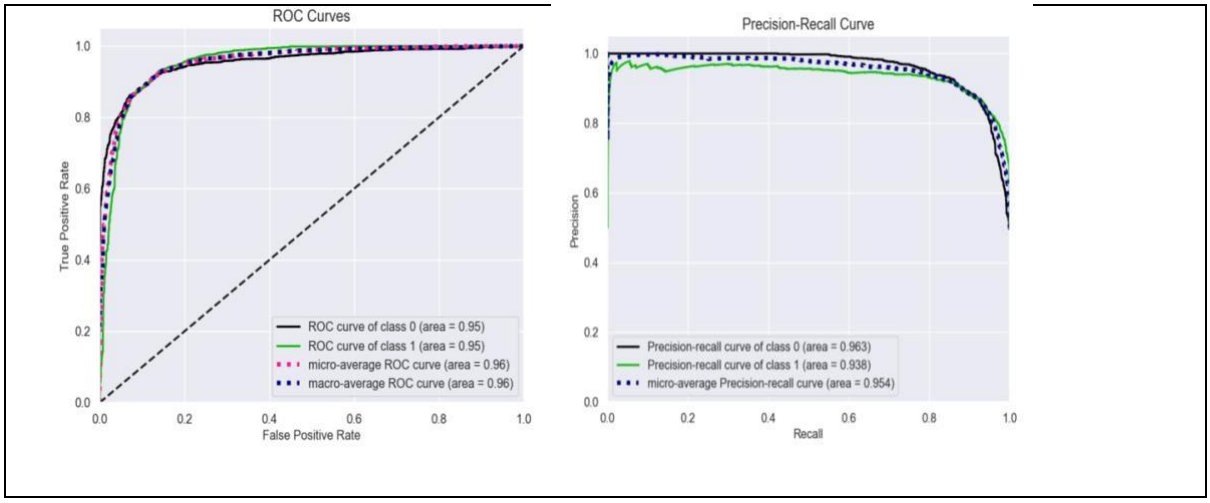
Support vector machines are learning algorithms and supervised learning models that are used for regression and classification data processing. The SVM training algorithm builds a model that assigns new instances to one of the two classes given a set of training instances that are all designated as belonging to one or the other of the two classes; this makes the model a non-probabilistic two Meta linear classifier. Instances are represented by the SVM model as points in space, and the mapping makes sure that instances of different classes are as far apart as is reasonably detectable. Then, based on which side of the interval the new instances fall on, map them to the same space and determine which class they belong to.

```
Use SVM to evaluate on the validation set
auc: 0.9547582803011045
accuracy score: 0.8921212121212121
Classification Report:
              precision    recall  f1-score   support

           0       0.914     0.868     0.890      1666
           1       0.872     0.917     0.894      1634

    accuracy                          0.892      3300
   macro avg       0.893     0.892     0.892      3300
weighted avg       0.893     0.892     0.892      3300
```

```
K-fold Cross Validation

SVM: Mean Accuracy = 0.9122042557969794, Std Deviation = 0.0036671696245686243


Stratified K-fold Cross Validation

SVM: Mean Accuracy = 0.9121024800812572, Std Deviation = 0.007946099040294852
```

```
Hyperparameter tuning with grid serach

Best parameters for SVM: {'C': 10, 'gamma': 'scale'}
Best cross-validation score for SVM: 0.9233189908177275


Hyperparameter tuning with random serach

Best parameters for SVM: {'gamma': 'scale', 'C': 10}
Best cross-validation score for SVM: 0.9233189908177275
```





- The SVM model's f1-score is 0.890, like the logistic regression model, and its accuracy is 89%. With an AUC value of 0.95, the ROC curve is near the diagonal, indicating that the model performs as expected. Furthermore, the PR curve is entirely concentrated on positive samples, and class 0, or the line of interest, is near the upper-right corner, with a value of 0.963.
- The SVM model performs consistently while using various cross-validation strategies. The model shows consistent performance across folds in K-fold cross-validation, with a mean accuracy of roughly 0.912 and a reduced standard deviation of approximately 0.003. Similarly, the SVM model retains a little higher standard deviation of around 0.007 and a mean accuracy of roughly 0.912 in stratified K-fold cross-validation, indicating significantly less variability in performance across folds.
- The SVM model performed best with the following parameters after hyperparameter tweaking using both grid search and random search techniques: a regularization parameter (C) of 10 and a gamma value of "scale". The robustness and consistency of the optimization process were demonstrated by the identical cross-validation scores and optimal parameters obtained from the grid search and random search techniques. The model's

enhanced performance upon tuning is reflected in the best cross-validation score of roughly 0.923, highlighting the significance of choosing the right hyperparameters to increase the efficacy of SVM models.

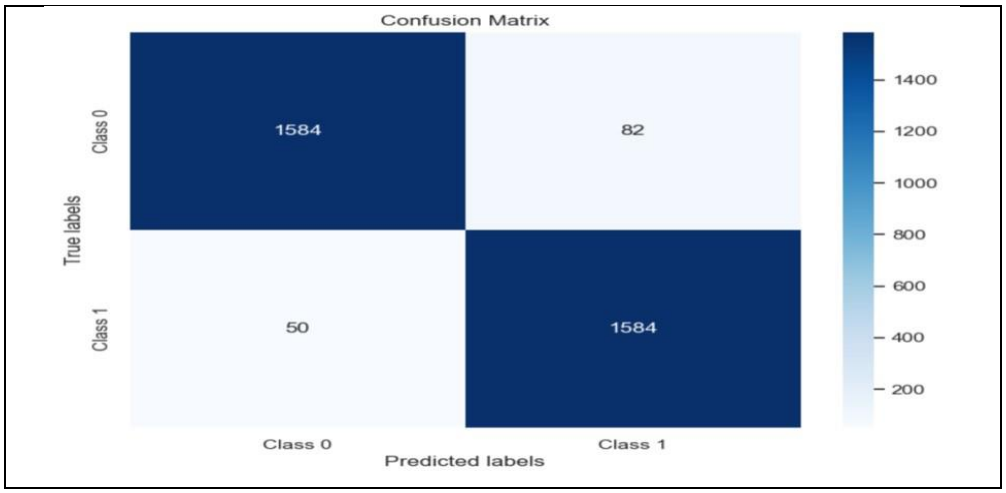
6. **XGBoost:**

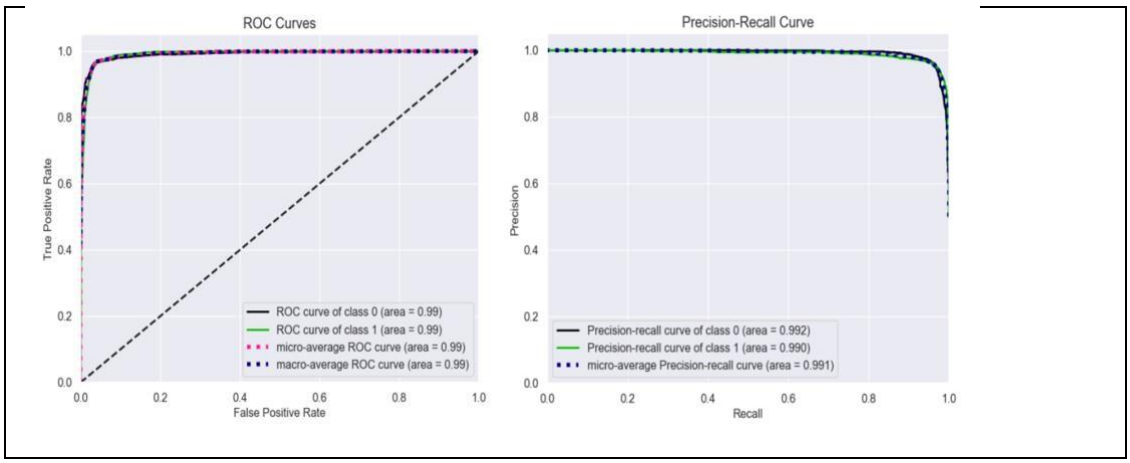
One of the boosting methods is called XGBoost. The goal of the boosting algorithm is to create a strong classifier by combining multiple weak classifiers. As a boosted tree model, XGBoost combines several tree models to create a powerful classifier. CART regression tree model is the model being employed.

```
Use XGBoost to evaluate on the validation set
auc: 0.9911277240394321
accuracy score: 0.96
Classification Report:
              precision    recall  f1-score   support

           0      0.969     0.951     0.960      1666
           1      0.951     0.969     0.960      1634

    accuracy                          0.960      3300
   macro avg      0.960     0.960     0.960      3300
weighted avg      0.960     0.960     0.960      3300
```

```
K-fold Cross Validation

XGBoost: Mean Accuracy = 0.9571634485328271, Std Deviation = 0.002392598883523044


Stratified K-fold Cross Validation

XGBoost: Mean Accuracy = 0.9566574282491412, Std Deviation = 0.003639891503736746
```

```
Hyperparameter tuning with grid serach

Best parameters for XGBoost: {'max_depth': 7, 'n_estimators': 200}
Best cross-validation score for XGBoost: 0.9589819876378745


Hyperparameter tuning with random serach

Best parameters for XGBoost: {'n_estimators': 200, 'max_depth': 7}
Best cross-validation score for XGBoost: 0.9589819876378745
```





- The XGBoost model has a 96% accuracy rate and a f1-score of 0.960, which are both relatively high. Additionally, the model performs far better with an AUC value of 0.99 and a ROC curve that is at the top-left corner. Furthermore, the PR curve is entirely concentrated on positive samples, and class 0, or the line of interest, is near the upper-right corner, with a value of 0.992.
- The XGBoost model exhibits consistent and high performance across different cross-validation techniques. With a higher mean accuracy of 0.95 and minimal variance, XGBoost model generalizes well to unseen data and is robust against variations in the training data, making it a reliable choice for bankruptcy prediction tasks.
- Grid search and random search methods were used to optimize the hyperparameters of the XGBoost model. The two methods reached comparable optimal parameter configurations; the combination of 200 estimators and a maximum depth of 7 was found to be the most effective. With these parameters, the appropriate cross-validation score for XGBoost was roughly 0.958, suggesting good performance.

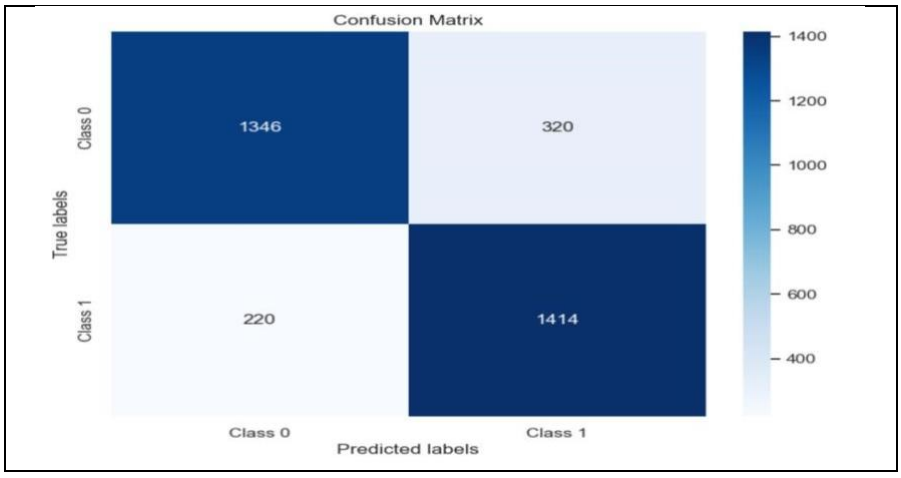
7. **Naïve Bayes:**

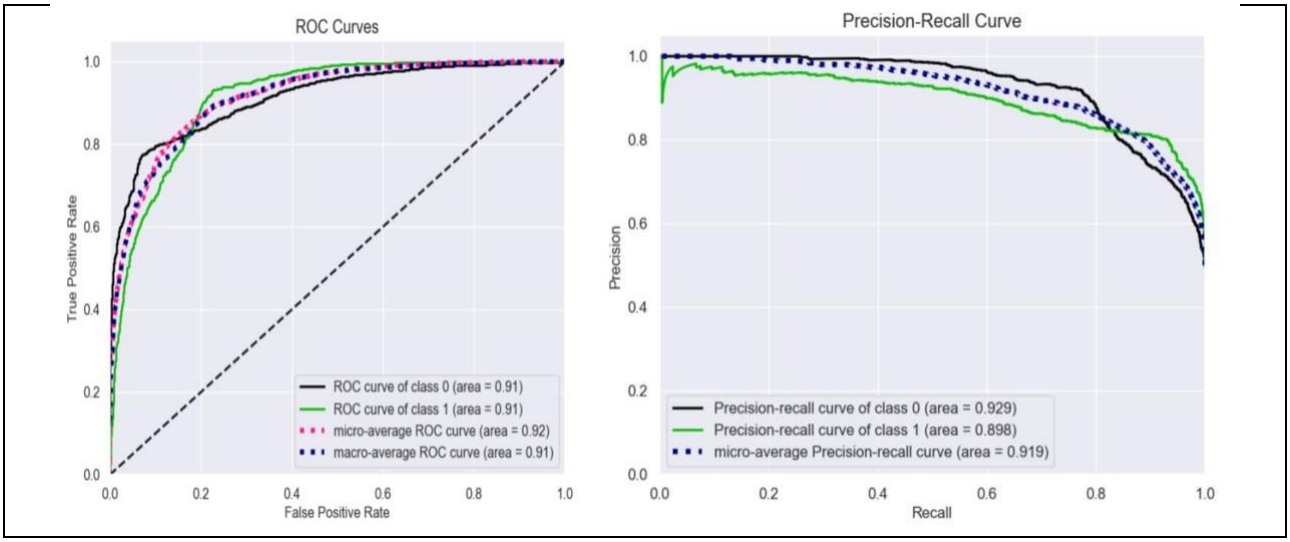
A probabilistic machine learning approach called Naive Bayes is useful for many different types of categorization problems. The premise that the features employed in the model are independent of one another is referred to as "naive." Stated differently, altering the value of one feature does not immediately affect or modify the values of any other features that are incorporated into the algorithm.

```
Use Naive Bayes to evaluate on the validation set
auc: 0.9145546835625316
accuracy score: 0.8363636363636363
Classification Report:
              precision    recall  f1-score   support

           0      0.860     0.808     0.833      1666
           1      0.815     0.865     0.840      1634

    accuracy                          0.836      3300
   macro avg      0.837     0.837     0.836      3300
weighted avg      0.838     0.836     0.836      3300
```

```
K-fold Cross Validation:
Gaussian Naive Bayes: Mean Accuracy = 0.8404716697036051, Std Deviation = 0.011079541094425846


Stratified K-fold Cross Validation:
Gaussian Naive Bayes: Mean Accuracy = 0.8404730478100866, Std Deviation = 0.006150154385218358
```

```
Hyperparameter tuning with grid search:
Best parameters: {'var_smoothing': 1e-09}
Best cross-validation score: 0.8404735582198948


Hyperparameter tuning with random search:
Best parameters: {'var_smoothing': 1e-09}
Best cross-validation score: 0.8404735582198948
```





- The Naive Bayes model has an 83% accuracy rate and a low f1-score of 0.833. Furthermore, the model's AUC value of 0.91 indicates that it performs poorly, as the ROC curve is around the top-left corner. Furthermore, the PR curve is entirely concentrated on positive samples, and class 0, or the line of interest, is near the upper-right corner, with a value of 0.929.
- Gaussian Naive Bayes in K-fold cross-validation yielded an approximate mean accuracy of 84% and a standard deviation of 0.011. Similar results were obtained with stratified K-fold cross-validation, where the mean accuracy was roughly 84% with a standard deviation of roughly 0.006. The stability of the model's predictions is demonstrated by these findings, which show constant performance across many folds.
- Both grid search and random search approaches produced the same optimal settings for Gaussian Naive Bayes after hyperparameter tuning: {'var_smoothing': 1e-09}. The model's performance did not considerably increase with hyperparameter modification, as indicated by its corresponding best cross-validation score of about 84%.

Summary of all Model's Performance

	Model	Recall	Accuracy	Precision	ROC AUC	F1-score
0	Logistic Regression	0.876	0.891	0.907	0.954	0.891
1	Random Forest	0.938	0.955	0.973	0.989	0.955
2	Decision Tree	0.929	0.935	0.942	0.935	0.935
3	KNN	0.898	0.927	0.955	0.974	0.926
4	SVM	0.868	0.892	0.914	0.954	0.890
5	XGBoost	0.951	0.960	0.969	0.991	0.960
6	Naive Bayes	0.808	0.836	0.860	0.914	0.833

After comparing all the models based on Recall, Accuracy, Precision, ROC AUC and F1-Score, we sought the following conclusion—

1. **XGBoost** has the highest accuracy (0.960) and F1-score (0.960), indicating their robustness and effectiveness in predicting the target variable.
2. **Random Forest** also performs exceptionally well with an accuracy of higher 0.955 and F1-score of 0.955.
3. **Naive Bayes** has the lowest scores in all metrics except ROC AUC.
4. **KNN** excels in recall (0.898), but has slightly lower precision compared to Random Forest and XGBoost:
5. **Logistic Regression** also performs reasonably well across all metrics, making it a suitable choice when interpretability of the model is important.
6. **SVM** shows decent performance but lags behind Random Forest and XGBoost in terms of recall and precision.
7. **Decision Tree** exhibits slightly lower performance compared to Random Forest and XGBoost but still provide competitive results, especially in terms of recall and F1-score.

Best Fit Model Recommendation

Based on the provided metrics and performance values, XGBoost emerges as the preferred choice for the final model. Here's why:

- Accuracy: XGBoost has the highest accuracy score of 0.960 among all the models. This indicates that it correctly predicts the class labels for most of the instances in the validation set.
- Recall: XGBoost has the second-highest recall score of 0.951, indicating its ability to correctly identify the positive class (bankruptcy) instances. High recall is desirable as it means fewer instances of bankruptcy are missed.
- Precision: XGBoost has a precision score of 0.969, which is among the highest. This indicates that when it predicts a company to be bankrupt, it is correct 96% of the time. High precision is desirable as it reduces the number of false positive predictions.
- ROC AUC: XGBoost achieves the highest ROC AUC score of 0.991, indicating excellent performance in distinguishing between bankrupt and non-bankrupt companies. A higher ROC AUC score suggests better model performance.
- F1-score: XGBoost also has the highest F1-score of 0.960, which is a harmonic mean of precision and recall. It provides a balance between precision and recall, making it a robust choice for classification tasks.

Overall, considering its superior performance across multiple metrics, XGBoost is selected as the final model for bankruptcy prediction.

Conclusion

Determining if a Bank is bankrupt or not was the primary goal of this project. When this problem was solved using real-world data, it was discovered that the number of non-bankrupts is far more than that of bankrupt. The class of interest was thus the one that was represented by label 0. Our project can predict a company's likelihood of going bankrupt more accurately to some extent.

To find out the best fit model, we evaluated all the models with various evaluation metrics. We observed that XGBoost has the highest AUC score, the highest F1-score, the highest accuracy, and the highest precision of all the evaluation indicators. The capacity to accurately classify observations into their respective classes is represented by accuracy, which is the proportion of correctly classified records; precision, on the other hand, is the fraction of observations that the model classifies as positive observations are class of interest. The F1-score can determine the best possible balance between recall and precision. Taking everything into account, we concluded that XGBoost is the most appropriate classification model for our data.