Kandivli Education Society's
B. K. SHROFF COLLEGE OF ARTS &
M. H. SHROFF COLLEGE OF COMMERCE
An Autonomous College | NAAC Re-accredited 'A' Grade
ISO 9001 : 2015 Certified • 'Best College 2017-18' award from University of Mumbai

ASSIGNMENT

IN THE MAJOR COURSE OF

Machine Learning

'Bankruptcy Prediction Model'

BY

CHAUDHARI RAJVI YATIN

MSC DS -I

FDMSCDS001

SEMESTER **II**

UNDER THE GUIDANCE OF

ASST.PROF. MEGHNA SINGH

ACADEMIC YEAR

2023-2024

Introduction:

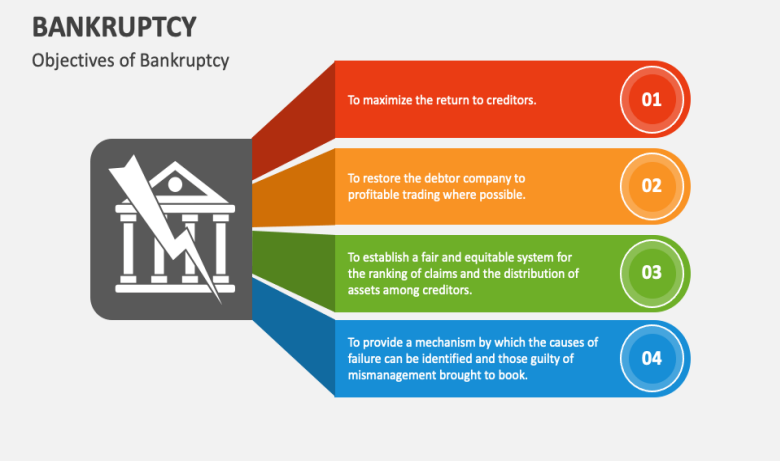
Bankruptcy Prevention

Bankruptcy is a legal status of an individual or entity that cannot repay their debts to creditors. It is a legal process through which individuals and businesses that are unable to meet their financial obligations can seek relief from some or all of their debts. The primary goals of bankruptcy are to provide a fresh start for debtors and to ensure fair treatment of creditors.

The bankruptcy process begins with a petition filed by the debtor, which is most common, or on behalf of creditors, which is less common. All of the debtor's assets are measured and evaluated, and the assets may be used to repay a portion of the outstanding debt.

Bankruptcy offers an individual or business a chance to start fresh by forgiving debts that they can't pay. Meanwhile, creditors have a chance to get some repayment based on the individual's or business's assets available for liquidation.

This is a classification based project.



Libraries:

- Pandas:

  Pandas provides easy-to-use data structures, such as Series and DataFrame, along with a variety of functions to manipulate and analyze structured data. Here it is widely used for data analysis, data cleaning and transformation, indexing and selection.

- Numpy:

  NumPy (Numerical Python) provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

- matplotlib.pyplot:

  Matplotlib.pyplot is a sub-module that provides a MATLAB-like interface for creating static, interactive, and animated visualizations. Here data is visualized using pie chart and histogram.

- %matplotlib inline:

  It is a magic command in Jupyter Notebooks that allows matplotlib plots to be displayed directly in notebook output, convinent for visualization.

- Seaborn:

  Seaborn provides a high-level interface for creating informative and attractive statistical graphics and it is particularly well-suited for exploring and understanding complex datasets with multiple variables. Here data is visualized using countplot and correlation is visualized using heatmap.

- Sklearn:

  It provides a wide range of tools for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, and more. Scikit-learn is built on NumPy, SciPy, and Matplotlib, and it is designed to be simple and efficient, making it a popular choice among machine learning practitioners.

- Warnings:

  It is a Python module that provides a mechanism for issuing warning messages during program execution. It allows developers to display warnings for potential issues without causing the program to terminate.

Module:

- sklearn.model_selection:

  sklearn.model_selection is a module in the scikit-learn library (often referred to as sklearn) that provides tools for model selection and evaluation.

- sklearn.linear_model:

It is a module in scikit-learn, a machine learning library in Python. This module provides implementations of linear models, including linear regression, logistic regression, and other linear classifiers. It's widely used for various supervised learning tasks.

- sklearn.metrics :

  It implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values.

Methods:

- bank.head():

  It is a method used to display the first few rows of a DataFrame.
- bank.tail():

  It is a method used to display the first few rows of a DataFrame.
- bank.info():

  It is a method in pandas provides a concise summary of the DataFrame, including the data types of each column, the number of non-null values, and memory usage. It is a quick way to get an overview of the structure and completeness of the dataset.
- bank.describe():

  It is a method that generates descriptive statistics of the numerical columns in the DataFrame, including measures of central tendency, dispersion, and shape of the distribution.

- bank.columns:
  the columns attribute can be used to get the column labels of the DataFrame.
- bank.skew():
  The skew() method calculates the skew for each column. By specifying the column axis (axis='columns'), the skew() method searches column-wise and returns the skew of each row.

Expression:

- bank.isnull().sum():
  It is an expression is used to check the number of missing values in each column of the DataFrame. It returns a Series with the count of null values for each column.

Function:

- LogisticRegression:
  Basically, it measures the relationship between the categorical dependent variable and one or more independent variables by estimating the probability of occurrence of an event using its logistics function. It used to implement logistic regression.
- confusion_matrix:
  Confusion matrix is a very popular measure used while solving classification problems. It can be applied to binary classification as well as for multiclass classification problems.
- accuracy_score:
  This package assigns subset accuracy in multi-label classification. It is required that the labels the model has predicted for the given sample and the true labels of the sample match exactly. Accuracy describes the model's behaviour across all classes.

- train_test_split:
  train_test_split is commonly used for splitting a dataset into training and testing sets, allowing you to evaluate the performance of a machine learning model on unseen data.

Algorithm:

Logistic regression

Logistic regression is a simple and more efficient method for binary and linear classification problems. It is a classification model, which is very easy to realize and achieves very good performance with linearly separable classes. It is an extensively employed algorithm for classification in industry.

Visualizations:

1. Count Plot:

A Count Plot is a bar plot where the count of occurrences of each unique value in a categorical variable is depicted.

2. Heatmap:

A Heatmap is a data visualization technique where values in a matrix are represented as colors. It is useful for visualizing relationships between two variables in a matrix format.

3. Distplot:

A Distplot or distribution plot, depicts the variation in the data distribution. Seaborn Distplot represents the overall distribution of continuous data variables. The Seaborn module along with the Matplotlib module is used to depict the distplot with different variations in it.

Skewness : Skewness is a measurement of the distortion of symmetrical distribution or asymmetry in a data set. Skewness is demonstrated on a bell curve when data points are not distributed symmetrically to the left and right sides of the median on a bell curve.

Confusion matrix : A confusion matrix is a table used in machine learning and statistics to assess the performance of a classification model. It summarizes the results of classification by showing the counts of true positive, true negative, false positive, and false negative predictions.

Accuracy score : Accuracy score in machine learning is an evaluation metric that measures the number of correct predictions made by a model in relation to the total number of predictions made. We calculate it by dividing the number of correct predictions by the total number of predictions.

Code:

Importing necessary libraries:

In [8]:
```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
logisticclassifier = LogisticRegression()
import warnings
warnings.filterwarnings('ignore')
```

Importing the Dataset:

In [9]:
```
#reading dataset
bank = pd.read_csv(r"C:\Users\neetac1973\Downloads\bankruptcy-prevention1.csv")
```

Exploring the Dataset:

In [10]:
```
bank
```

Out[10]:

	industrial_risk	management_risk	financial_flexibility	credibility	competitiveness	operating_risk	class
0	0.5	1.0	0.0	0.0	0.0	0.5	bankruptcy
1	0.0	1.0	0.0	0.0	0.0	1.0	bankruptcy
2	1.0	0.0	0.0	0.0	0.0	1.0	bankruptcy
3	0.5	0.0	0.0	0.5	0.0	1.0	bankruptcy
4	1.0	1.0	0.0	0.0	0.0	1.0	bankruptcy
...	...	...	...	...	...	...	...
245	0.0	1.0	1.0	1.0	1.0	1.0	non-bankruptcy
246	1.0	1.0	0.5	1.0	1.0	0.0	non-bankruptcy
247	0.0	1.0	1.0	0.5	0.5	0.0	non-bankruptcy
248	1.0	0.0	0.5	1.0	0.5	0.0	non-bankruptcy
249	1.0	0.0	0.5	0.5	1.0	1.0	non-bankruptcy

250 rows × 7 columns

In [11]:
```
bank.head()
```

Out[11]:

	industrial_risk	management_risk	financial_flexibility	credibility	competitiveness	operating_risk	class
0	0.5	1.0	0.0	0.0	0.0	0.5	bankruptcy
1	0.0	1.0	0.0	0.0	0.0	1.0	bankruptcy
2	1.0	0.0	0.0	0.0	0.0	1.0	bankruptcy
3	0.5	0.0	0.0	0.5	0.0	1.0	bankruptcy
4	1.0	1.0	0.0	0.0	0.0	1.0	bankruptcy

It is a method used to display the first few rows of a DataFrame. It is a quick way to inspect the structure and contents of the dataset.

```
In [12]: 1 bank.columns #gives columns name

Out[12]: Index(['industrial_risk', 'management_risk', 'financial_flexibility',
                'credibility', 'competitiveness', 'operating_risk', 'class'],
               dtype='object')
```

```
In [13]: 1 bank.describe() #gives description of dataset
```

Out[13]:

	industrial_risk	management_risk	financial_flexibility	credibility	competitiveness	operating_risk
count	250.000000	250.000000	250.000000	250.000000	250.000000	250.000000
mean	0.518000	0.614000	0.376000	0.470000	0.476000	0.570000
std	0.411526	0.410705	0.401583	0.415682	0.440682	0.434575
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.500000	0.000000	0.000000	0.000000	0.000000
50%	0.500000	0.500000	0.500000	0.500000	0.500000	0.500000
75%	1.000000	1.000000	0.500000	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

It is the method used to show the columns name. And describe method is used to give the description of the dataset.

```
In [28]: 1 bank.info() #gives information about dataset

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   industrial_risk        250 non-null    float64
 1   management_risk        250 non-null    float64
 2   financial_flexibility  250 non-null    float64
 3   credibility            250 non-null    float64
 4   competitiveness        250 non-null    float64
 5   operating_risk         250 non-null    float64
 6   class                  250 non-null    object
dtypes: float64(6), object(1)
memory usage: 13.8+ KB
```

Displaying a concise summary of a DataFrame, including information about the data types, non-null values, total number of entries (rows) and memory usage.

```
In [33]: 1 bank.dtypes

Out[33]: industrial_risk         float64
         management_risk         float64
         financial_flexibility   float64
         credibility             float64
         competitiveness         float64
         operating_risk          float64
         class                    object
         dtype: object
```

Counting the number of missing (null or NaN) values in each column of a DataFrame

```
In [30]: 1 bank.isnull().sum() #checking missing values in dataset

Out[30]: industrial_risk         0
         management_risk         0
         financial_flexibility   0
         credibility             0
         competitiveness         0
         operating_risk          0
         class                   0
         dtype: int64
```

Create a new copy of the DataFrame to perform required analysis and to avoid modifying the original data unintentionally.

```
In [16]: bank_new = bank.iloc[:,:] # used to select rows & columns from Series using integer-based indexing.
         bank_new
```

Out[16]:

	industrial_risk	management_risk	financial_flexibility	credibility	competitiveness	operating_risk	class
0	0.5	1.0	0.0	0.0	0.0	0.5	bankruptcy
1	0.0	1.0	0.0	0.0	0.0	1.0	bankruptcy
2	1.0	0.0	0.0	0.0	0.0	1.0	bankruptcy
3	0.5	0.0	0.0	0.5	0.0	1.0	bankruptcy
4	1.0	1.0	0.0	0.0	0.0	1.0	bankruptcy
...	...	...	...	...	...	...	...
245	0.0	1.0	1.0	1.0	1.0	1.0	non-bankruptcy
246	1.0	1.0	0.5	1.0	1.0	0.0	non-bankruptcy
247	0.0	1.0	1.0	0.5	0.5	0.0	non-bankruptcy
248	1.0	0.0	0.5	1.0	0.5	0.0	non-bankruptcy
249	1.0	0.0	0.5	0.5	1.0	1.0	non-bankruptcy

```
In [19]: bank.skew() #tells about skewness
```

```
Out[19]: industrial_risk          -0.067011
         management_risk          -0.444701
         financial_flexibility     0.479134
         credibility               0.112955
         competitiveness           0.093906
         operating_risk           -0.275547
         dtype: float64
```

It returns unbiased skew over requested axis Normalized by N-1.

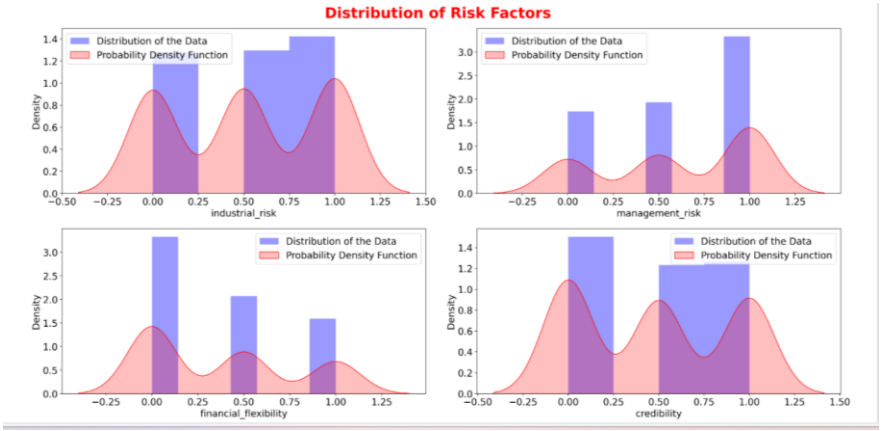
Visualization

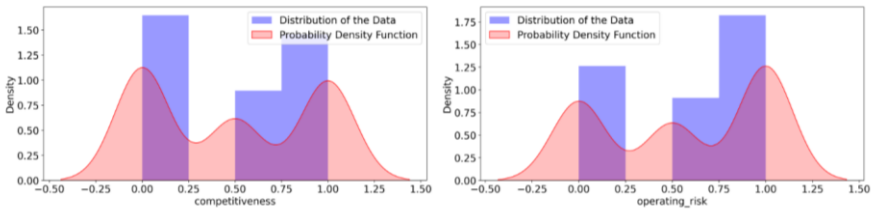
```
In [17]: #Visualization Plot settings

         plt.rcParams['figure.figsize'] = (20,15)
         plt.rcParams['figure.dpi'] = 350
         plt.rcParams['font.size'] = 16
```

```
In [18]: #Distribution plot for each column

         fig, ax = plt.subplots(nrows=3, ncols=2)
         fig.suptitle('Distribution of Risk Factors', fontsize = 25, fontweight = 'bold', color = 'red')
         sns.distplot(bank['industrial_risk'], ax=ax[0,0], kde_kws= {'fill':True, 'color':'red','label':'Probability Density Function
         ax[0,0].legend()
         sns.distplot(bank['management_risk'], ax=ax[0,1], kde_kws= {'fill':True, 'color':'red', 'label':'Probability Density Functio
         ax[0,1].legend()
         sns.distplot(bank['financial_flexibility'], ax=ax[1,0], kde_kws= {'fill':True, 'color':'red', 'label':'Probability Density F
         ax[1,0].legend()
         sns.distplot(bank['credibility'], ax=ax[1,1], kde_kws= {'fill':True, 'color':'red', 'label':'Probability Density Function'},
         ax[1,1].legend()
         sns.distplot(bank['competitiveness'], ax=ax[2,0], kde_kws= {'fill':True, 'color':'red', 'label':'Probability Density Functio
         ax[2,0].legend()
         sns.distplot(bank['operating_risk'], ax=ax[2,1], kde_kws= {'fill':True, 'color':'red', 'label':'Probability Density Function
         ax[2,1].legend()
         plt.tight_layout()
```





Observation: Here Mean of Industrial risk, Management risk (highly skewed), Operating risk are greater than median, so it is negatively skewed.

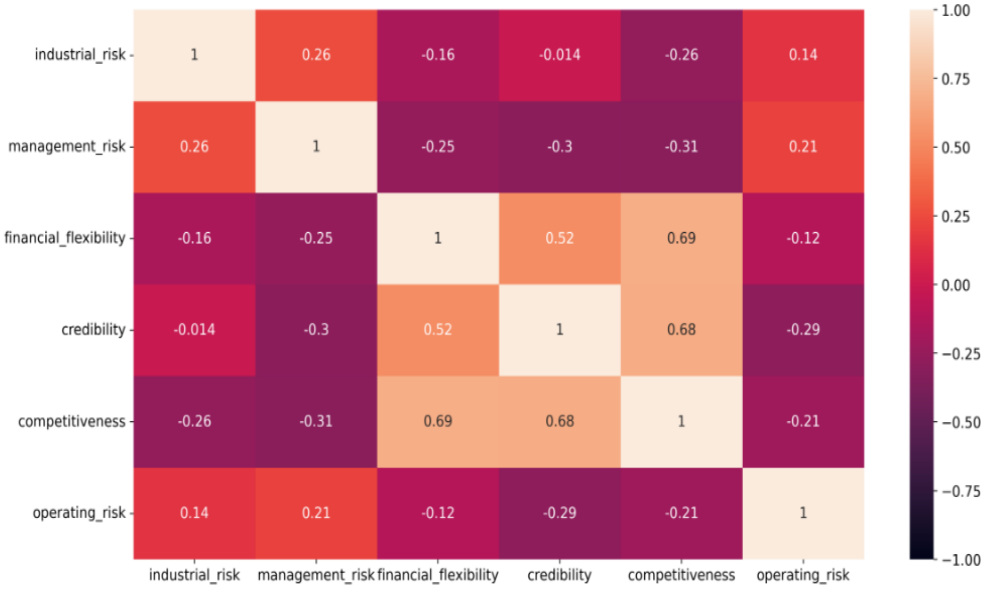
Median of Financial flexibility, credibility (highly skewed), Competitveness are greater than mean, so it is positively skewed.

plt.rcParams It's a dictionary of most matplotlib styling that you set at the start of your notebook and it will apply to all your plots. Printing out rcParams in a notebook you can see the key-value pair structure.

The purpose of using plt. figure() is to create a figure object. The whole figure is regarded as the figure object. It is necessary to explicitly use plt.

tight_layout automatically adjusts subplot params so that the subplot(s) fits in to the figure area. This is an experimental feature and may not work for some cases. It only checks the extents of ticklabels, axis labels, and titles.

```
In [21]: sns.heatmap(bank.corr(), vmin = -1, vmax = 1, annot = True)
Out[21]: <AxesSubplot:>
```



To show user behavior on specific web pages or webpage templates.

The target variable is the feature of a dataset that you want to understand more clearly. Changing the variable to target variable.

```
In [22]: np.shape(bank) # a tuple of integers
Out[22]: (250, 7)
```

```
In [23]: # Input
         x = bank.iloc[:,:-1]

         # Target variable
         y = bank.iloc[:,-1]
```

```
In [24]: x
Out[24]:
```

	industrial_risk	management_risk	financial_flexibility	credibility	competitiveness	operating_risk
0	0.5	1.0	0.0	0.0	0.0	0.5
1	0.0	1.0	0.0	0.0	0.0	1.0
2	1.0	0.0	0.0	0.0	0.0	1.0
3	0.5	0.0	0.0	0.5	0.0	1.0
4	1.0	1.0	0.0	0.0	0.0	1.0
...	...	...	...	...	...	...
245	0.0	1.0	1.0	1.0	1.0	1.0
246	1.0	1.0	0.5	1.0	1.0	0.0
247	0.0	1.0	1.0	0.5	0.5	0.0
248	1.0	0.0	0.5	1.0	0.5	0.0
249	1.0	0.0	0.5	0.5	1.0	1.0

250 rows × 6 columns

```
In [59]: y
Out[59]: 0      1
         1      1
         2      1
         3      1
         4      1
                ..
         245    1
         246    1
         247    1
         248    1
         249    1
         Name: class_yn, Length: 250, dtype: int64
```

Splitting the data into training and testing

```
In [22]:
#Splitting the data into training and testing
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.80, random_state= 0)
```

```
In [23]:
X_train
```

Out[23]:

	industrial_risk	management_risk	financial_flexibility	credibility	competitiveness	operating_risk
71	1.0	1.0	0.0	0.0	0.0	1.0
160	0.5	1.0	0.5	0.5	1.0	0.5
180	0.0	0.0	0.5	0.5	1.0	1.0
75	1.0	1.0	0.0	0.0	0.0	0.5
59	1.0	1.0	0.0	0.5	0.0	0.0
...	...	...	...	...	...	...
67	1.0	1.0	0.0	0.5	0.0	1.0
192	0.5	1.0	0.0	0.0	1.0	0.0
117	0.0	0.0	1.0	1.0	0.5	0.0
47	1.0	1.0	0.0	0.0	0.0	1.0
172	0.5	0.5	0.0	1.0	1.0	0.0

200 rows × 6 columns

```
In [24]:
X_test
```

Out[24]:

	industrial_risk	management_risk	financial_flexibility	credibility	competitiveness	operating_risk
225	0.0	0.0	1.0	1.0	0.5	0.0

```
In [25]:
y_train
```

```
Out[25]:
71          bankruptcy
160    non-bankruptcy
180    non-bankruptcy
75          bankruptcy
59          bankruptcy
              ...
67          bankruptcy
192    non-bankruptcy
117    non-bankruptcy
47          bankruptcy
172    non-bankruptcy
Name: class, Length: 200, dtype: object
```

```
In [26]:
y_test
```

```
Out[26]:
225    non-bankruptcy
122    non-bankruptcy
92          bankruptcy
157    non-bankruptcy
154    non-bankruptcy
161    non-bankruptcy
198    non-bankruptcy
83          bankruptcy
63          bankruptcy
155    non-bankruptcy
218    non-bankruptcy
231    non-bankruptcy
108    non-bankruptcy
186    non-bankruptcy
116    non-bankruptcy
73          bankruptcy
203    non-bankruptcy
139    non-bankruptcy
152    non-bankruptcy
```

Model building

Algogrithm used to predict the model is Logistic Regression

```
In [61]:
##Preparing models by Logistic Regression
logisticlassifier = LogisticRegression()
```

```
In [62]:
logisticlassifier
```

```
Out[62]: LogisticRegression()
```

```
In [64]:
logisticlassifier.fit(X_train, y_train)

logisticlassifier.coef_ # coefficients of features
```

```
Out[64]: array([[-0.47708154, -0.76388501,  2.44455271,  2.37453278,  3.75882044,
        -0.48513004]])
```

After training the model we predict the test data

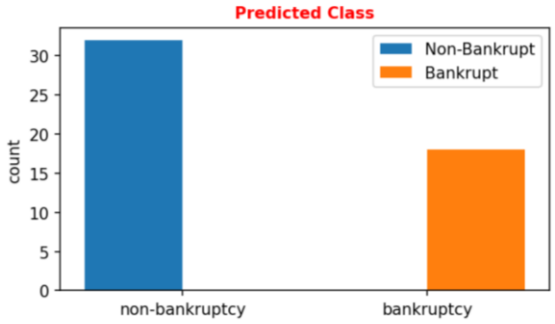
```
In [67]:
# after the training the model then we predict test data
y_pred = logisticlassifier.predict(X_test)
y_pred
```

```
Out[67]: array(['non-bankruptcy', 'non-bankruptcy', 'bankruptcy', 'non-bankruptcy',
       'non-bankruptcy', 'non-bankruptcy', 'non-bankruptcy', 'bankruptcy',
       'bankruptcy', 'non-bankruptcy', 'non-bankruptcy', 'non-bankruptcy',
       'non-bankruptcy', 'non-bankruptcy', 'non-bankruptcy', 'bankruptcy',
       'non-bankruptcy', 'non-bankruptcy', 'non-bankruptcy', 'bankruptcy',
       'non-bankruptcy', 'bankruptcy', 'non-bankruptcy', 'non-bankruptcy',
       'non-bankruptcy', 'bankruptcy', 'bankruptcy', 'bankruptcy',
       'non-bankruptcy', 'bankruptcy', 'non-bankruptcy', 'bankruptcy',
       'non-bankruptcy', 'non-bankruptcy', 'non-bankruptcy',
       'non-bankruptcy', 'non-bankruptcy', 'bankruptcy', 'bankruptcy',
       'non-bankruptcy', 'non-bankruptcy', 'bankruptcy', 'non-bankruptcy',
       'non-bankruptcy', 'bankruptcy', 'bankruptcy', 'non-bankruptcy',
       'bankruptcy', 'bankruptcy', 'non-bankruptcy'], dtype=object)
```

```
In [74]:
plt.rcParams['figure.figsize'] = (5,3)
plt.rcParams['font.size'] = 10
plt.rcParams['figure.dpi'] = 150

plt.title('Predicted Class', fontsize = 10, fontweight = 'bold', color = 'red')
sns.countplot(x = ypred,hue= ypred, saturation=1.2)
plt.legend(labels=['Non-Bankrupt','Bankrupt'])

plt.tight_layout()
```



Once the model is trained we test the performance of the model by using confusion matrix.

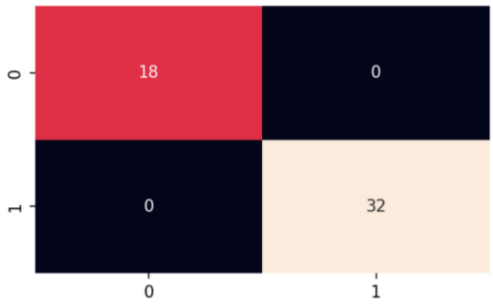
```
In [68]:
#testing performance of model by using confusion matrix
from sklearn.metrics import confusion_matrix

confusion_logist = confusion_matrix(y_test, y_pred)

confusion_logist
```

```
Out[68]: array([[18,  0],
       [ 0, 32]], dtype=int64)
```

```
In [76]: sns.heatmap(confusion_matrix(ytest, ypred),annot= True, cbar= False)
Out[76]: <AxesSubplot:>
```




```
In [75]: confusion_matrix(ytest, ypred)
Out[75]: array([[18,  0],
       [ 0, 32]], dtype=int64)
```

18 represents the number of True Positive and correctly classified into bankruptcy class 0 represents the number of False Positive (no classes are incorrectly classified) 0 represents the number of False Negative (no classes are incorrectly classified) 32 represents the number of True Negative and correctly classified into non-bankruptcy class

Checking the accuracy of the model

#Train accuracy

In [69]:
```
##Accuracy of a Model
# Train Accuracy

train_acc_logist = np.mean(logisticlassifier.predict(X_train)== y_train)
train_acc_logist
```

Out[69]: 0.995

#Test accuracy

In [70]:
```
# Test Accuracy

test_acc_logist = np.mean(logisticlassifier.predict(X_test)== y_test)
test_acc_logist
```

Out[70]: 1.0

In [71]:
```
from sklearn.metrics import accuracy_score

logistic_acc = accuracy_score(y_test, y_pred)
logistic_acc
```

Out[71]: 1.0

The model has predicted all classes correctly.

CONCLUSION : Bankruptcy prediction is the problem of detecting financial distress in businesses which will lead to eventual bankruptcy.