

Prediction of Companies' Bankruptcy

Project report submitted in partial fulfillment of the credits for the course

Applied Machine Learning

End-Semester Project Report

Submitted by:

Ojas Srivastava

Group-29

Department of Electrical Engineering

School of Engineering

Abstract

Bankruptcy prediction has been a subject of interests for almost a century and it still ranks high among hottest topics in economics. The aim of predicting financial distress is to develop a predictive model that combines various econometric measures and allows to foresee a financial condition of a firm.

This project will be aimed at developing a classifier to predict whether a particular company is likely become bankrupt. The dataset has a lot of features (95 features) which will need to be processed, moreover, the data has skewed distribution of the two classes (bankrupt & non-bankrupt) which will demand extra provisions to be included in the classifier.

Table of Contents

<i>List of Figures</i>	<i>6</i>
<i>List of Tables</i>	<i>8</i>
<i>Chapter 1: Introduction</i>	<i>9</i>
The Problem Statement:	9
Dataset:	9
Background Research:	15
1) Bankruptcy:	15
2) Taiwan Bankruptcy Code:	15
Problem Formulation	16
<i>Chapter 2: Literature Survey</i>	<i>17</i>
<i>Chapter 3: Technology and Libraries used.....</i>	<i>18</i>
Python:.....	18
Pandas.....	18
NumPy	18
Matplotlib.....	18
Seaborn.....	19
SciKit Learn	19
Keras	19
<i>Chapter 4: Dataset Exploration & EDA</i>	<i>20</i>
Dataset Exploration:.....	20
1) Check for Missing Values	22
Basic EDA.....	23
1) Occurrence of classes:	23
2) Correlation of features:	24
Advanced EDA.....	25

1) ROA: Return on Assets:	25
2) Total Assets Growth Rate v/s ROA:	27
3) Debt Ratio:	30
4) Cash flow ratios	31
5) Reinvestment into Business	33
6) Tax Rate	35
7) Conclusion.....	35
Chapter-4 Data Preparation	36
Dummy Variables (Categorical Feature Handling)	36
Normalization (Numeric Feature Handling).....	38
Chapter-5 Feature Selection.....	39
Variance Threshold Method.....	39
ANOVA Feature Selection	40
Feature Selection by Domain Knowledge.....	42
Chapter-6 Evaluation Metrics	44
1) Misclassification Error	44
2) Confusion Matrix.....	44
3) Precision & Recall.....	45
4) Precision-Recall Curve	46
Chapter-7 Dealing with Class Imbalance	47
Class Balancing.....	47
SMOTE.....	48
Threshold Moving.....	49
1) Finding Optimal Threshold Point	50
Ensemble Learning.....	51
Chapter-8 Models Used.....	52
1) Logistic Regression	52
2) Support Vector Classifier	52
3) Gaussian Naïve Bayes	52

4) Decision Tree.....	52
5) Random Forest.....	53
6) XGBoost.....	54
7) Neural Network	54
Chapter-9 Predictive Modelling & Results	55
Before Mid-Sem	55
1) Modelling & Hyperparameter Tuning	55
2) Best Model Results:.....	56
After Mid-Sem	57
1) Cost – Sensitive Modelling & Hyperparameter Tuning	57
2) Best models and results (Cost-Sensitive Learning)	59
3) Neural Network Results (Cost-Sensitive Learning)	60
4) Ensemble Learning Results	62
Chapter-10 Conclusion	64
Appendix-A: Important Links	65
References	66

List of Figures

<i>Figure 1 Workflow of this project.....</i>	<i>9</i>
<i>Figure 2 Logo of Python Programming Language.....</i>	<i>18</i>
<i>Figure 3 Logo of Sci-Kit Learn Library</i>	<i>19</i>
<i>Figure 4 Logo of Keras library.....</i>	<i>19</i>
<i>Figure 5 Snapshot of the shape of the dataset.....</i>	<i>20</i>
<i>Figure 6 Code for getting information about columns</i>	<i>20</i>
<i>Figure 7 Column Information.....</i>	<i>22</i>
<i>Figure 8 Code & Result for missing values.....</i>	<i>22</i>
<i>Figure 9 Code snippet for creating bar graph of classes</i>	<i>23</i>
<i>Figure 10 Bar plot of classes.....</i>	<i>23</i>
<i>Figure 11 Code & Result for Class occurrence by number.....</i>	<i>23</i>
<i>Figure 12 Formula of Pearson's Correlation.....</i>	<i>24</i>
<i>Figure 13 Code for generating correlation plot.....</i>	<i>24</i>
<i>Figure 14 Correlation Map of features</i>	<i>25</i>
<i>Figure 15 Formula of Return on Assets.....</i>	<i>26</i>
<i>Figure 16 Correlation map of ROAs</i>	<i>26</i>
<i>Figure 17 Code for generating scatter plots of Total Asset Growth rate v/s ROA(A).....</i>	<i>27</i>
<i>Figure 18 Total Asset Growth Rate v/s ROA(A).....</i>	<i>28</i>
<i>Figure 19 Total Asset Growth Rate v/s ROA(B).....</i>	<i>28</i>
<i>Figure 20 Total Asset Growth Rate v/s ROA(C).....</i>	<i>29</i>
<i>Figure 21 Code snippet for a Box Plot (Debt Ratio%).....</i>	<i>30</i>
<i>Figure 22 Box Plot of Debt Ratio% v/s Class Labels.....</i>	<i>30</i>
<i>Figure 23 Boxplot of Cost of Debt to Classes.....</i>	<i>31</i>
<i>Figure 24 Boxplot of Cash/Total Assets v/s Class.....</i>	<i>32</i>
<i>Figure 25 Boxplot of Cash flow from Operating/ Current Liabilities to classes.....</i>	<i>33</i>
<i>Figure 26 Boxplot of Net Worth/Total Assets to Classes.....</i>	<i>33</i>
<i>Figure 27 Box plot for Cash Reinvestment Ratio to Classes.....</i>	<i>34</i>

<i>Figure 28 Boxplot of R&D Expenses/ Net Sales.....</i>	<i>34</i>
<i>Figure 29 Code Snippet & result of Average Tax Rate by Class.....</i>	<i>35</i>
<i>Figure 30 Code snippet for Data Preparation: Dummy Variable processing</i>	<i>37</i>
<i>Figure 31 Dropping of column</i>	<i>38</i>
<i>Figure 32 Code Snippet for normalizing dataset.....</i>	<i>38</i>
<i>Figure 33 Code Snippet and result of Variance Threshold Feature Selection.....</i>	<i>39</i>
<i>Figure 34 Code snippet for Selecting K best features based on ANOVA</i>	<i>40</i>
<i>Figure 35 Feature Importance of Features</i>	<i>41</i>
<i>Figure 36 Confusion matrix for a Binary Classifier.....</i>	<i>45</i>
<i>Figure 37 Mock PR Curve</i>	<i>46</i>
<i>Figure 38 Working of SMOTE algorithm</i>	<i>48</i>
<i>Figure 39 SMOTE Code Snippet & Result</i>	<i>49</i>
<i>Figure 40 Code snippet for custom function to display classification reports</i>	<i>50</i>
<i>Figure 41 Code snippet for finding Optimal Threshold point from PR curve.....</i>	<i>51</i>
<i>Figure 42 Code snippet for Train-Test Split.....</i>	<i>55</i>
<i>Figure 43 List of models</i>	<i>55</i>
<i>Figure 44 Code Snippet for GridSearchCV and result logging</i>	<i>56</i>
<i>Figure 45 Results of GridSearchCV</i>	<i>56</i>
<i>Figure 46 ROC Curve of Best Model: XGBoost.....</i>	<i>57</i>
<i>Figure 47 Code snippet for models used</i>	<i>58</i>
<i>Figure 48 Architecture of the Neural Network used.....</i>	<i>58</i>
<i>Figure 49 Mock Sigmoid Graph</i>	<i>60</i>
<i>Figure 50 Formula for Binary Cross Entropy Loss.....</i>	<i>61</i>
<i>Figure 51 Activation and stages of the Neural Network.....</i>	<i>61</i>
<i>Figure 52 Compilation configuration of Neural Network.....</i>	<i>61</i>
<i>Figure 53 Configuration used for fitting Neural Network.....</i>	<i>61</i>

List of Tables

<i>Table 1 Dataset Description</i>	<i>10</i>
<i>Table 2 Description of Columns of Dataset.....</i>	<i>15</i>
<i>Table 3 Dummy variable processing with Mock Feature: $x(i)$</i>	<i>36</i>
<i>Table 4 Selected Features based on domain Knowledge</i>	<i>43</i>
<i>Table 5 Results of Best Model: XGBoost.....</i>	<i>57</i>
<i>Table 6 Results for Cost-Sensitive Learning (SVC & LR)</i>	<i>60</i>
<i>Table 7 Results for Cost-Sensitive Learning on Neural Network.....</i>	<i>62</i>
<i>Table 8 Results of Ensemble Learning.....</i>	<i>63</i>

Chapter 1: Introduction

The Problem Statement:

- This project is about detecting whether a company is likely to go bankrupt based on 95 financial indicators of the company.
- The problem will be solved by using Machine Learning Classification Algorithms.
- My work consists of EDA, feature selection methods, best model selection and prediction.
- The complete lifecycle of the project is described in the figure given below and follows the main methods prescribed in the course book for learning with imbalanced datasets:

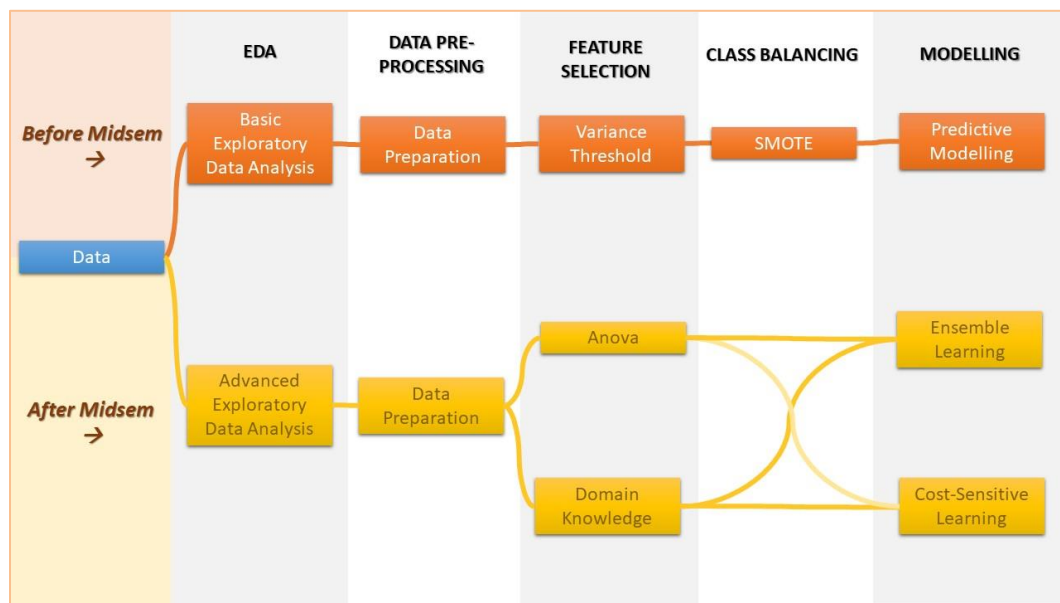


Figure 1 Workflow of this project

Dataset:

The dataset has been taken from the UCI Machine Learning Repository. The data were collected from the Taiwan Economic Journal for the years 1999 to 2009. Company bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange. The dataset is titled "[Taiwanese Bankruptcy Prediction Data Set](#)". The following table summarizes the dataset:

Data Set Characteristics:	Multivariate	Number of Instances:	6819
Attribute Characteristics:	Integer	Number of Attributes:	96
Associated Tasks:	Classification	Missing Values?	N/A

Table 1 Dataset Description

The dataset is provided in csv format and the following table describes its fields [\[link to dataset\]](#) .

Column entry denoted by **Y** is the independent variable while those denoted by **X_i** are features:

Variable	Variable Name & Description	Data Type (Categorical/Numerical)
$s^{(i)}$	Bankrupt or not?	Categorical (Class Label) [0: Non-Bankrupt 1:Bankrupt]
X_1	ROA(C) before interest and depreciation before interest : Return On Total Assets(C)	Numerical
X_2	ROA(A) before interest and % after tax: Return On Total Assets(A)	Numerical
X_3	ROA(B) before interest and depreciation after tax: Return On Total Assets(B)	Numerical
X_4	Operating Gross Margin: Gross Profit/Net Sales	Numerical
X_5	Realized Sales Gross Margin: Realized Gross Profit/Net Sales	Numerical
X_6	Operating Profit Rate: Operating Income/Net Sales	Numerical
X_7	Pre-tax net Interest Rate: Pre-Tax Income/Net Sales	Numerical
X_8	After-tax net Interest Rate: Net Income/Net Sales	Numerical
X_9	Non-industry income and expenditure/revenue:	Numerical

	Net Non-operating Income Ratio	
X ₁₀	Continuous interest rate (after tax) Net Income- Exclude Disposal Gain or Loss/Net Sales	Numerical
X ₁₁	Operating Expense Rate: Operating Expenses/Net Sales	Numerical
X ₁₂	Research and development expense rate: (Research and Development Expenses)/Net Sales	Numerical
X ₁₃	Cash flow rate: Cash Flow from Operating/Current Liabilities	Numerical
X ₁₄	Interest-bearing debt interest rate: Interest- bearing Debt/Equity	Numerical
X ₁₅	Tax rate (A): Effective Tax Rate	Numerical
X ₁₆	Net Value Per Share (B) : Book Value Per Share(B)	Numerical
X ₁₇	Net Value Per Share (A) : Book Value Per Share(A)	Numerical
X ₁₈	Net Value Per Share (C): Book Value Per Share(C)	Numerical
X ₁₉	Persistent EPS in the Last Four Seasons: EPS- Net Income	Numerical
X ₂₀	Cash Flow Per Share	Numerical
X ₂₁	Revenue Per Share (Yuan ¥): Sales Per Share	Numerical
X ₂₂	Operating Profit Per Share (Yuan ¥): Operating Income Per Share	Numerical
X ₂₃	Per Share Net profit before tax (Yuan ¥): Pretax Income Per Share	Numerical
X ₂₄	Realized Sales Gross Profit Growth Rate	Numerical
X ₂₅	Operating Profit Growth Rate: Operating Income Growth	Numerical

X ₂₆	After-tax Net Profit Growth Rate: Net Income Growth	Numerical
X ₂₇	Regular Net Profit Growth Rate: Continuing Operating Income after Tax Growth	Numerical
X ₂₈	Continuous Net Profit Growth Rate: Net Income-Excluding Disposal Gain or Loss Growth	Numerical
X ₂₉	Total Asset Growth Rate	Numerical
X ₃₀	Net Value Growth Rate	Numerical
X ₃₁	Total Asset Return Growth Rate Ratio: Return on Total Asset Growth	Numerical
X ₃₂	Cash Reinvestment %	Numerical
X ₃₃	Current Ratio	Numerical
X ₃₄	Quick Ratio	Numerical
X ₃₅	Interest Expense Ratio: Interest Expenses/Total Revenue	Numerical
X ₃₆	Total debt/Total net worth: Total Liability/Equity Ratio	Numerical
X ₃₇	Debt ratio %: Liability/Total Assets	Numerical
X ₃₈	Net worth/Assets: Equity/Total Assets	Numerical
X ₃₉	Long-term fund suitability ratio (A) : (Long-term Liability+Equity)/Fixed Assets	Numerical
X ₄₀	Borrowing dependency: Cost of Interest-bearing Debt	Numerical
X ₄₁	Contingent liabilities/Net worth: Contingent Liability/Equity	Numerical
X ₄₂	Operating profit/Paid-in capital: Operating Income/Capital	Numerical
X ₄₃	Net profit before tax/Paid-in capital: Pretax Income/Capital	Numerical

X ₄₄	Inventory and accounts receivable/Net value: (Inventory+Accounts Receivables)/Equity	Numerical
X ₄₅	Total Asset Turnover	Numerical
X ₄₆	Accounts Receivable Turnover	Numerical
X ₄₇	Average Collection Days: Days Receivable Outstanding	Numerical
X ₄₈	Inventory Turnover Rate (times)	Numerical
X ₄₉	Fixed Assets Turnover Frequency	Numerical
X ₅₀	Net Worth Turnover Rate (times): Equity Turnover	Numerical
X ₅₁	Revenue per person	Numerical
X ₅₂	Operating profit per person	Numerical
X ₅₃	Allocation rate per person: Fixed Assets Per Employee	Numerical
X ₅₄	Working Capital to Total Assets	Numerical
X ₅₅	Quick Assets/Total Assets	Numerical
X ₅₆	Current Assets/Total Assets	Numerical
X ₅₇	Cash/Total Assets	Numerical
X ₅₈	Quick Assets/Current Liability	Numerical
X ₅₉	Cash/Current Liability	Numerical
X ₆₀	Current Liability to Assets	Numerical
X ₆₁	Operating Funds to Liability	Numerical
X ₆₂	Inventory/Working Capital	Numerical
X ₆₃	Inventory/Current Liability	Numerical
X ₆₄	Current Liabilities/Liability	Numerical
X ₆₅	Working Capital/Equity	Numerical
X ₆₆	Current Liabilities/Equity	Numerical
X ₆₇	Long-term Liability to Current Assets	Numerical
X ₆₈	Retained Earnings to Total Assets	Numerical

X ₆₉	Total income/Total expense	Numerical
X ₇₀	Total expense/Assets	Numerical
X ₇₁	Current Asset Turnover Rate: Current Assets to Sales	Numerical
X ₇₂	Quick Asset Turnover Rate: Quick Assets to Sales	Numerical
X ₇₃	Working capital Turnover Rate: Working Capital to Sales	Numerical
X ₇₄	Cash Turnover Rate	Numerical
X ₇₅	Cash Flow to Sales	Numerical
X ₇₆	Fixed Assets to Assets	Numerical
X ₇₇	Current Liability to Liability	Numerical
X ₇₈	Current Liability to Equity	Numerical
X ₇₉	Equity to Long-term Liability	Numerical
X ₈₀	Cash Flow to Total Assets	Numerical
X ₈₁	Cash Flow to Liability	Numerical
X ₈₂	CFO to Assets	Numerical
X ₈₃	Cash Flow to Equity	Numerical
X ₈₄	Current Liability to Current Assets	Numerical
X ₈₅	Liability-Assets Flag	Categorical [1 if Total Liability exceeds Total Assets, 0 otherwise]
X ₈₆	Net Income to Total Assets	Numerical
X ₈₇	Total assets to GNP price	Numerical
X ₈₈	No-credit Interval	Numerical
X ₈₉	Gross Profit to Sales	Numerical
X ₉₀	Net Income to Stockholder's Equity	Numerical
X ₉₁	Liability to Equity	Numerical

X ₉₂	Degree of Financial Leverage (DFL)	Numerical
X ₉₃	Interest Coverage Ratio (Interest expense to EBIT)	Numerical
X ₉₄	Net Income Flag	Categorical [1 if Net Income is Negative for the last two years, 0 otherwise]
X ₉₅	Equity to Liability	Numerical

Table 2 Description of Columns of Dataset

Background Research:

1) Bankruptcy:

- a) According to popular finance website, Investopedia, Bankruptcy is a legal proceeding involving a business that is unable to repay their outstanding debts with their current business operations. [1]
- b) The bankruptcy process begins with a petition, all the debtor's assets are measured and evaluated, and the assets may be used to repay a portion of outstanding debt.
- c) Bankruptcy offers an individual or business a chance to start fresh by forgiving debts that simply cannot be paid while giving creditors a chance to obtain some measure of repayment based on the business's assets available.

2) Taiwan Bankruptcy Code:

- a) This dataset classifies a company as bankrupt per the Taiwan's Stock Exchange rules.
- b) In this project we are particularly dealing with Corporate Bankruptcy.
- c) Bankruptcy may be initiated by either the debtor or a creditor when the debtor is unable to meet its debts as they are due. A debtor who has ceased to make repayments shall be presumed to be unable to pay its debts. If the bankruptcy application is made by the debtor, the debtor must file with the court a report on the status of its assets and a list of creditors. If the application is made by a creditor, it must file a description of the debt owed. Upon receiving the bankruptcy application, the court will begin its

investigation and seek the opinion of the debtors, creditors and other interested parties. Although the court is required to accept or reject the bankruptcy application within seven days of receiving it, it can take longer, and the seven-day period is not a statutory deadline. When an application for bankruptcy is received, the court may, ex officio or upon application of the creditors, arrest or detain the debtor or give an order for precautionary measures, such as to freeze the debtor's property from being disposed of, transferred or sold, before the bankruptcy is adjudicate. [2]

Problem Formulation

This problem is a classification problem. The dataset will be split in to 2 sets *training & testing sets*. Training set $\{\mathbf{x}, y\}$ are input-output data, \mathbf{x} is an input vector with 95 features $x_j; j=1,2,3,\dots,95$, as its components and output y is a discrete class $y_q; q=1,2$. The goal is to find to predict the output values for new inputs (from testing set) and deciding which of the classes (y_1, y_2) each new vector \mathbf{x} belongs, based on training from examples of each class.

Chapter 2: Literature Survey

1. Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study by Liang, Deron, Lu, Chia-Chi, Tsai, Chih-Fong, Shih, Guan-An [3]:

- a. Effective bankruptcy prediction is critical for financial institutions to make appropriate lending decisions. In general, the input variables (or features), such as financial ratios, and prediction techniques, such as statistical and machine learning techniques, are the two most important factors affecting the prediction performance. While many related works have proposed novel prediction techniques, very few have analyzed the discriminatory power of the features related to bankruptcy prediction. In the literature, in addition to financial ratios (FRs), corporate governance indicators (CGIs) have been found to be another important type of input variable. However, the prediction performance obtained by combining CGIs and FRs has not been fully examined. Only some selected CGIs and FRs have been used in related studies and the chosen features may differ from study to study. Therefore, the aim of this paper is to assess the prediction performance obtained by combining seven different categories of FRs and five different categories of CGIs. The experimental results, based on a real-world dataset from Taiwan, show that the FR categories of solvency and profitability and the CGI categories of board structure and ownership structure are the most important features in bankruptcy prediction. Specifically, the best prediction model performance is obtained with a combination in terms of prediction accuracy, Type I/II errors, ROC curve, and misclassification cost. However, these findings may not be applicable in some markets where the definition of distressed companies is unclear and the characteristics of corporate governance indicators are not obvious, such as in the Chinese market.

2. Applied Machine Learning by Madan Gopal [4]

- a. The prescribed book for the course EED363: Applied Machine Learning.

Chapter 3: Technology and Libraries used

Python:



Figure 2 Logo of Python Programming Language

In this project I have used Python [5] programming language. Python is an interpreted high-level general-purpose programming language. It has a strong community of active developers who have made it the to-go language for data analytics needs. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming.

Pandas

Pandas [6] is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. It offers data structures and operations for manipulating numerical tables and time series.

NumPy

NumPy [7] is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Matplotlib

Matplotlib [8] is a comprehensive library for creating static, animated, and interactive visualizations in Python. It provides an object-oriented API for embedding plots into applications using general-purpose GUI.

Seaborn

Seaborn [9] is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Its plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

SciKit Learn



Figure 3 Logo of Sci-Kit Learn Library

There are many libraries built on Python offering machine learning capabilities. I have chosen scikit learn as it has a strong community and extensive documentation on its website [10] . This library provided me efficient tools for modelling.

Keras



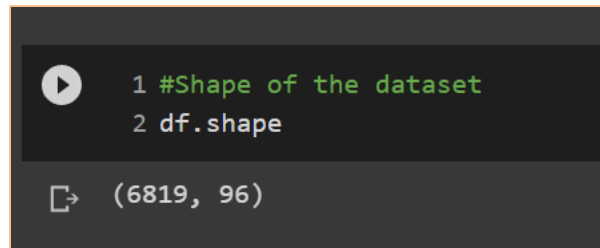
Figure 4 Logo of Keras library

Keras [11] is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow [12] library.

Chapter 4: Dataset Exploration & EDA

Dataset Exploration:

- The dataset has total of 95 features and one column for the class.
- There are 6819 entries in the dataset out of which just 220 are from the Bankrupt Class which accounts for just 3.2% of the dataset.
- All the cells in the data frame are float points.
- There is no missing, NaN data in the whole dataset.
- Following snapshots contain the code snippets and the results:

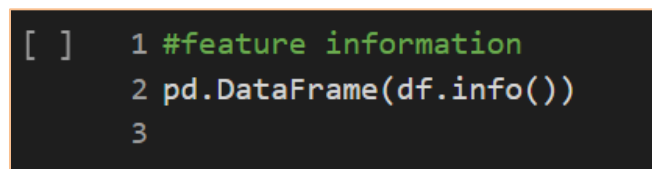


```
1 #Shape of the dataset
2 df.shape
```

(6819, 96)

Figure 5 Snapshot of the shape of the dataset

- This (figure 4) shows that the data has 1 class label field & 95 features along with 6819 entries.



```
[ ] 1 #feature information
    2 pd.DataFrame(df.info())
    3
```

Figure 6 Code for getting information about columns

- This (figure 5) contains the code for checking the information about the columns. Method `info` is an in-built pandas method. The following figure will show the output:

Data columns (total 96 columns):			
#	Column	Non-Null Count	Dtype
0	Bankrupt?	6819 non-null	int64
1	ROA(C) before interest and depreciation before interest	6819 non-null	float64
2	ROA(A) before interest and % after tax	6819 non-null	float64
3	ROA(B) before interest and depreciation after tax	6819 non-null	float64
4	Operating Gross Margin	6819 non-null	float64
5	Realized Sales Gross Margin	6819 non-null	float64
6	Operating Profit Rate	6819 non-null	float64
7	Pre-tax net Interest Rate	6819 non-null	float64
8	After-tax net Interest Rate	6819 non-null	float64
9	Non-industry income and expenditure/revenue	6819 non-null	float64
10	Continuous interest rate (after tax)	6819 non-null	float64
11	Operating Expense Rate	6819 non-null	float64
12	Research and development expense rate	6819 non-null	float64
13	Cash flow rate	6819 non-null	float64
14	Interest-bearing debt interest rate	6819 non-null	float64
15	Tax rate (A)	6819 non-null	float64
16	Net Value Per Share (B)	6819 non-null	float64
17	Net Value Per Share (A)	6819 non-null	float64
18	Net Value Per Share (C)	6819 non-null	float64
19	Persistent EPS in the Last Four Seasons	6819 non-null	float64
20	Cash Flow Per Share	6819 non-null	float64
21	Revenue Per Share (Yuan ¥)	6819 non-null	float64
22	Operating Profit Per Share (Yuan ¥)	6819 non-null	float64
23	Per Share Net profit before tax (Yuan ¥)	6819 non-null	float64
24	Realized Sales Gross Profit Growth Rate	6819 non-null	float64
25	Operating Profit Growth Rate	6819 non-null	float64
26	After-tax Net Profit Growth Rate	6819 non-null	float64
27	Regular Net Profit Growth Rate	6819 non-null	float64
28	Continuous Net Profit Growth Rate	6819 non-null	float64
29	Total Asset Growth Rate	6819 non-null	float64
30	Net Value Growth Rate	6819 non-null	float64
31	Total Asset Return Growth Rate Ratio	6819 non-null	float64
32	Cash Reinvestment %	6819 non-null	float64
33	Current Ratio	6819 non-null	float64
34	Quick Ratio	6819 non-null	float64
35	Interest Expense Ratio	6819 non-null	float64
36	Total debt/Total net worth	6819 non-null	float64
37	Debt ratio %	6819 non-null	float64
38	Net worth/Assets	6819 non-null	float64
39	Long-term fund suitability ratio (A)	6819 non-null	float64
40	Borrowing dependency	6819 non-null	float64
41	Contingent liabilities/Net worth	6819 non-null	float64
42	Operating profit/Paid-in capital	6819 non-null	float64
43	Net profit before tax/Paid-in capital	6819 non-null	float64
44	Inventory and accounts receivable/Net value	6819 non-null	float64
45	Total Asset Turnover	6819 non-null	float64
46	Accounts Receivable Turnover	6819 non-null	float64
47	Average Collection Days	6819 non-null	float64
48	Inventory Turnover Rate (times)	6819 non-null	float64
49	Fixed Assets Turnover Frequency	6819 non-null	float64
50	Net Worth Turnover Rate (times)	6819 non-null	float64
51	Revenue per person	6819 non-null	float64
52	Operating profit per person	6819 non-null	float64
53	Allocation rate per person	6819 non-null	float64
54	Working Capital to Total Assets	6819 non-null	float64
55	Quick Assets/Total Assets	6819 non-null	float64
56	Current Assets/Total Assets	6819 non-null	float64
57	Cash/Total Assets	6819 non-null	float64
58	Quick Assets/Current Liability	6819 non-null	float64
59	Cash/Current Liability	6819 non-null	float64
60	Current Liability to Assets	6819 non-null	float64
61	Operating Funds to Liability	6819 non-null	float64
62	Inventory/Working Capital	6819 non-null	float64
63	Inventory/Current Liability	6819 non-null	float64
64	Current Liabilities/Liability	6819 non-null	float64
65	Working Capital/Equity	6819 non-null	float64
66	Current Liabilities/Equity	6819 non-null	float64
67	Long-term Liability to Current Assets	6819 non-null	float64
68	Retained Earnings to Total Assets	6819 non-null	float64

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

Figure 7 Column Information

- This (figure 6) shows that there are no missing values and that there are only numerical fields.

1) Check for Missing Values

I ran a test on the whole dataset to see any missing values. This is to be doubly sure. No missing values were found.

```

1 #Checking for missing values
2 c=0
3 for col in df.columns:
4     null_values_count = df[col].isnull().sum()
5     if null_values_count > 0:
6         c=c+1
7         print(col,null_values_count)
8
9 if c==0:
10     print("No missing values found in the dataset")

```

No missing values found in the dataset

Figure 8 Code & Result for missing values

Basic EDA

1) Occurrence of classes:

I checked the frequency of occurrence of both the classes and observed a big inequality between the occurrences. Following figures show the code snippet and the bar graph.

```
1 df["Bankrupt?"].value_counts().plot(kind="bar")
2 plt.title("Bar plot of Companies")
3 plt.xticks([0,1],["Non-Bankrupt(0)","Bankrupt(1)"],rotation = 45)
4 plt.ylabel("No. of Comapnies")
```

Figure 9 Code snippet for creating bar graph of classes

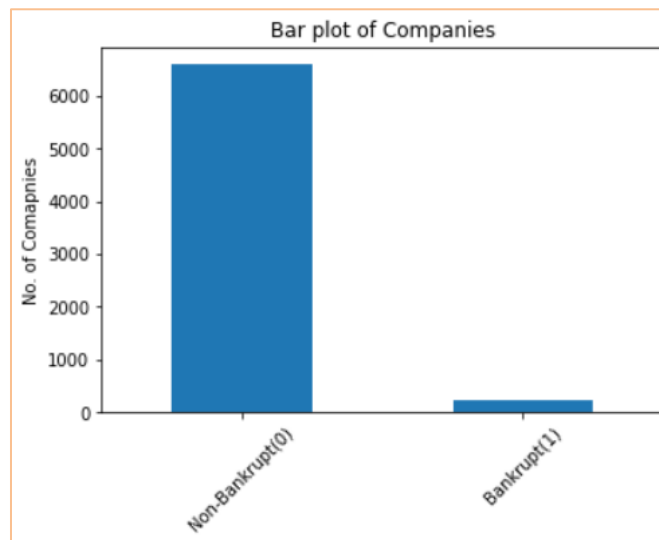


Figure 10 Bar plot of classes

```
[ ] 1 df.groupby('Bankrupt?')['Bankrupt?'].count()

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

Figure 11 Code & Result for Class occurrence by number

- These figures show that a huge skew is there in class distribution. Special care will be taken

to remove bias towards the majority class in my training.

2) Correlation of features:

- Pearson correlation was used to see the linear correlation between features.
- In statistics, the Pearson correlation coefficient, or the bivariate correlation is a measure of linear correlation between two sets of data. It is the covariance of two variables, divided by the product of their standard deviations; thus, it is essentially a normalized measurement of the covariance, such that the result always has a value between -1 and 1 .
- Formula for Pearson's correlation is:

$$\rho_{X,Y} = \frac{\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]}{\sqrt{\mathbb{E}[X^2] - (\mathbb{E}[X])^2} \sqrt{\mathbb{E}[Y^2] - (\mathbb{E}[Y])^2}}.$$

Figure 12 Formula of Pearson's Correlation

```
1 #Checking for multicollinearity
2 df_v1_corr = df_v1.corr()
3 df_v1_corr.style.background_gradient(cmap=sns.diverging_palette(600, 10, as_cmap=True))
```

Figure 13 Code for generating correlation plot

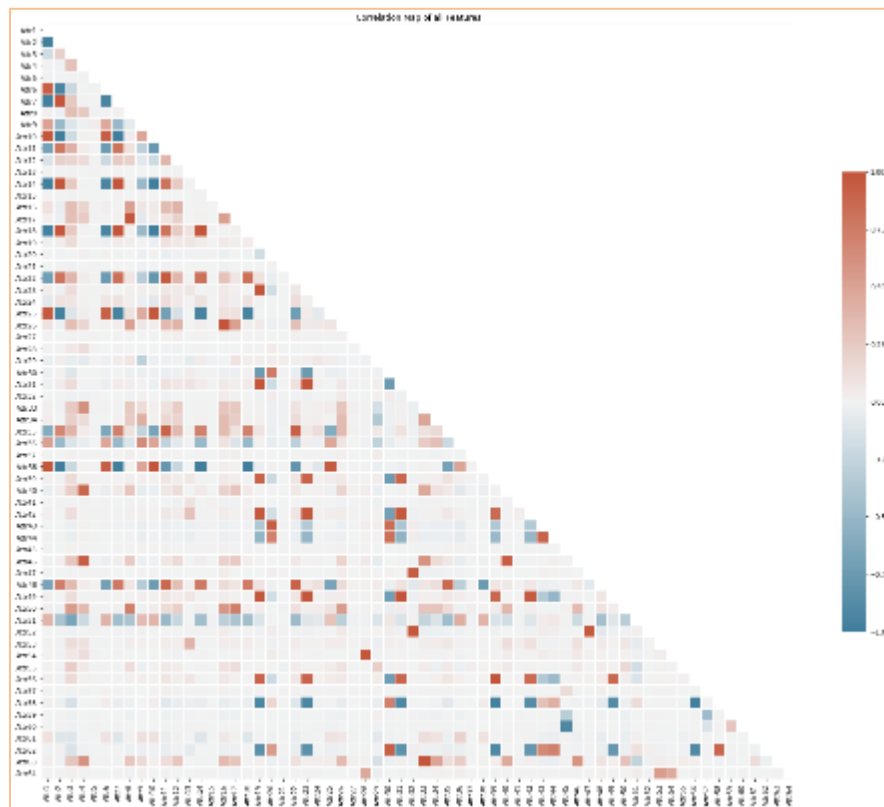


Figure 14 Correlation Map of features

Advanced EDA

I used an EDA library called as AutoViz [13] to get quick insights into the data. Complete report generated by me from the tool can be found here: [Comprehensive EDA Report](#)

A few interesting trends were found and are discussed below:

1) ROA: Return on Assets:

Return on assets (ROA) [14] is an indicator of how profitable a company is relative to its total assets. ROA gives an idea as to how efficient a company's management is at using its assets to generate earnings. ROA is displayed as a percentage, the higher the ROA the better. It is imperative to maintain a healthy ROA to prevent the company from going bankrupt.

$$\text{Return on Assets} = \frac{\text{Net Income}}{\text{Total Assets}}$$

Figure 15 Formula of Return on Assets

In this dataset we have three ROA metrics:

- ROA (A) : ROA before interest and % after tax: Return On Total Assets
- ROA (B) : ROA before interest and depreciation after tax: Return On Total Assets
- ROA (C) : ROA before interest and depreciation before interest: Return On Total Assets.

I plotted a correlation map of the Return on Assets and saw a revealing fact:



Figure 16 Correlation map of ROAs

- This (figure 15) suggests that Return on Assets before interest does not change much (given high levels of correlation between them) even if it is accounted for taxation and depreciation. This means almost every company will enjoy similar levels of ROA after depreciation and taxation as both depreciation and interest are tax deductible.
- It further implies that ROA for companies will start diverging from each other once interest payments are considered, thereby, giving an indication of bankruptcy. This shows that irresponsible credit borrowing can affect a company's financial health.

2) Total Assets Growth Rate v/s ROA:

Total Asset Growth rate is an indicator which tells us about the growth rate of the assets. A company generally takes loans to generate new kinds of assets for itself, and uses the revenue generated in turn from the use of the assets to repay the debt. If a company has low levels of Asset growth, then it would become difficult to repay the rising amount of debt on its books. I have plotted these three graphs of Total Asset Growth Rate v/s ROA (A) , ROA (B) , ROA (C) .

```
1 fig, ax = plt.subplots(figsize=(20, 12))
2 # plot heatmap
3 sns.scatterplot(data=df, y=" Total Asset Growth Rate", x=" ROA(A) before interest and % after tax", hue="Bankrupt?", style="Bankrupt?",
4                 size="Bankrupt?", sizes=(150,5))
5 plt.axvline(0.55, color='r')
6 plt.title ("Total Asset Growth Rate v/s ROA(A)")
7 plt.show()
8
```

Figure 17 Code for generating scatter plots of Total Asset Growth rate v/s ROA(A)

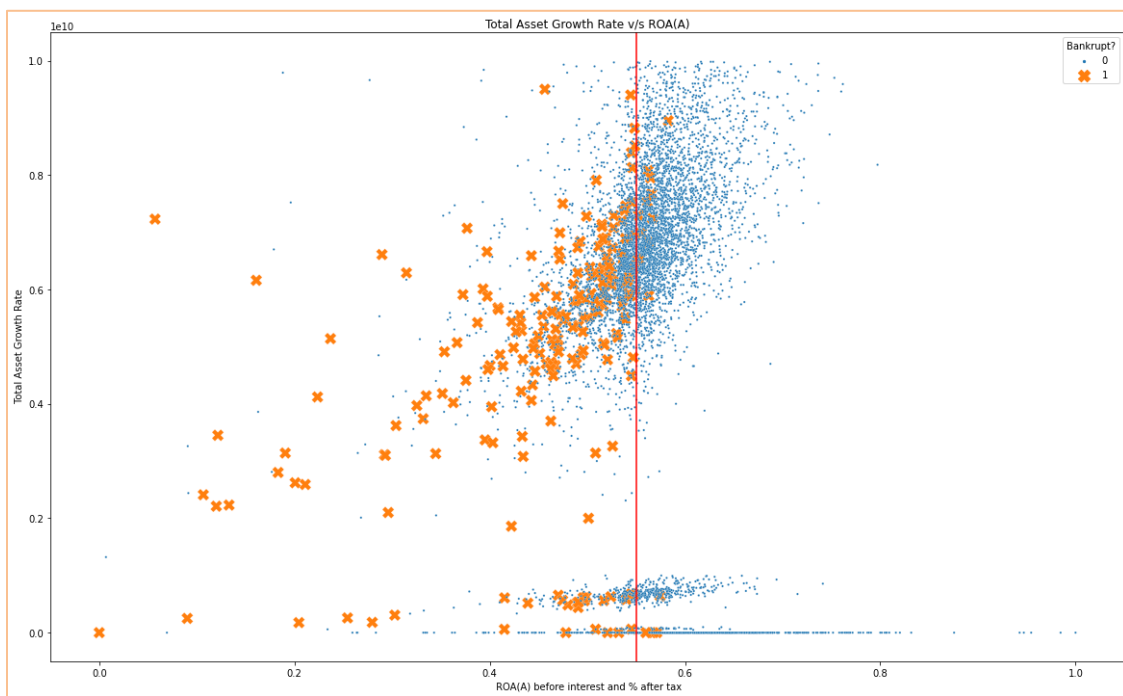


Figure 18 Total Asset Growth Rate v/s ROA(A)

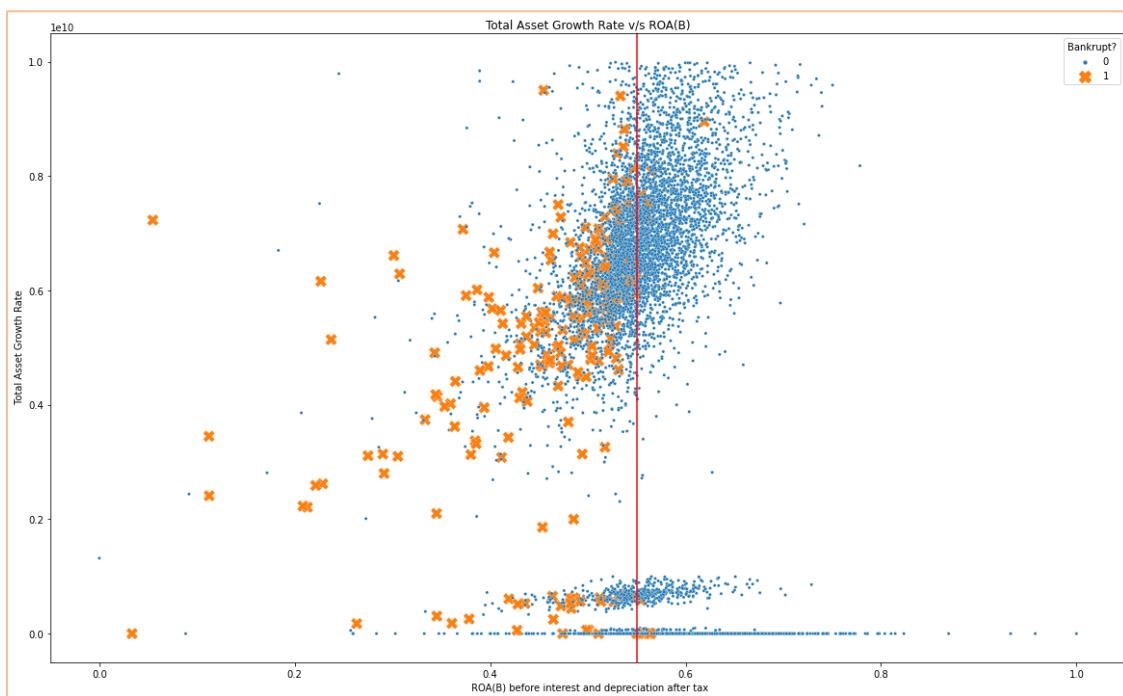


Figure 19 Total Asset Growth Rate v/s ROA(B)

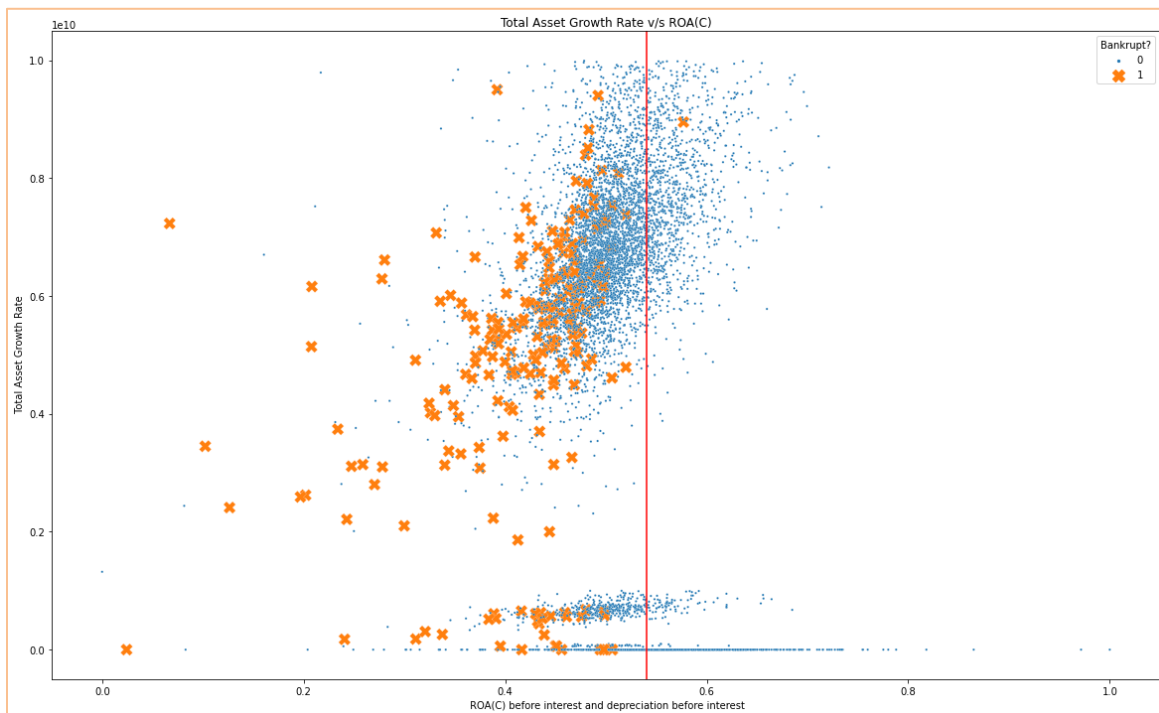


Figure 20 Total Asset Growth Rate v/s ROA(C)

- In figures 17,18 & 19 we see a similar trend, the Bankrupt (class:1) companies have ROA generally lower than ~ 0.5 . The red line on each graph divides the graph into 2 sections: one where occurrence of Bankrupt companies are generally observed while in the other section generally non-bankrupt companies are observed.
- The red separating line in each figure corresponds to:
 - *Figure 17* $\rightarrow x=0.55$
 - *Figure 18* $\rightarrow x=0.55$
 - *Figure 19* $\rightarrow x=0.54$
- It can be said that Bankrupt companies generally have ROA less than 0.55
- Further it can be seen that Bankrupt companies generally have low Total Asset Growth Rate and low ROA, making a deadly mix of bad financial health riddled with unhealthy debt.

3) Debt Ratio:

In this section we will evaluate Debt Ratio of companies. Formula of debt ratio is given below:

$$\text{Debt Ratio \%} = \frac{\text{Liabilities}}{\text{Total Assets}}$$

X37-Debt ratio % column in the dataset could indicate amount of debt a company holds compared to its assets. Higher level of debts means the company needs to pay higher to repay its debt. One would expect both Bankrupt and non-Bankrupt companies pay the same rate of interest on their loans, i.e., at the country's average corporate lending rate. However, an interesting trend is observed.

```
1 fig, ax = plt.subplots(figsize=(10,7))
2 # plot heatmap
3 sns.boxplot(data=df, x="Bankrupt?", y=" Debt ratio %",
4             palette="Set3")
5 ax.set_xticklabels(["Non-Bankrupt", "Bankrupt"])
6 plt.title("Boxplot of Debt Ratio%")
7 plt.show()
8
```

Figure 21 Code snippet for a Box Plot (Debt Ratio%)

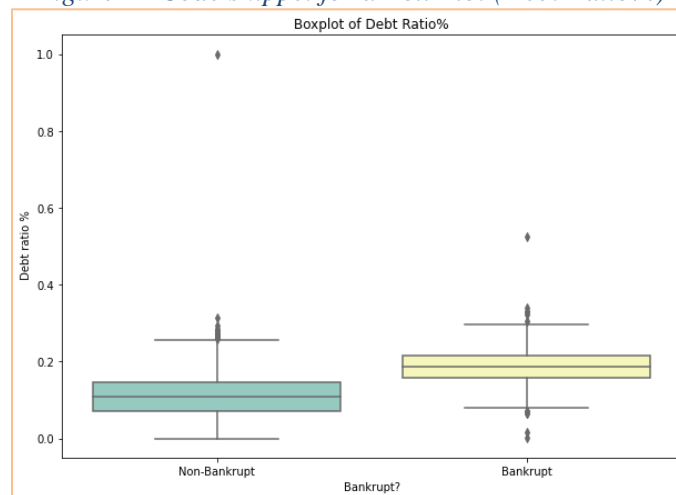


Figure 22 Box Plot of Debt Ratio% v/s Class Labels

- A very stark difference is observed between Debt Ratio % of the 2 classes of the

companies. It is surprising to observe that the 25th quartile of Debt Ratio% for Bankrupt companies is greater than the 75th quartile mark for Non-Bankrupt companies.

- Also, one can see the huge difference between the average Debt Ratio% of both the classes of companies. On one hand where Non-bankrupt companies maintain Debt Ratio% at 0.1%, the Bankrupt Companies maintain Debt Ratio% at 0.2% on an average which is higher than even the highest quartile for non-Bankrupt companies.
- It can also be inferred that Bankrupt companies take on debt which is not in sync with their current assets' ability to generate revenues that can help the company to meet its debt obligations.
- This claim is also supported by another graph which shows the X40 - Borrowing dependency: Cost of Interest-bearing Debt of the company.

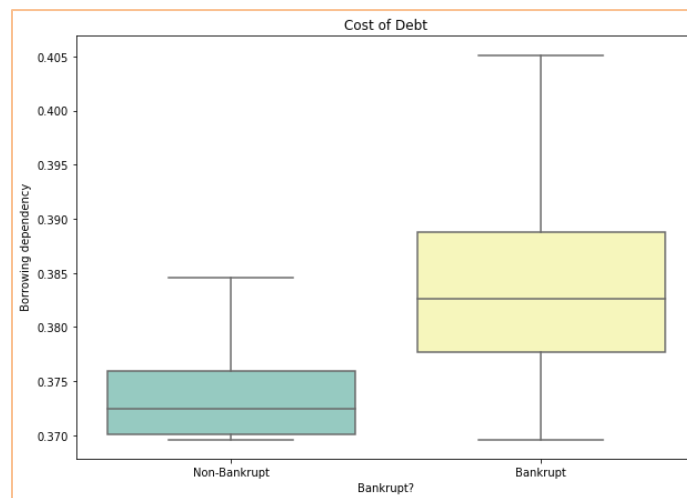


Figure 23 Boxplot of Cost of Debt to Classes

4) Cash flow ratios

Cash flows [15] are the net amount of cash and cash-equivalents being transferred into and out of a business. Cash received are inflows, and money spent are outflows.

At a fundamental level, a company's ability to create value for shareholders is determined by its ability to generate positive cash flows, or more specifically, maximize long-term free cash flow (FCF). Free cash flow is the cash that a company generates from its normal business operations after subtracting any money spent on capital expenditures.

Since, cashflow is such an important component for a company's survival, I plotted boxplots for X57 - Cash/Total Assets to see the trend. The following graph shows the trend:

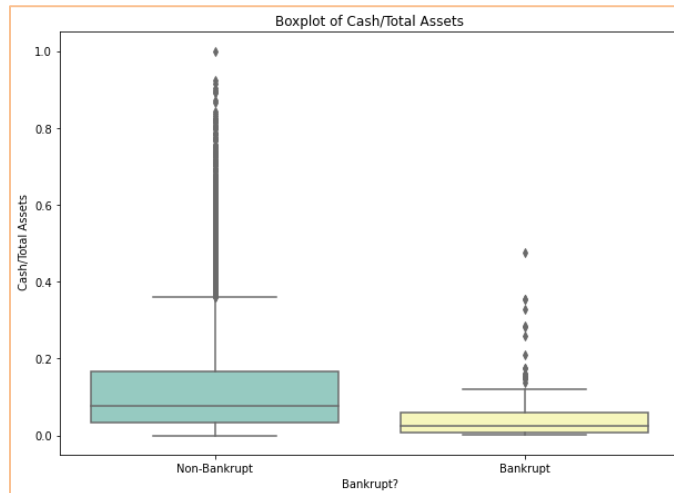


Figure 24 Boxplot of Cash/Total Assets v/s Class

- It can be observed that the 75th quartile of Cash/Total Assets for Bankrupt companies is lower than the average of the same for Non-Bankrupt companies.
- Such a trend indicates that Bankrupt companies can't generate good amount of cash for each unit of their asset.
- One can also see the X13 - Cash flow rate: Cash Flow from Operating/Current Liabilities boxplot to see that relatively Bankrupt companies are generating low cash from operations to their liabilities:

PTO →

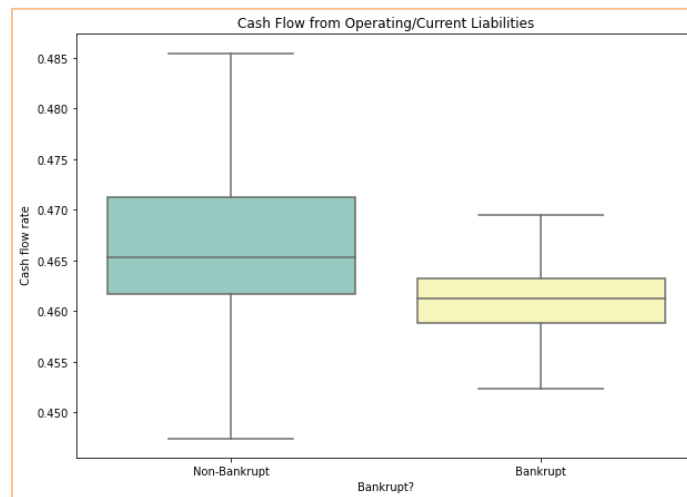


Figure 25 Boxplot of Cash flow from Operating/ Current Liabilities to classes

- A reduced cash flow is indicative of the fact that the net income of the company is less when compared to its expenses as the CFO are started with the top line in its calculation as Net Income. The following graph is indicative of this:

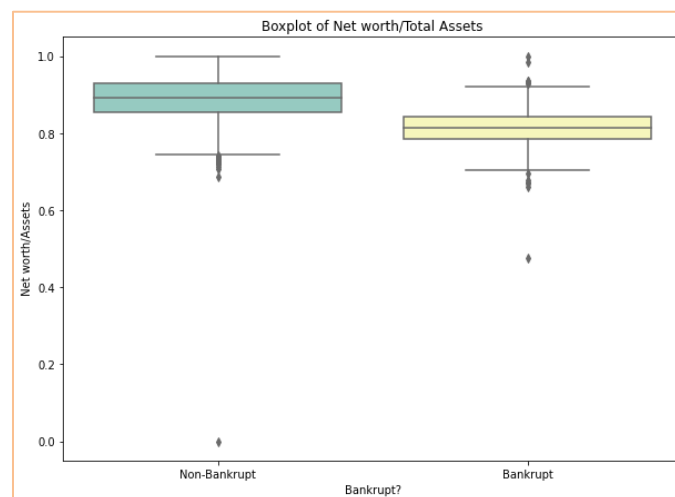


Figure 26 Boxplot of Net Worth/Total Assets to Classes

5) Reinvestment into Business

Reinvesting into a Business can help bring resources to innovate, restructure or create new processes and products, thereby, improving the prospects of generating profits.

Unfortunately, we do not see such a trend in Bankrupt companies. The following 2 graphs show that Bankrupt companies re-invest relatively less to their non-Bankrupt counterparts. One of the reasons could be high payments of interest thereby restraining management to divert funds. Other, could be general disinterest in reinvestment by the management. However, no causal relationship could be established by me till now to explain low levels of re-investment. It will need further investigation, which is out of the scope of this report.

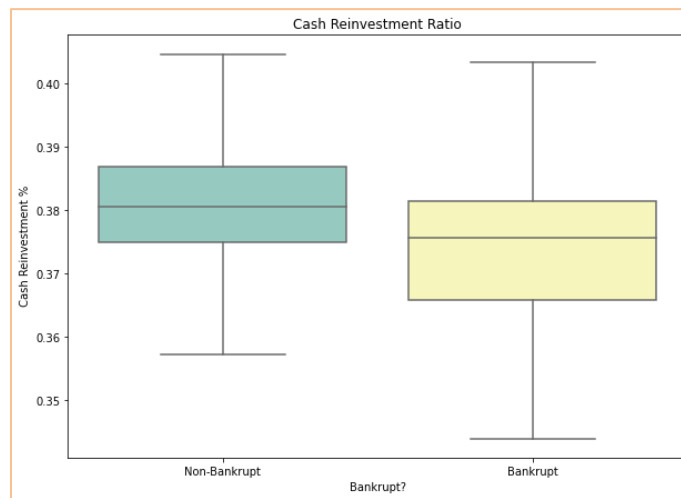


Figure 27 Box plot for Cash Reinvestment Ratio to Classes

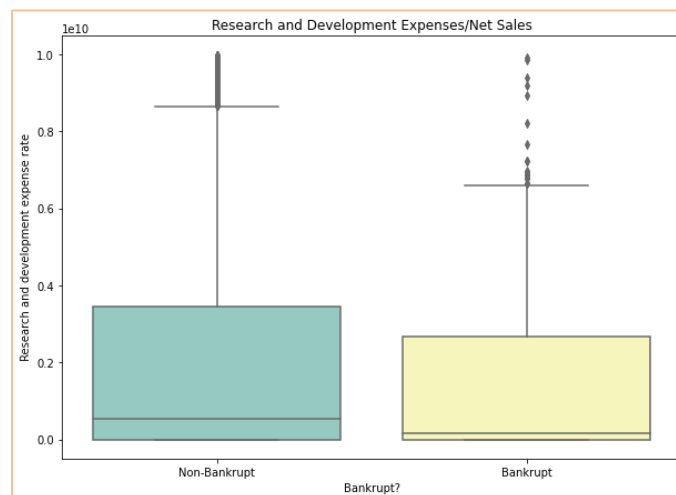


Figure 28 Boxplot of R&D Expenses/ Net Sales

6) Tax Rate

It is observed that Bankrupt companies pay lesser tax on an average than Non- Bankrupt companies. This is because of reduction in taxable income as these companies can deduct the interest on their debt, lower their tax liability, and make themselves more profitable than those that rely solely on equity.

No causal relationship could be established either from external sources like regulations or existing dataset.

```
1 bkr= df["Bankrupt?"]==1
2 mean_for_tax_bkr = df.loc[bkr, ' Tax rate (A)'].mean()
3
4 not_bkr= df["Bankrupt?"]==0
5 mean_for_tax_not_bkr = df.loc[not_bkr, ' Tax rate (A)'].mean()
6
7 print("Mean Tax for Bankrupt Companies : ", round(mean_for_tax_bkr*100,2),
8       "%\nMean Tax for Non- Bankrupt Companies: ",round(mean_for_tax_not_bkr*100,2),"%")

Mean Tax for Bankrupt Companies :  3.17 %
Mean Tax for Non- Bankrupt Companies:  11.78 %
```

Figure 29 Code Snippet & result of Average Tax Rate by Class

7) Conclusion

In conclusion, it's been observed that Bankrupt companies tend to take on unreasonable amount of debt compared to their assets. They then do not develop their assets, which could potentially repay the interests on their debt. This leads to the firm diverting large amounts of their cash flows to interest payments, thereby, undermining funding for important R&D and innovation required for the business to compete in the market. This leads to a vicious cycle of indebtedness and poor financial health.

Chapter-4 Data Preparation

A dataset needs to be prepared well before predictive modelling. In this chapter I will discuss the steps taken to prepare my dataset according to the best practices as illustrated in “*Applied Machine Learning by M.Gopal*” [4]

Dummy Variables (Categorical Feature Handling)

Categorical data attributes (in this case *nominal data*) contain label values rather than numeric values. Some of the Machine Learning Algorithms can work with categorical data (e.g. decision trees) while others require the user to enter numeric values to the algorithms (e.g. Logistic regression, SVC). This requires the categorical data to be converted into a numerical form.

There are many types of encoding which can be done , however, for this project I will use the method suggested in the course book [4] : *dummy variables method* . Other methods like *Integer Encoding* work well with ordinal data and cannot be used in this case. The *dummy variables method* will be applied to the categorical features: X85: Liability-Assets Flag; X94: Net Income Flag. In the following, mock data-table I describe the process with a mock feature $x_{(i)}$ which has two data categories: *yes* & *no*.

$x_{(i)}$	After dummy variable processing →	$x_{(i)}d_1$	$x_{(i)}d_2$
yes		1	0
no		0	1
yes		1	0
yes		1	0
no		0	1

Table 3 Dummy variable processing with Mock Feature: $x_{(i)}$

Pandas library [6] `get_dummies()` function was used for creating dummy columns. The following code snippet shows how it was done in this project:

```

Dummy Variables for categorical features

[15] 1 X[" Net Income Flag"].nunique()

1

[19] 1 X[" Net Income Flag"].value_counts()

1    6819
Name: Net Income Flag, dtype: int64

[17] 1 X[" Liability-Assets Flag"].value_counts()

0    6811
1      8
Name: Liability-Assets Flag, dtype: int64

[9] 1 X_dummy = pd.get_dummies(data=X,prefix_sep='_',columns=[" Liability-Assets Flag"," Net Income Flag"])
2
3 X_dummy.head()

```

Figure 30 Code snippet for Data Preparation: Dummy Variable processing

- It is observed that (Figure 29: Cell 15 & 19) that the feature: X94: Net Income Flag has only one kind of categorical entry (i.e., category: 1) as it returns 1 to the *nunique()* function. This is verified by the *pandas.value_counts()* function which returns a list of frequency of occurrence of all unique entries in a particular column. It returns 6819 which is exactly the number of entries in the data frame.
- After executing *pd.get_dummies()* function, following changes were made to the data frame:
 - X85: Liability-Assets Flag -> Liability-Assets Flag_0
Liability-Assets Flag_1
 - X94: Net Income Flag -> Net Income Flag_1
- Notice only one dummy column is made for X94 this is because of the reasons mentioned in the preceding points.
- Since, all entries of X94 are same, it will not contribute to classification and should be removed to reduce the complexity of our model. Following code snippet shows the dropping of the column:

```
1 X = X.drop(labels=" Net Income Flag_1" ,axis=1, inplace=False)
```

Figure 31 Dropping of column

Normalization (Numeric Feature Handling)

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information.

Normalization prevents some algorithms from getting biased towards features with higher numerical values by bringing each feature to the same scale. All numeric features are scaled and brought to a common scale (a float between 0 &1).

In Python, I used *MinMaxScaler()* function from the Scikit Learn [10] pre-processing library.

```
▼ Normalization

[25] 1 from sklearn import preprocessing
      2 min_max_scaler = preprocessing.MinMaxScaler()
      3
      4 X = min_max_scaler.fit_transform(X)
      5 pd.X = pd.DataFrame(X)

[30] 1 Y = min_max_scaler.fit_transform(Y)
      2 pd.Y = pd.DataFrame(Y)
```

Figure 32 Code Snippet for normalizing dataset

Chapter-5 Feature Selection

Feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model. Statistical-based feature selection methods involve evaluating the relationship between each input variable and the target variable using statistics and selecting those input variables that have the strongest relationship with the target variable. These methods can be fast and effective, although the choice of statistical measures depends on the data type of both the input and output variables. [16]. The key is to choose features that [4] are computationally feasible ; lead to ‘good’ machine-learning success; and reduce the problem data into a manageable amount of information without discarding valuable (or vital) information.

Variance Threshold Method

This method removes features with variation below a certain cutoff. The idea is when a feature doesn’t vary much within itself, it generally has very little predictive power. Variance Threshold doesn’t consider the relationship of features with the target variable. [16] [17]

Generally, variance cutoff is chosen between 0 to 5. In this project I have chosen threshold as 3.0. I first found out the variance of all the features and then applied thresholding which resulted in selection of 24 features.

```
from sklearn.feature_selection import VarianceThreshold

[ ] 1 var_thres = VarianceThreshold(3.0)
    2 var_thres.fit(X)

VarianceThreshold(threshold=3.0)

[ ] 1 required_features = [col for col in X.columns if col in X.columns[var_thres.get_support()]]
    2 print(required_features)

[' Operating Expense Rate', ' Research and development expense rate', ' Interest-bearing debt interest
```

Figure 33 Code Snippet and result of Variance Threshold Feature Selection

List of selected features is : [' Operating Expense Rate', ' Research and development expense rate', ' Interest-bearing debt interest rate', ' Revenue Per Share

```
(Yuan ¥)', ' Total Asset Growth Rate', ' Net Value Growth Rate', ' Current  
Ratio', ' Quick Ratio', ' Total debt/Total net worth', ' Accounts Receivable  
Turnover', ' Average Collection Days', ' Inventory Turnover Rate (times)',  
' Fixed Assets Turnover Frequency', ' Revenue per person', ' Allocation rate  
per person', ' Quick Assets/Current Liability', ' Cash/Current Liability',  
' Inventory/Current Liability', ' Long-term Liability to Current Assets',  
' Current Asset Turnover Rate', ' Quick Asset Turnover Rate', ' Cash Turnover  
Rate', ' Fixed Assets to Assets', ' Total assets to GNP price']
```

ANOVA Feature Selection

ANOVA [18] stands for “Analysis of variance” and is a parametric statistical hypothesis test to check if the means of two or more groups that are significantly different from each other.

An F-statistic, or F-test, is a class of statistical tests that calculate the ratio between variances values, such as the variance from two different samples or the explained and unexplained variance by a statistical test, like ANOVA. The ANOVA method is a type of F-statistic referred to here as an ANOVA f-test.

ANOVA is used when one variable is numeric and one is categorical, such as numerical input variables and a classification target variable, which is the task at hand.

The results of this test can be used for feature selection where those features that are independent of the target variable can be removed from the dataset.

I have used the following code to do ANOVA feature selection:

```
[121] 1 from sklearn.feature_selection import f_classif  
      2 from sklearn.feature_selection import SelectKBest  
  
[122] 1 fs = SelectKBest(score_func=f_classif, k=20)  
      2 # learn relationship from training data  
      3 fs.fit(X, Y)  
      4 # transform train input data  
      5 X_fs = fs.transform(X)
```

Figure 34 Code snippet for Selecting K best features based on ANOVA

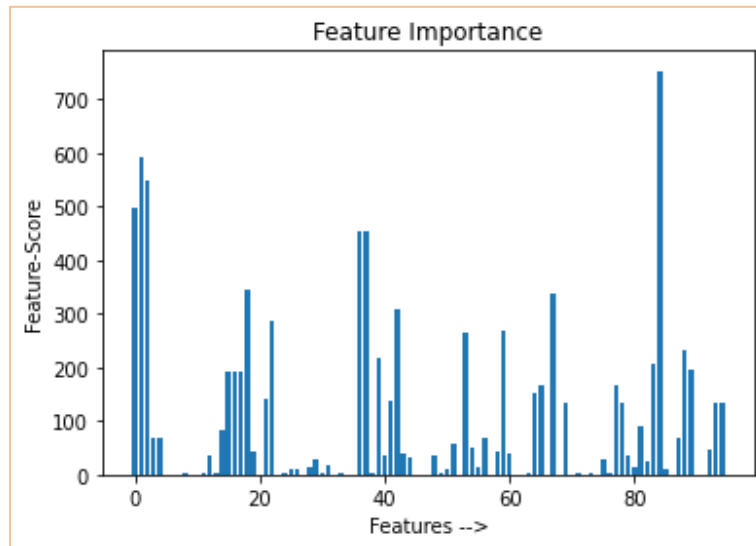


Figure 35 Feature Importance of Features

I selected top 20 most important features, the selected features are given below:

1. *X85: Net Income to Total Assets*
2. *X2: ROA(A) before interest and % after tax*
3. *X3: ROA(B) before interest and depreciation after tax*
4. *X1: ROA(C) before interest and depreciation before interest*
5. *X37: Debt ratio %*
6. *X38: Net worth/Assets*
7. *X19: Persistent EPS in the Last Four Seasons*
8. *X68: Retained Earnings to Total Assets*
9. *X43: Net profit before tax/Paid-in capital*
10. *X23: Per Share Net profit before tax (Yuan ¥)*
11. *X60: Current Liability to Assets*
12. *X54 Working Capital to Total Assets*
13. *X89: Net Income to Stockholder's Equity*
14. *X40: Borrowing dependency*
15. *X84: Current Liability to Current Assets*
16. *X90: Liability to Equity*
17. *X17: Net Value Per Share (A)*
18. *X16: Net Value Per Share (B)*
19. *X18: Net Value Per Share (C)*
20. *X66: Current Liabilities/Equity*

Feature Selection by Domain Knowledge

The main rationale behind choosing the following features is to highlight the main picture of the company. The ratios involving asset growth tell us how efficient the assets of the company are growing to help the company to finance its debt obligations i.e., the interest payments. The ratios pertaining to interest with respect to the revenues of the company helps us to establish what % of revenues are paid out as interests. A higher percent can put the company on the brink of bankruptcy unless the growth of assets is outstanding. The Debt ratio %, Net Worth/Assets, and Long-term fund suitability ratio tells us how much debt a company has w.r.t to its equity and how efficiently they can pay their debt using their assets. A low D/E and high Net Worth/Assets and Long-term fund suitability ratio is preferred. The ratios pertaining to the working capital of the company such as Average Collection Days: Days Receivable Outstanding, Long-term Liability to Current Assets, Current Liability to Current Assets highlights the fact that how the company is using its current assets to pay their current liabilities such as short-term debt, which constitutes a company's total debt. In other words, it tells us how a company is efficiently financing its working capital, which is an important measure as it can help us predict which company is on the brink of bankruptcy.

Applying domain knowledge to the problem at hand, I have chosen the following features as the most important features to be considered:

Selected Variable	Selected Variable Name & Description	Selected Data Type (Categorical/Numerical)
X ₁	ROA(C) before interest and depreciation before interest: Return On Total Assets(C)	Numerical
X ₂	ROA(A) before interest and % after tax: Return On Total Assets(A)	Numerical
X ₃	ROA(B) before interest and depreciation after tax: Return On Total Assets(B)	Numerical
X ₈	After-tax net Interest Rate: Net Income/Net Sales	Numerical
X ₁₄	Interest-bearing debt interest rate: Interest-	Numerical

	bearing Debt/Equity	
X ₂₉	Total Asset Growth Rate	Numerical
X ₃₁	Total Asset Return Growth Rate Ratio: Return on Total Asset Growth	Numerical
X ₃₅	Interest Expense Ratio: Interest Expenses/Total Revenue	Numerical
X ₃₆	Total debt/Total net worth: Total Liability/Equity Ratio	Numerical
X ₃₇	Debt ratio %: Liability/Total Assets	Numerical
X ₃₈	Net worth/Assets: Equity/Total Assets	Numerical
X ₃₉	Long-term fund suitability ratio (A) : (Long-term Liability + Equity)/Fixed Assets	Numerical
X ₄₀	Borrowing dependency: Cost of Interest-bearing Debt	Numerical
X ₄₇	Average Collection Days: Days Receivable Outstanding	Numerical
X ₆₇	Long-term Liability to Current Assets	Numerical
X ₇₉	Equity to Long-term Liability	Numerical
X ₈₄	Current Liability to Current Assets	Numerical
X ₈₅	Liability-Assets Flag	Categorical [1 if Total Liability exceeds Total Assets, 0 otherwise]
X ₉₂	Degree of Financial Leverage (DFL)	Numerical
X ₉₃	Interest Coverage Ratio (Interest expense to EBIT)	Numerical

Table 4 Selected Features based on domain Knowledge

Chapter-6 Evaluation Metrics

A classifier's performance can be judged by many evaluation metrics; however, one needs to be careful in choosing appropriate metrics depending upon the problem at hand.

In this project, the problem statement is to “*Identify Bankrupt Companies*”. Nevertheless, one needs to cognizant of the fact that no one would like to have large numbers of *False Negatives* (i.e., labelling *Bankrupt Companies* and *non-Bankrupt*) in such a situation due to the huge financial cost of such a mistake. One would like to have high detection rate of *Bankrupt Companies* while not generating *False Positives* (i.e., labelling *non-Bankrupt Companies* and *Bankrupt*) as it would mean unnecessary problems for a well-functioning company.

In the interest of continuity, I would like to re-iterate that:

Positive Class → *Class Label: 1 (Bankrupt)*

Negative Class → *Class Label: 0 (Non-Bankrupt)*

1) Misclassification Error

The overall success rate on a given test set is the number of correct classifications divided by the total number of classifications.

$$\text{Success Rate} = \frac{TP + TN}{TP + TN + FP + FN}$$

This is also referred to as the *overall recognition rate* of the classifier, that is, it reflects how well the classifier recognizes instances of various classes. The misclassification rate of a classifier is simply ($1 - \text{recognition rate}$).

$$\text{Misclassification Rate} = \frac{FP + FN}{TP + TN + FP + FN}$$

2) Confusion Matrix

A confusion matrix is a table that is used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. [19] For a binary classifier, the confusion matrix looks like :

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Figure 36 Confusion matrix for a Binary Classifier

3) Precision & Recall

In this project we will primarily use Precision & Recall for evaluation of models. The formulae for Precision & recall are given below:

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \quad precision = \frac{true\ positives}{true\ positives + false\ positives}$$

These metrics are chosen as they align with the business requirements of my project namely : lower number of False Negatives (i.e, high recall) & lower number of False Positives (i.e, high precision). Therefore, a high precision and recall are desired.

To see overall performance, I will also look at F_1 -score which is the Harmonic mean of precision & recall.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

4) Precision-Recall Curve

Precision-Recall curve (PR curve) is a curve which plots *Precision & Recall* values on a 2-axis graph by varying the decision threshold (discussed in Chapter-7) of the model. The curve represents the precision & recall which will be achieved by varying the threshold. The following graph shows a skilful and a dumb classifier on a PR curve. It shall also be noted that an ideal PR curve is one which forms a right angle at the top-right corner. Classifiers which tends towards the top right are better.

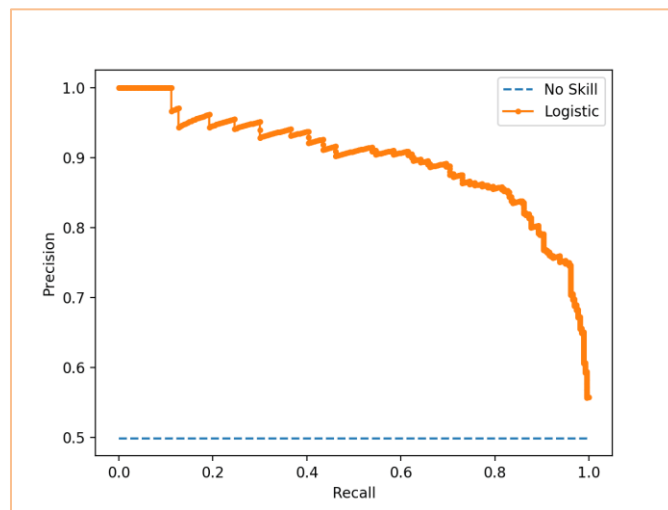


Figure 37 Mock PR Curve

Chapter-7 Dealing with Class Imbalance

Class imbalance is one of the major problems in real-world analytics and modelling. Seldom, will one find data which is perfectly balanced. This situation demands special attention because, If I were to train models without any treatment (under sampling, oversampling etc.) to data, my models will show a classification accuracy close to the percentage frequency of the majority class due to the huge bias it will inherit into itself. Such models are a waste and will not produce any real-life value.

According to the prescribed course book “*Applied Machine Learning by M.Gopal*” [4] there are four main ways to deal with the problem of class imbalance in a dataset :

1. Oversampling
2. Under sampling
3. Threshold moving
4. Ensemble Learning

All these methods aim to do a single thing: reducing bias in the model. In this project I have used methods 1,3 &4. I will explain the bias reduction method used by each in their respective sections.

Class Balancing

- My dataset has a very large imbalance with only 3.2% instances of non-Bankrupt class. Therefore, it is necessary to take steps to ensure that the model does not bias itself towards the majority class.
 - Class balancing can be done by either oversampling or under sampling. In both the sampling, instances of both the classes are made equal in the training set, therefore, providing the model with equal instances to learn from both the classes. This method ensures bias reduction happens even before the models are trained.
 - Here I have used oversampling method using a technique called as **SMOTE-Synthetic Minority Oversampling Technique**. [20]
 - Synthetic instances are made for the minority class (i.e, Bankrupt Class) to match the number of instances of the majority class.
-

SMOTE

- SMOTE [21] is an oversampling technique but works differently than typical oversampling method used in ML.
- In classical oversampling technique, the minority data is duplicated from the minority population. While it increases the number of data, it does not give any new information or variation to the machine learning model.
- SMOTE works by utilizing *k-nearest neighbor algorithm* [22] to create synthetic data. SMOTE first starts by choosing a random data instance from the minority class, then *k*-nearest neighbors (typically 5) from the data are set.
- After the selection of nearest neighbors, lines are drawn between the sampled observations of the minority class. The synthetic points could lie anywhere on those lines. I have made the following illustration, showing the feature space, for better understanding:

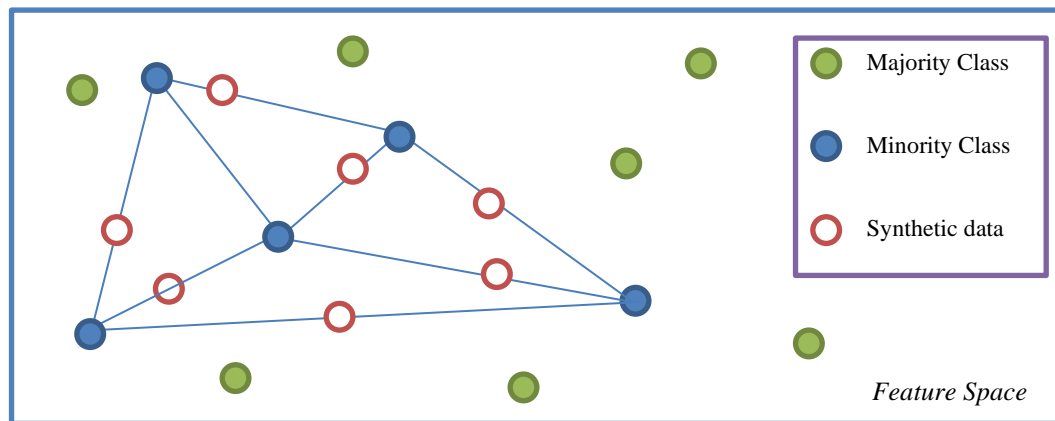


Figure 38 Working of SMOTE algorithm

- In the above figure made by me, one can see that the synthetic data is created between the 5 randomly selected minority class data points.
- In Python, I used `SMOTE()` function provided by `imblearn.over_sampling` library.

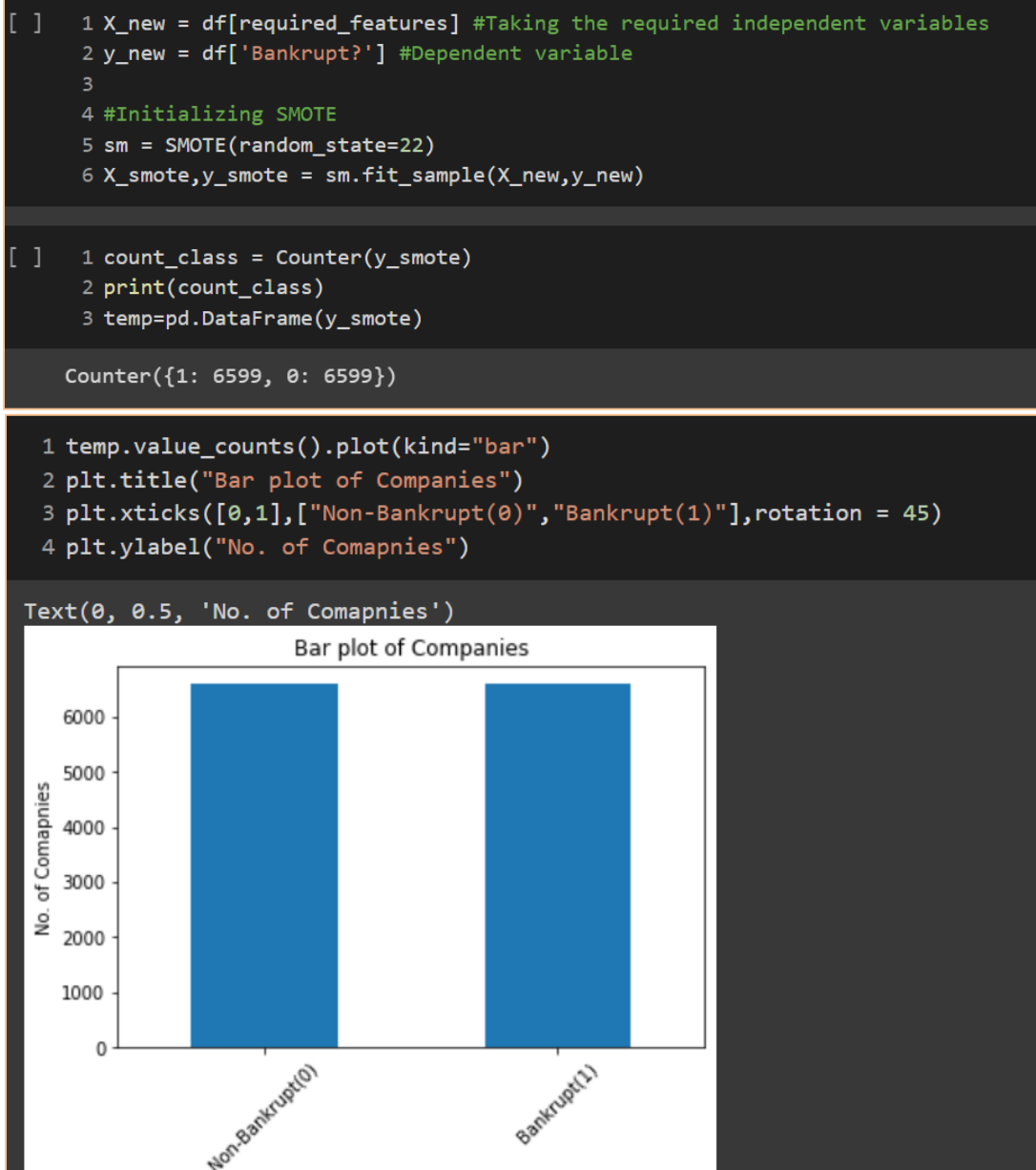


Figure 39 SMOTE Code Snippet & Result

Threshold Moving

Classification models result in the prediction of class labels. However, many machine learning algorithms predict a probability or scoring of class membership. This means models attach a numeric score to each unseen data point which could vary between 0 & 1. Consequently, class

labels are assigned based on a threshold (*typically 0.5*), where all values greater than or equal to the threshold are mapped to one class while others are mapped to the second class. For those classification problems which have severe class imbalance, default threshold renders poor performance as the model is inherently biased towards the majority class. Such a bias is reflected by the fact that many test set data points will have score favoring the majority class. To avoid such a bias, we lower the threshold or decision boundary of the model to a different number. The process of finding the *optimal threshold point* involves the study of ROC (Receiver Operating Characteristics) & PR(Precision-Recall) curves. In the interest of this project I will be using the PR curves primarily for my evaluation and thresholding purposes.

1) Finding Optimal Threshold Point

The process of finding the *optimal threshold point* on a PR Curve is straightforward, one needs to simply find a point on the curve where F_1 score is maximum. The threshold corresponding to the maximum F_1 score will be related to a threshold value, which will then become our *optimal threshold point*. The following code snippet shows a custom function made by me to get the Optimal point along with other important classification metrics/reports.

```
1 #our_results
2 def our_results(X_train,X_test,y_train,y_test,y_pred):
3     Y_pred = [1 if i >0.5 else 0 for i in y_pred]
4     print("=====")
5     print("=====BEFORE THRESHOLD=====")
6     results(y_test,Y_pred)
7     fpr,tpr, thresholds = roc_curve(y_test,y_pred)
8     score= roc_auc_score(y_test, Y_pred)
9     threshold_ROC=Find_Optimal_CutoffROC(tpr,fpr,y_pred,thresholds)
10    precision, recall, th = precision_recall_curve(y_test, y_pred)
11    threshold_PR=Find_Optimal_CutoffPR(precision,recall,th,y_test)
12    print("=====")
13    print('ROC AUC score:',score)
14    print("Threshold for ROC: ",threshold_ROC)
15    print("=====")
16    Y_pred = [1 if i >threshold_ROC else 0 for i in y_pred]
17    print("=====AFTER THRESHOLD ROC=====")
18    results(y_test,Y_pred)
19    print("=====")
20    Y_pred = [1 if i >threshold_PR else 0 for i in y_pred]
21    print("=====AFTER THRESHOLD PR CURVE=====")
22    results(y_test,Y_pred)
```

Figure 40 Code snippet for custom function to display classification reports

```

1 def Find_Optimal_CutoffPR(pr,re,th,yvl):
2     # convert to f score
3     fscore = (2 * pr * re) / (pr+re+0.000001)
4     # locate the index of the largest f score
5     ix = np.argmax(fscore)
6     print("=====")
7     print('Best Threshold Precision-Recall Curve=%f, F-Score=%.3f' % (th[ix], fscore[ix]))
8     prcurve(yvl,pr,re,ix)
9     print("=====")
10    return th[ix]

```

Figure 41 Code snippet for finding Optimal Threshold point from PR curve

Ensemble Learning

Ensemble learning methods use a combination of learned models (classifiers) with the aim of creating an improved composite model. Many models are used to output a single result which could be either through majority voting etc. In this project I will show results for ensemble methods like Random forests etc.

This method reduces bias by considering the decision of various models which thereby averages out the inherent bias, if any, in one single model.

Chapter-8 Models Used

In this Chapter I will describe, in brief, all the models which I have used for modelling in the next chapter. Detailed explanations and working of each model can be found in the prescribed course book : “*Applied Machine Learning by M.Gopal*” [4].

2) Logistic Regression

Logistic regression [23] is a statistical model that uses a logistic function to model a binary dependent variable. Logistic Regression is used when the dependent variable (target) is categorical.

Parameters used in this project:

- “c”: Inverse of regularization strength.
- “penalty”: Used to specify the norm used in the penalization
- “solver”: Algorithm to use in the optimization problem.

3) Support Vector Classifier

Support vector machine [24] algorithm works by finding a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points into two different planes.

Parameters used in this project:

- “c”: Regularization parameter
- “kernel”: Specifies the kernel type to be used in the algorithm

4) Gaussian Naïve Bayes

Naive Bayes [25] classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. Naive Bayes can be extended to real-valued attributes, by assuming a Gaussian distribution. This extension of naive Bayes is called Gaussian Naive Bayes. Gaussian (or Normal distribution) is the easiest to work with because one only needs to estimate the mean and the standard deviation from the training data.

5) Decision Tree

Decision trees [26] can be applied to both regression and classification problems. In a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs. In interpreting the results of a classification tree, we are often interested not only in the class prediction corresponding to a particular terminal node region, but also in the class proportions among the training observations that fall into that region. A classification decision tree is grown by recursive binary splitting. We plan classification to assign an observation in each region to the most commonly occurring error rate class of training observations in that region, the classification error rate is simply the fraction of the training observations in that region that do not belong to the most common class (popularly known as impurity in a node)

Parameters used in this project:

- “criterion”: The function to measure the quality of a split “kernel”: Specifies the kernel type to be used in the algorithm

6) Random Forest

Random forest [27] is an ensemble model using bagging as the ensemble method and decision tree as the individual model. Random forest consists of many individual decision trees that operate as an ensemble. Each individual tree in the random forest outputs a class prediction and the class with the most votes becomes our model’s prediction

Parameters used in this project:

- “n_estimators”: The number of trees in the forest.
- “criterion”: The function to measure the quality of a split.
- “max_features”: The number of features to consider when looking for the best split
- “bootstrap”: Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.
- “max_depth”: The maximum depth of the tree.
- “min_samples_split”: The minimum number of samples required to split an internal node.
- “min_samples_leaf”: The minimum number of samples required to be at a leaf node.
- “max_leaf_nodes”: Grow trees with max_leaf_nodes in best-first fashion. Best nodes

are defined as relative reduction in impurity.

7) XGBoost

XGBoost [28] is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. It is an implementation of gradient boosting machines created by Tianqi Chen. All in all, it's an optimized and powerful version of Gradient Boosting algorithm used in Decision trees.

Parameters used in this project:

- “max_depth”: The maximum depth of a tree. Used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.

8) Extra Trees Classifier

Extra Trees classifier (also known as Extremely Randomized Trees) are similar to a Random Forest classifier. It is used to introduce more variation into the ensemble. Some changes are made from a conventional Random forest algorithm. Each decision stump will be built with the following criteria:

- All the data available in the training set is used to build each stump.
- To form the root node or any node, the best split is determined by searching in a subset of randomly selected features of size $\sqrt{\text{number of features}}$. The split of each selected feature is chosen at random.
- The maximum depth of the decision stump is one.

Parameters used in this project:

- “n_estimators”: The number of trees in the forest.
- “criterion”: The function to measure the quality of a split.
- “max_features”: The number of features to consider when looking for the best split

9) Neural Network

Neural networks [29] reflect the behavior of the human brain, allowing computer to recognize patterns. Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

Chapter-9 Predictive Modelling & Results

In this section I will discuss about the models deployed by me. For a detailed plan please refer back to [Figure 1 Workflow of this project](#).

In this project I have maintained a 70:30 train, test split for the data set.

```
1 #Splitting training and testing data
2 X_train, X_test, y_train, y_test = train_test_split(X_smote,y_smote,test_size=0.3, random_state=42)
```

Figure 42 Code snippet for Train-Test Split

Before Mid-Sem

My Jupyter Notebook for Mid-Sem work can be found in Appendix-A.

1) Modelling & Hyperparameter Tuning

Before Mid Semester Report submission, I had deployed models based on Oversampled data (by SMOTE) and had done feature selection by Variance Threshold Method. Following algorithms were deployed:

- SVM [30]
- XGBoost [31]
- Random Forest [32]
- Logistic Regression [33]
- Naïve Bayes Gaussian [34]
- Decision tree [35]

Following figure shows a code snippet of models:

```
1 my_models = {
2     'SVM': {'model': svm.SVC(gamma='auto', C=5, kernel='rbf'), 'params': {'C': [1,5,10]}},
3     'xgboost': {'model': xgb.XGBClassifier(), 'params': {'max_depth': [4,6,8]}},
4     'random_forest': {'model': RandomForestClassifier(), 'params': {'n_estimators': [1,5]}},
5     'logistic_regression': {'model': LogisticRegression(solver='liblinear', multi_class='auto'), 'params': {'C': [1,5]}},
6     'naive_bayes_gaussian': {'model': GaussianNB(), 'params': {}},
7     'decision_tree': {'model': DecisionTreeClassifier(), 'params': {'criterion': ['gini', 'entropy']}}
8 }
```

Figure 43 List of models

All these models were passed through a `GridSearchCV` [36] function (for Hyperparameter optimisation) which allows the user to perform Grid Search with cross validation. I had chosen 2-fold cross validation. Metric used to compare models was misclassification rate as the dataset has now been balanced.

```

1 scores = []
2
3 for model_name, mp in my_models.items():
4     clf = GridSearchCV(mp['model'], mp['params'], cv=2, return_train_score=False)
5     clf.fit(X_smote, y_smote)
6     scores.append({
7         'model': model_name,
8         'best_score': clf.best_score_,
9         'best_params': clf.best_params_
10    })
11
12 df_model = pd.DataFrame(scores, columns=['model', 'best_score', 'best_params'])
13 df_model

```

Figure 44 Code Snippet for GridSearchCV and result logging

➤ The results of the GridSearchCV is as follows:

	model	best_score	best_params
0	SVM	0.509926	{'C': 1}
1	xgboost	0.944916	{'max_depth': 8}
2	random_forest	0.927262	{'n_estimators': 5}
3	logistic_regression	0.554251	{'C': 1}
4	naive_bayes_gaussian	0.520155	{}
5	decision_tree	0.918094	{'criterion': 'gini'}

Figure 45 Results of GridSearchCV

- We see that XGBoost [31] Algorithm performs the best with *max_depth*=8.

2) Best Model Results:

The best model is the XGBoost Model as it has the highest accuracy rate amongst all. I achieved Testing set accuracy at 95.43%. Test set results from the XGBoost Model:

Class	Precision	Recall	F1-Score
0	0.97	0.94	0.95
1	0.94	0.97	0.95

Table 5 Results of Best Model: XGBoost

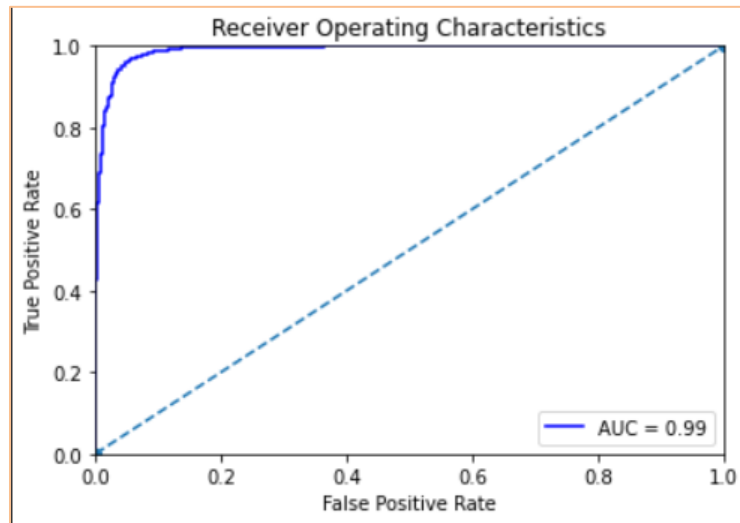


Figure 46 ROC Curve of Best Model: XGBoost

After Mid-Sem

My Jupyter Notebook for after Mid-Sem work can be found in Appendix-A.

After Mid-Sem I have done feature selection based on *ANOVA Test & Domain Knowledge*. Consequently, two types of models were deployed namely *Ensemble Models & Cost-Sensitive Models*.

1) Cost – Sensitive Modelling & Hyperparameter Tuning

As discussed earlier, cost-sensitive learning pertains to the methodology of modelling where a custom optimal threshold level is used for assigning class labels to test data. Hyperparameter Tuning was done for Logistic Regression & Support Vector Classifier using GridSearchCV as shown earlier. Following models were used:

- Logistic Regression
- Support Vector Classifier
- Neural Network

Following figures show the code snippet for the declaration of Logistic Regression & Support Vector Classifier along with the architecture of the Neural Network used:

```
1 my_models = {
2     'SVM': {'model': svm.SVC(gamma='auto', C=5, kernel='rbf'),
3           'params': {'C': [1,2,3,4,5,6,7,8,9,10],
4                     "kernel": ["linear", "poly", "sigmoid"]}},
5     'logistic_regression': {'model': LogisticRegression(solver='liblinear', multi_class='auto'),
6                           'params': {'C': [1,2,3,4,5,6,7,8,9,10],
7                                     "penalty": ["l1", "l2"]}}
8 }
```

Figure 47 Code snippet for models used

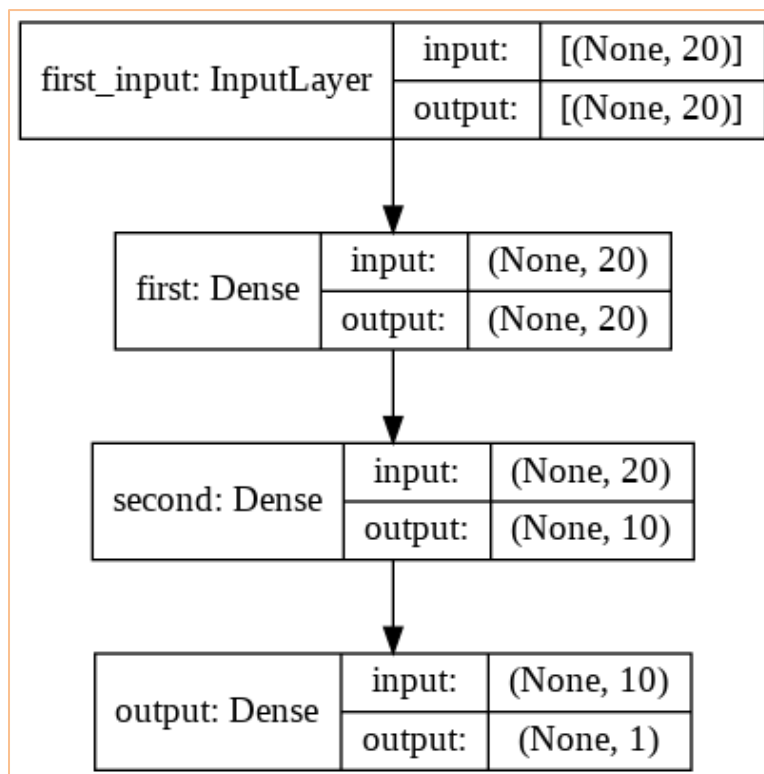
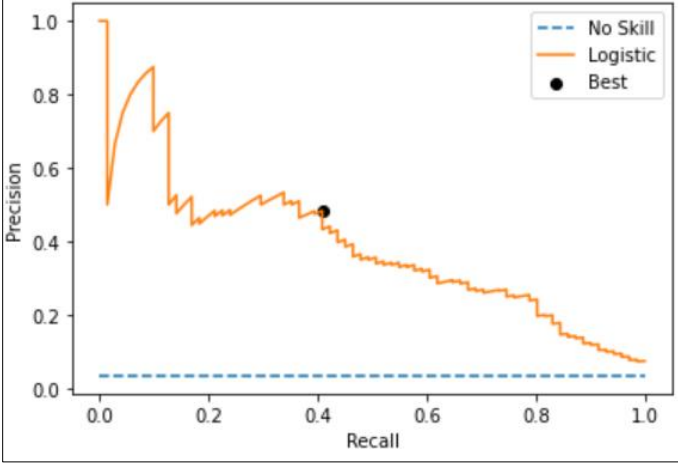
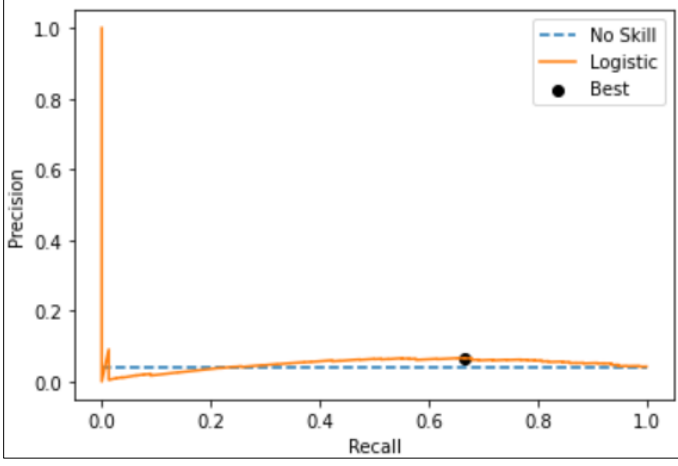


Figure 48 Architecture of the Neural Network used

2) Best models and results (Cost-Sensitive Learning)

Here I will discuss the results for the best selected from SVC & Logistic models and will show the impact of threshold moving on the models. Following table shows the results:

ANOVA Feature Selected	Domain Knowledge Feature Selected
<i>Best Selected Model by GridSearchCV</i>	
Logistic Regression [c=1; penalty=l1]	SVC [c=1; kernel=rbf; gamma=auto]
<i>Training Set Accuracy Score of Best Model</i>	
96.77%	96.18%
<i>Classification report of Best model</i>	
<pre> precision recall f1-score support 0.0 0.97 1.00 0.98 1975 1.0 0.78 0.10 0.17 71 accuracy 0.97 2046 macro avg 0.87 0.55 0.58 2046 weighted avg 0.96 0.97 0.96 2046 </pre>	<pre> precision recall f1-score support 0.0 0.96 1.00 0.98 1968 1.0 0.00 0.00 0.00 78 accuracy 0.96 2046 macro avg 0.48 0.50 0.49 2046 weighted avg 0.93 0.96 0.94 2046 </pre>
<i>PR Curve of Best Model</i>	
	
<i>Selected Threshold & Corresponding F-score</i>	
Threshold = 0.144	Threshold = 0.033

F-Score = 0.443					F-Score = 0.121								
Classification report after threshold shifting													
<div>precision recall f1-score support</div>					<div>precision recall f1-score support</div>								
0.0	0.98	0.98	0.98	1975	0.0	0.98	0.62	0.76	1968				
1.0	0.47	0.39	0.43	71	1.0	0.07	0.67	0.12	78				
accuracy			0.96	2046	accuracy			0.62	2046				
macro avg			0.73	0.69	0.71	2046	macro avg			0.52	0.64	0.44	2046
weighted avg			0.96	0.96	0.96	2046	weighted avg			0.94	0.62	0.74	2046

Table 6 Results for Cost-Sensitive Learning (SVC & LR)

3) Neural Network Results (Cost-Sensitive Learning)

Now I will present the results for Neural Network Model. I have built a sequential Neural Network with total 4 layers all of which have *Sigmoid activation*. The optimizer chosen is *Adam* and the loss is calculated as *binary cross entropy*. I have used 20% of training data as validation data here, 500 epochs & 100 as bucket size.

- Sigmoid Activation: The sigmoid function generates a smooth non-linear curve that squashes the incoming values between 0 and 1.
 - Sigmoid Function : $f(x) = \frac{1}{1+e^{-x}}$
 - Sigmoid graph:

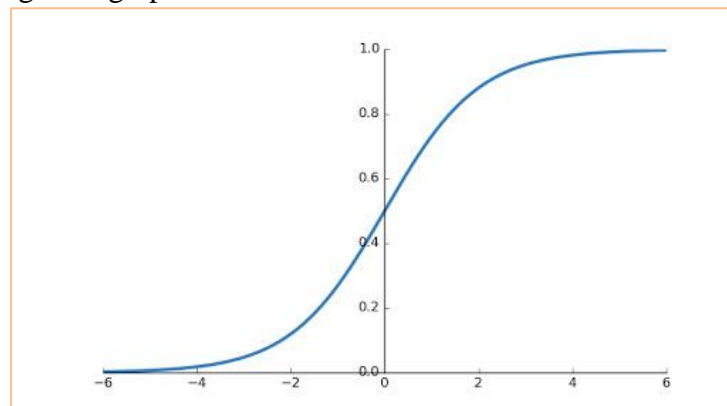


Figure 49 Mock Sigmoid Graph

- Binary Cross Entropy loss : Cross-entropy is a measure of the difference between two probability distributions for a given random variable or set of events. Cross-entropy is

widely used as a loss function when optimizing classification models. In Binary Neural Network classification problems, we use Binary Cross-entropy loss

- Binary Cross Entropy formula:

$$\text{Loss} = -\frac{1}{\text{output size}} \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Figure 50 Formula for Binary Cross Entropy Loss

```
1 model = Sequential()
2 model.add(Dense(input_dim, input_shape=(input_dim,), activation='sigmoid',name="first"))
3 model.add(Dense(input_dim/2, activation='sigmoid',name="second"))
4 model.add(Dense(1, activation='sigmoid',name="output"))
5
6 model.summary()
```

Figure 51 Activation and stages of the Neural Network

```
1 model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Figure 52 Compilation configuration of Neural Network

```
1 history_dropout = model.fit(
2     X_domain_train,
3     y_domain_train,
4     epochs=500,
5     batch_size=100,
6     validation_split=0.2,
7     shuffle=True,
8     workers=2
9 )
```

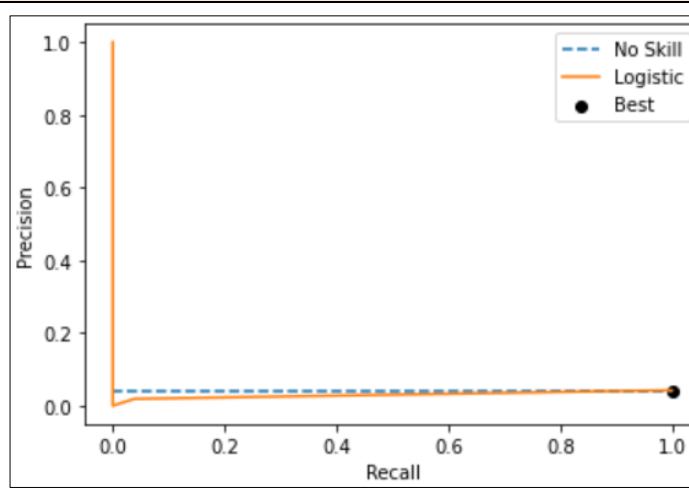
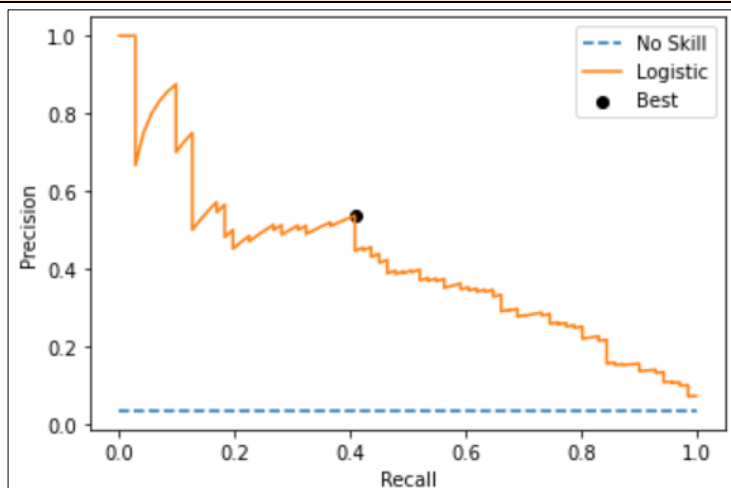
Figure 53 Configuration used for fitting Neural Network

ANOVA Feature Selected	Domain Knowledge Feature Selected
Training Set Accuracy Score of NN	
96.82%	96.18%
Classification report of NN	

	precision	recall	f1-score	support
0.0	0.97	1.00	0.98	1975
1.0	0.88	0.10	0.18	71
accuracy			0.97	2046
macro avg	0.92	0.55	0.58	2046
weighted avg	0.97	0.97	0.96	2046

	precision	recall	f1-score	support
0.0	0.96	1.00	0.98	1968
1.0	0.00	0.00	0.00	78
accuracy			0.96	2046
macro avg	0.48	0.50	0.49	2046
weighted avg	0.93	0.96	0.94	2046

PR Curve



Selected Threshold & Corresponding F-score

Threshold = 0.231

F-Score = 0.464

Threshold = 0.0172

F-Score = 0.082

Classification report after threshold shifting

	precision	recall	f1-score	support
0.0	0.99	0.89	0.94	1975
1.0	0.21	0.83	0.34	71
accuracy			0.89	2046
macro avg	0.60	0.86	0.64	2046
weighted avg	0.97	0.89	0.92	2046

	precision	recall	f1-score	support
0.0	0.96	0.92	0.94	1968
1.0	0.02	0.04	0.03	78
accuracy			0.89	2046
macro avg	0.49	0.48	0.48	2046
weighted avg	0.92	0.89	0.90	2046

Table 7 Results for Cost-Sensitive Learning on Neural Network

4) Ensemble Learning Results

In this section, I will discuss the results of ensemble learning models. Two algorithms were used

namely: Random Forest & Extra Trees classifier

ANOVA Feature Selected	Domain Knowledge Feature Selected
Best Model	
Extra Trees Model [n_estimators=5; criterion="entropy"; max_features=2]	RandomForestClassifier[n_estimators = 2500, bootstrap=False, max_depth=1000, max_features=2, min_samples_leaf=1, min_samples_split=2, max_leaf_nodes=280]
Training Set Accuracy Score of Best Model	
96.18%	97.06%
Classification report of Best model	
<pre> precision recall f1-score support 0.0 0.98 0.99 0.98 1650 1.0 0.37 0.25 0.30 55 accuracy 0.96 1705 macro avg 0.67 0.62 0.64 1705 weighted avg 0.96 0.96 0.96 1705 </pre>	<pre> precision recall f1-score support 0.0 0.97 1.00 0.99 1647 1.0 1.00 0.14 0.24 58 accuracy 0.97 1705 macro avg 0.99 0.57 0.61 1705 weighted avg 0.97 0.97 0.96 1705 </pre>
Confusion Matrix	
<pre> CONFUSION Matrix [[1626 24] [41 14]] </pre>	<pre> CONFUSION Matrix [[1647 0] [50 8]] </pre>

Table 8 Results of Ensemble Learning

Chapter-10 Conclusion

- We observed that the oversampled data model trained on XGBoost Algorithm provides with the best *F1-score* for `class:1` at 0.95 with precision and recall at 0.94 & 0.97 for `Class:1` respectively. This implies that 97% of times will the model be able to predict a Bankrupt Company among the pool of actual Bankrupt companies, which is impressive given the imbalance inherited.
 - However, one must know in real world, models will not expend time in balancing the classes via sampling methods and will have to rely on other means to reduce majority class bias.
 - Among other bias reducing techniques deployed by me, the best model came from the *Threshold Moving Method* where Logistic Regression has provided with the highest recall for the minority `class:1` at 0.39 and *F1 score* at 0.45.
 - It is noticed that after applying *Threshold Moving Method* every model shows a marked performance improvement in terms of better precision & recall for the minority class.
 - Domain-knowledge selected features have not performed well in *Threshold-Moving Method* while it did perform average in the *Ensemble Learning Method*.
-

Appendix-A: Important Links

1. Link to Dataset Source:

<https://archive.ics.uci.edu/ml/datasets/Taiwanese+Bankruptcy+Prediction>

References

- [1] A. TUOVILA. [Online]. Available: <https://www.investopedia.com/terms/b/bankruptcy.asp>.
- [2] Baker McKenzie, "Global Restructuring & Insolvency Guide - Taiwan".
- [3] D. L. C.-C. Liang, "Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study".
- [4] P. M. Gopal, Applied Machine Learning, McGraw Hill Education, 2018.
- [5] [Online]. Available: <https://www.python.org/>.
- [6] [Online]. Available: <https://pandas.pydata.org/>.
- [7] [Online]. Available: <https://numpy.org/>.
- [8] [Online]. Available: <https://matplotlib.org/>.
- [9] "Seaborn Open Source," [Online]. Available: <https://seaborn.pydata.org/>.
- [10] [Online]. Available: <https://scikit-learn.org/stable/>.
- [11] "Keras Open Source," [Online]. Available: <https://keras.io/>.
- [12] [Online]. Available: <https://www.tensorflow.org/>.
- [13] "github/Auto-Viz PyData," [Online]. Available: <https://github.com/AutoViML/AutoViz>.
- [14] "ROA," Investopedia, [Online]. Available: <https://www.investopedia.com/terms/r/returnonassets.asp>.
- [15] "Cashflow -Investopedia," [Online]. Available: <https://www.investopedia.com/terms/c/cashflow.asp>.
- [16] "Machine Learning Mastery," [Online].
Available: <https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/#:~:text=Feature%20selection%20is%20the%20process,the%20performance%20of%20the%20model..>
- [17] "Towards Data Science," [Online].
Available: <https://towardsdatascience.com/why-how-and-when-to-apply-feature-selection-e9c69adfabf2#:~:text=Variance%20Threshold,has%20very%20little%20predictive%20power.&text=Variance%20Threshold%20doesn%27t%20consider,features%20with%20the%20target%20variable..>
- [18] "Machine Learning Mastery," [Online]. Available: <https://machinelearningmastery.com/feature-selection->

with-numerical-input-data/.

[19] [Online]. Available:

<https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/#:~:text=A%20confusion%20matrix%20is%20a,related%20terminology%20can%20be%20confusing..>

[20] [Online]. Available:

<https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/>.

[21] K. W. B. O. H. W. P. K. N. V. Chawla, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*.

[22] "k-Nearest Neighbor Algorithm," Wikipedia.com, [Online].

Available:

https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#:~:text=In%20statistics%2C%20the%20k%2Dnearest,training%20examples%20in%20data%20set..

[23] S. Swaminathan, "towardsdatascience.com," [Online]. Available: <https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc>.

[24] S. Ray, "www.analyticsvidhya.com," [Online]. Available: <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>.

[25] J. Brownlee, "machinelearningmastery.com," [Online]. Available:

<https://machinelearningmastery.com/naive-bayes-for-machinelearning/#:~:text=This%20extension%20of%20naive%20Bayes,deviation%20from%20your%20training%20data..>

[26] "towardsdatascience.com," [Online]. Available:

<https://towardsdatascience.com/understanding-decision-tree-classifier-7366224e033b>.

[27] T. Yiu, "towardsdatascience.com," [Online]. Available:

<https://towardsdatascience.com/understanding-random-forest-58381e0602d2>.

[28] "towardsdatascience.com," [Online]. Available:

<https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d>.

[29] IBM Cloud, [Online]. Available: <https://www.ibm.com/cloud/learn/neural-networks>.

- [30] [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>.
- [31] [Online]. Available: <https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/>.
- [32] [Online]. Available:
<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>.
- [33] [Online]. Available:
https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic%20regression#sklearn.linear_model.LogisticRegression.
- [34] [Online]. Available:
https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html.
- [35] [Online]. Available:
<https://scikitlearn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decision%20tree#sklearn.tree.DecisionTreeClassifier>.
- [36] [Online]. Available:
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.
- [37] S. Y., "Towards Data Science," [Online]. Available: <https://towardsdatascience.com/3-feature-selection-techniques-of-scikit-learn-c9a5f7eb7364>.

~~End of Document~~