BANKRUPTCY PREDICTION

By

ARJUN BATHLA	**18BEC1236**
V. NIMAL YUGHAN	**18BEC1286**
S. SABARISH	**18BEC1280**

A project report submitted to

Dr SATHIYA NARAYANAN S

SCHOOL OF ELECTRONICS ENGINEERING

in partial fulfilment of the requirements for the course of

CSE3506 – ESSENTIALS OF DATA ANALYTICS

in

B. TECH. - ELECTRONICS AND COMMUNICATION ENGINEERING

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Vandalur – Kelambakkam Road

Chennai – 600127

JUNE 2021

BONAFIDE CERTIFICATE

Certified that this project report entitled "**Bankruptcy Prediction"** is a bonafide work of **Arjun Bathla (18BEC1236), V. Nimal Yughan (18BEC1286) and S. Sabarish (18BEC1280)** who carried out the project work under my supervision and guidance for **CSE3506 – Essentials of Data Analytics.**

Dr SATHIYA NARAYANAN S

Assistant Professor (Senior Grade 1)

School of Electronics Engineering (SENSE),

Vellore Institute of Technology (VIT Chennai)

Chennai – 600 127.

ABSTRACT

Bankruptcy is the state where a financial organization or institution is in a state of insolvency wherein it is unable to repay its creditors. If a company goes bankrupt, it affects the stakeholders of the company, economy of the country and its creditors also face a huge lose. Thus, it is one crucial aspect that a company has to monitor. There are many financial terms involved in determining whether a company has a risk of going bankrupt. For example, return on investments and equity to liability are some of the financial ratios involved in bankruptcy prediction. This project aims to simplify the process of determining bankruptcy by implementing machine learning models to predict it. The scope of this project is to build various supervised learning models on the dataset collected from the Taiwan Economic Journal for the years 1999 to 2009 and perform comparative analysis to determine the model which is best suited. The models in this project are built on the Logistic Regression, Random Forest, XGBoost and CatBoost classifiers. These models are then compared on varies parameters to determine the best suited model.

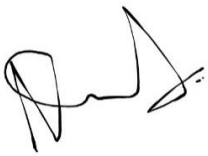
ACKNOWLEDGEMENT

First and foremost, we would like to express our gratitude to our project guide, **Dr. Sathiya Narayanan S**, School of Electronics Engineering, who made this entire course a smooth ride, considering the hard times of COVID-19. Not only did he help us throughout the project, but he also motivated us to build something utilitarian.

We thank **Dr. A. Sivasubramanian**, Dean of School of Electronics Engineering, and **Dr. P. Vetrivelan**, Head of the Department, for the support they provided throughout the project.

We would also like to thank **NASSCOM**, for providing us the opportunity to study this exceptional course.

We thank our families, who made it easier for us to focus on the project, even with everything around us falling apart. Their love and support have made everything possible.


ARJUN BATHLA **V. NIMAL YUGHAN** **S. SABARISH**

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

New start-ups are being formed every day as more people want to build their own organization and be independent. Experience is the most important criterion in running such an organization successfully. Due to lack of such experience, most start-ups go bankrupt at an early stage. So, bankruptcy is one crucial aspect to be monitored in an organization. The need for adopting machine learning models to predict bankruptcy came into light after The Great Recession in 2008. If a company goes bankrupt, it might have a direct impact on the financial market, it might incur great loss to its investors and in some case to may affect the country's economy as well. Researches are still going on in the industry to develop a better model to predict bankruptcy. So, in this project we aim to develop a model that can be helpful for even a start-up with little experience, to predict whether the organization might run a risk of going bankrupt in the near future, based on its current financial statement. The project implements supervised machine learning model based on several classification algorithms, which can classify whether a company will go bankrupt or not.

1.2 PROBLEM STATEMENT AND OBJECTIVES

The performance of any organization over a certain period is measured by a set of numerical values from its financial statements. A better picture is given by the relative magnitudes of different pairs among these values, called Financial Ratios. In this project, we compare and contrast different Classification Algorithms that classify organizations as "will soon go bankrupt" or "will not go bankrupt", based on their financial ratios and determine the best suited classification algorithm using weighted

measurements. This model can be used by corporations to predict bankruptcy and thus take the required measures to improve performance.

Our main objective is to maximize sensitivity of the models, as misclassification of a company that is likely to go bankrupt is not tolerable. We also aim to tackle the problem of class imbalance in the available dataset. Since there are a lot of features in the dataset, we aim to pick the best ones for feature selection as well.

CHAPTER 2
IMPLEMENTATION

The program for the prediction of bankruptcy has been implemented in Python 3, since it allows a convenient way to read data, besides having some of the finest machine learning modules like XGBoost and CatBoost. It also makes manipulating datasets and plotting figures for exploratory data analysis and visualization easier. Since it has a wide variety of libraries for different machine learning techniques like pipelining and SMOTE (Synthetic Minority Over-sampling Technique), a good bankruptcy prediction model can be built.

The program is designed to be used for corporations and organizations on any scale, be it MNCs or mere start-ups.

2.1 IMPLEMENTED METHOD

LIBRARIES USED:

The program makes use of a number of libraries for different purposes.

- The **NumPy** library is used for mathematical operations such as logarithmic and arithmetic computations, as well as for manipulation of datasets and array conversions.
- The **pandas** library is used to build a data frame from the .csv data obtained from UCI Machine Learning Repository, as well as to transform confusion matrices into data frames.
- The **seaborn** library is used to build count plots, boxplots as well as heatmaps.

- From **Scikit-learn**, modules for **Linear Regression, Random Forest Classification** and other essential machine learning methods are used to find the most accurate and sensitive model and then apply that for prediction.
- The newly popular **XGBoost** and **CatBoost** libraries are also used to build prediction models.
- From **ImbLearn** [1] (Imbalanced Learning), SMOTE and pipelining methods are used for oversampling the minority class and then training the models respectively.
- **Matplotlib** is also used for plotting and visualization.

PREPROCESSING AND EXPLORATORY DATA ANALYSIS:

- The **Taiwanese Bankruptcy Prediction Dataset** [2] from UCI Machine Learning Repository is imported as a data frame. Figure 2.1 shows the first 5 out of 6819 instances.

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	...	Net Income to Total Assets	Total assets to GNP price	No-credit Interval
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887	0.808809	0.302646	...	0.716845	0.009219	0.622879
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380	0.809301	0.303556	...	0.795297	0.008323	0.623652
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403	0.808388	0.302035	...	0.774670	0.040003	0.623841
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967	0.808966	0.303350	...	0.739555	0.003252	0.622929
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366	0.809304	0.303475	...	0.795016	0.003878	0.623521

5 rows × 96 columns

Figure 2.1: Dataset Head

- Next, the features are checked for their datatypes, and whether they have any NULL values. Figures 2.2.1, 2.2.2, 2.2.3 and 2.2.4 show the datatypes and non-null value counts for all 96 columns.

```
RangeIndex: 6819 entries, 0 to 6818
Data columns (total 96 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   Bankrupt?                                               6819 non-null   int64
 1    ROA(C) before interest and depreciation before interest 6819 non-null  float64
 2    ROA(A) before interest and % after tax                 6819 non-null   float64
 3    ROA(B) before interest and depreciation after tax      6819 non-null   float64
 4    Operating Gross Margin                                 6819 non-null   float64
 5    Realized Sales Gross Margin                            6819 non-null   float64
 6    Operating Profit Rate                                  6819 non-null   float64
 7    Pre-tax net Interest Rate                              6819 non-null   float64
 8    After-tax net Interest Rate                            6819 non-null   float64
 9    Non-industry income and expenditure/revenue            6819 non-null   float64
 10   Continuous interest rate (after tax)                   6819 non-null   float64
 11   Operating Expense Rate                                 6819 non-null   float64
 12   Research and development expense rate                  6819 non-null   float64
 13   Cash flow rate                                         6819 non-null   float64
 14   Interest-bearing debt interest rate                    6819 non-null   float64
 15   Tax rate (A)                                           6819 non-null   float64
 16   Net Value Per Share (B)                                6819 non-null   float64
 17   Net Value Per Share (A)                                6819 non-null   float64
 18   Net Value Per Share (C)                                6819 non-null   float64
 19   Persistent EPS in the Last Four Seasons                6819 non-null   float64
 20   Cash Flow Per Share                                    6819 non-null   float64
 21   Revenue Per Share (Yuan ¥)                             6819 non-null   float64
```

Figure 2.2.1: Datatypes and Non-Null Counts

```
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
```

Figure 2.2.2: Datatypes and Non-Null Counts

```
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 capitcal Turnover Rate                     6819 non-null   float64
74   Cash Turnover Rate                                 6819 non-null   float64
75   Cash Flow to Sales                                 6819 non-null   float64
```

Figure 2.2.3: Datatypes and Non-Null Counts

```
 76   Fixed Assets to Assets                             6819 non-null   float64
 77   Current Liability to Liability                     6819 non-null   float64
 78   Current Liability to Equity                        6819 non-null   float64
 79   Equity to Long-term Liability                      6819 non-null   float64
 80   Cash Flow to Total Assets                          6819 non-null   float64
 81   Cash Flow to Liability                             6819 non-null   float64
 82   CFO to Assets                                      6819 non-null   float64
 83   Cash Flow to Equity                                6819 non-null   float64
 84   Current Liability to Current Assets                6819 non-null   float64
 85   Liability-Assets Flag                              6819 non-null   int64
 86   Net Income to Total Assets                         6819 non-null   float64
 87   Total assets to GNP price                          6819 non-null   float64
 88   No-credit Interval                                 6819 non-null   float64
 89   Gross Profit to Sales                              6819 non-null   float64
 90   Net Income to Stockholder's Equity                 6819 non-null   float64
 91   Liability to Equity                                6819 non-null   float64
 92   Degree of Financial Leverage (DFL)                 6819 non-null   float64
 93   Interest Coverage Ratio (Interest expense to EBIT) 6819 non-null   float64
 94   Net Income Flag                                    6819 non-null   int64
 95   Equity to Liability                                6819 non-null   float64
dtypes: float64(93), int64(3)
memory usage: 5.0 MB
```

Figure 2.2.4: Datatypes and Non-Null Counts

- It can be seen from the above figures that the dataset has 6819 instances and no null values are present, eliminating the need for removal or imputation. Column 0, 'Bankrupt?' is the label, i.e., the target variable. It is of type Integer, and takes a value of either 1 or 0, 1 meaning that the company will soon go bankrupt, and 0 meaning that it won't. The remaining 95 columns contain the predictor variables, 93 of which take decimal values and 2 take integer values. Since there are only numerical values, classification becomes easier.
- The dataset is also checked for any duplicate entries. Figure 2.3 shows that there are none.

```
In [4]: data.duplicated().sum()
Out[4]: 0
```

Figure 2.3: Duplicate Entry Count

- Next, the class distribution is visualized through a count plot of the target variable, in Figure 2.4. It is observed that the two classes, 0 and 1, are highly imbalanced, class 1 being the minority class.

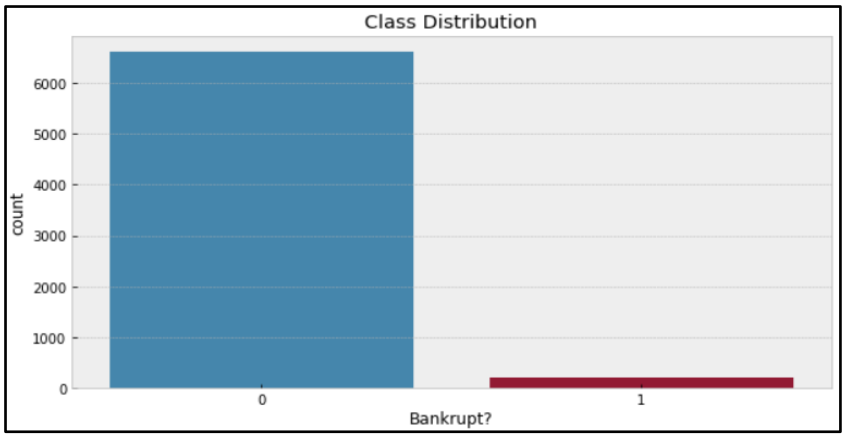


Figure 2.4: Class Distribution

- The data distributions can be seen in Figure 2.5. Horizontal boxplots for each feature have been plotted on a logarithmic scale. Naturally, some of these features have outliers.

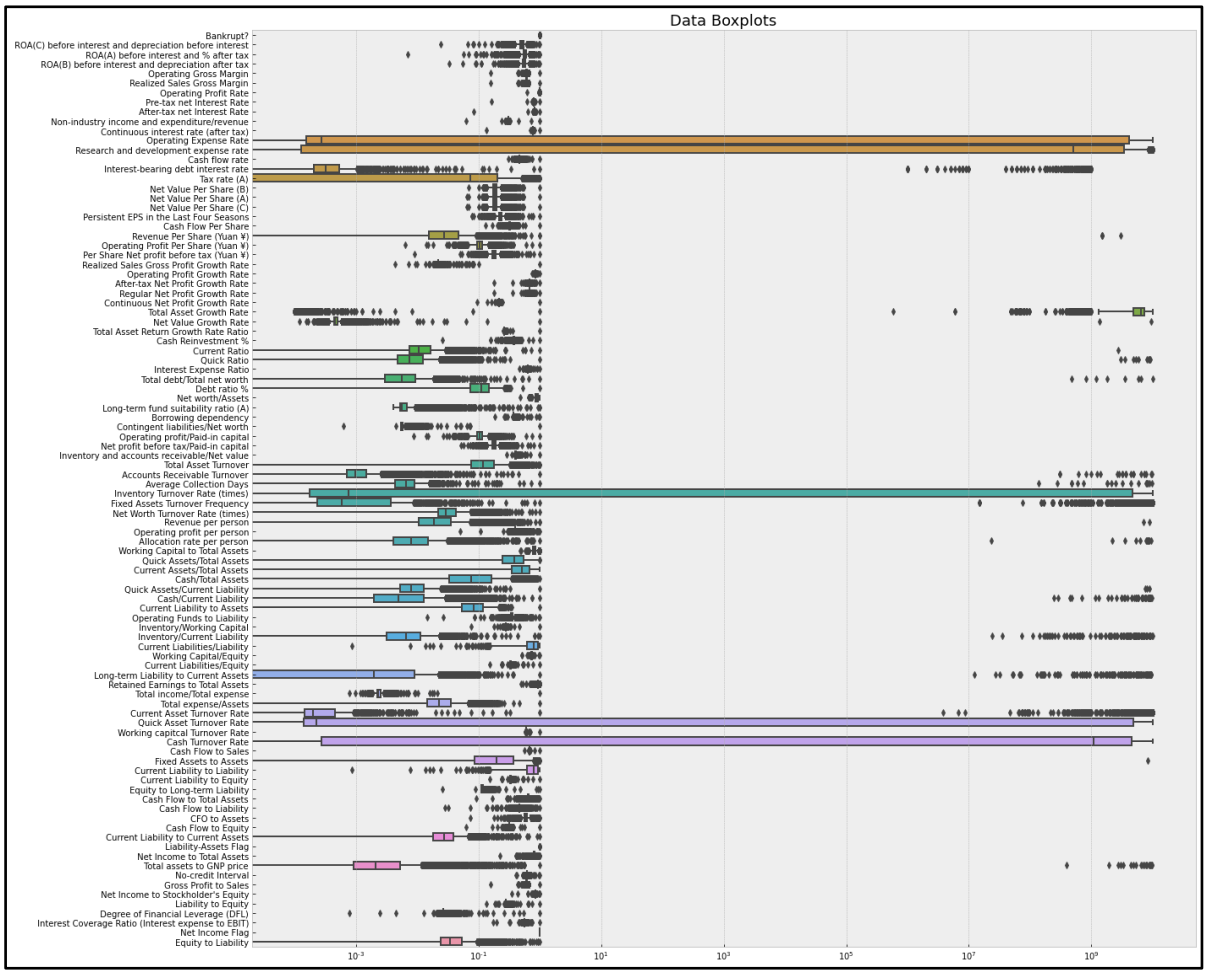


Figure 2.5: Data Boxplots

- These outliers need to be removed, but since there is a very high imbalance in classes, they are removed in such a manner that the class distribution is maintained, and minimal data is removed to retain important information. Figure 2.6 shows the devised outlier removal function, and Figure 2.7 shows the dimensions of the dataset after outlier removal.

Figure 2.8 shows the class distribution after outlier removal. It can be seen that ratio is almost maintained between counts of both classes.

```
def outlier_removal(feature,feat_name,dataset):

    q_25, q_75 = np.percentile(feature, 25), np.percentile(feature, 75)
    iqr = q_75 - q_25
    cut_off = iqr * 20000000000000    # for minimal but effective removal, maintaining the class distribution ratio
    lower, upper = q_25 - cut_off, q_75 + cut_off
    dataset = dataset.drop(dataset[(dataset[feat_name] > upper) | (dataset[feat_name] < lower)].index)
    return dataset
```

Figure 2.6: Outlier Removal Function

```
In [8]: out_rem_data.shape
Out[8]: (6000, 96)
```

Figure 2.7: Dimensions of Outlier-Removed Data

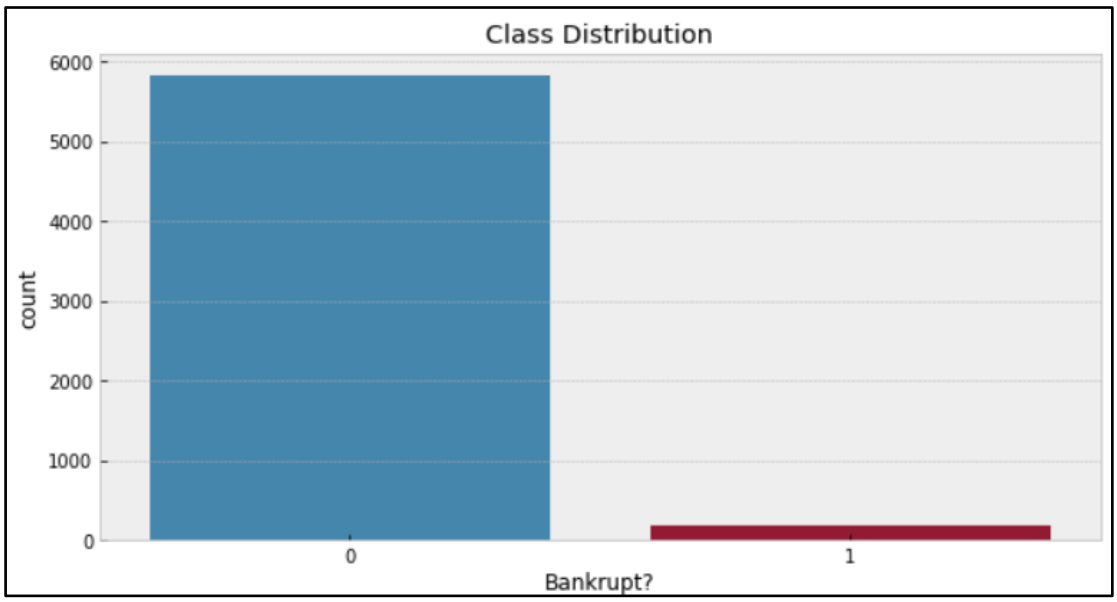


Figure 2.8: Class Distribution of Outlier-Removed Data

- From the outlier-removed dataset, class-wise boxplots of each feature are plotted. Figure 2.9 shows the first 16 boxplots. It is observed that most of these features have almost the same distributions for both the classes.

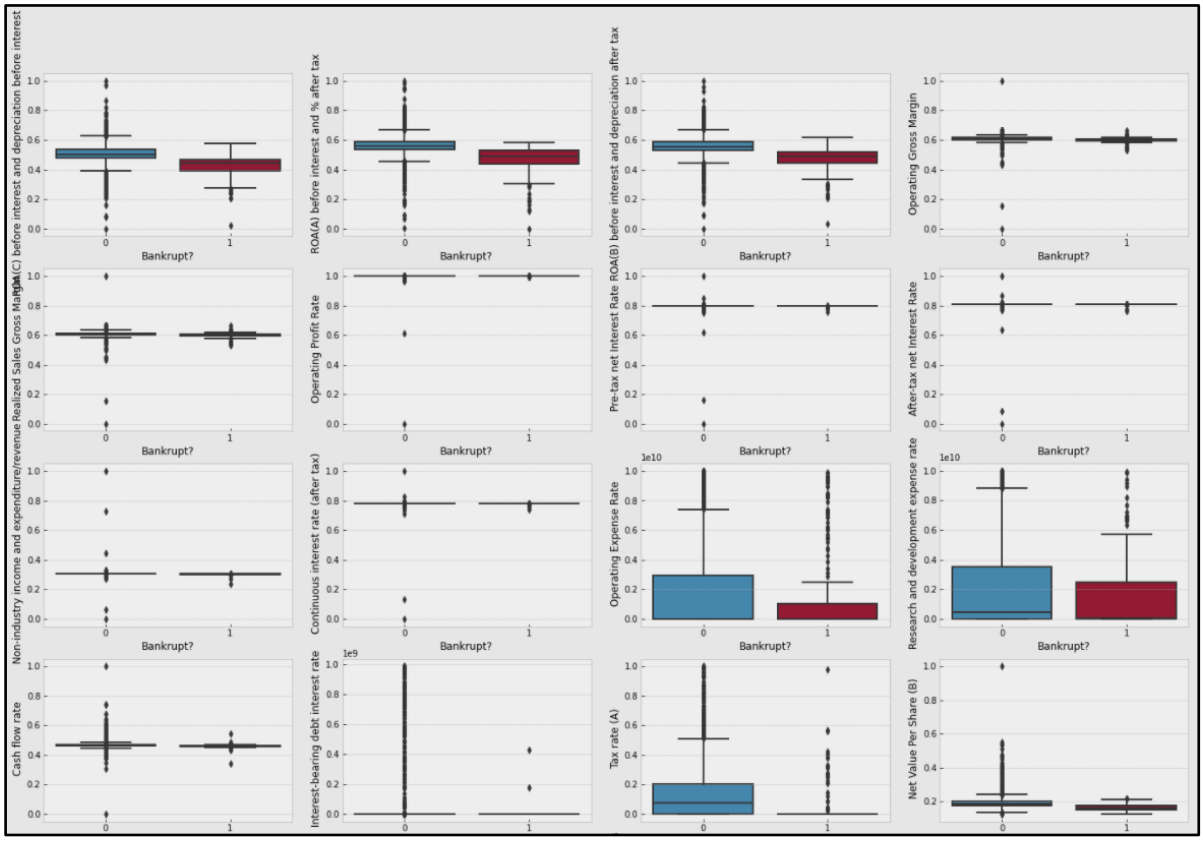


Figure 2.9: Class-Wise Boxplots of First 16 Features

- Out of all 95 features, there are 16 features that were observed, through visual analysis, to have a significant difference in the class-wise distributions. These 16 features along with their indices are:
  - 2. ROA(A) before interest and % after tax
  - 11. Operating Expense Rate
  - 12. Research and development expense rate
  - 15. Tax rate (A)
  - 29. Total Asset Growth Rate

- 37. Debt ratio %
- 38. Net worth/Assets
- 49. Fixed Assets Turnover Frequency
- 54. Working Capital to Total Assets
- 55. Quick Assets/Total Assets
- 57. Cash/Total Assets
- 60. Current Liability to Assets
- 72. Quick Asset Turnover Rate
- 74. Cash Turnover Rate
- 86. Net Income to Total Assets
- 95. Equity to Liability

Figure 2.10 shows the class-wise boxplots for each of these features.

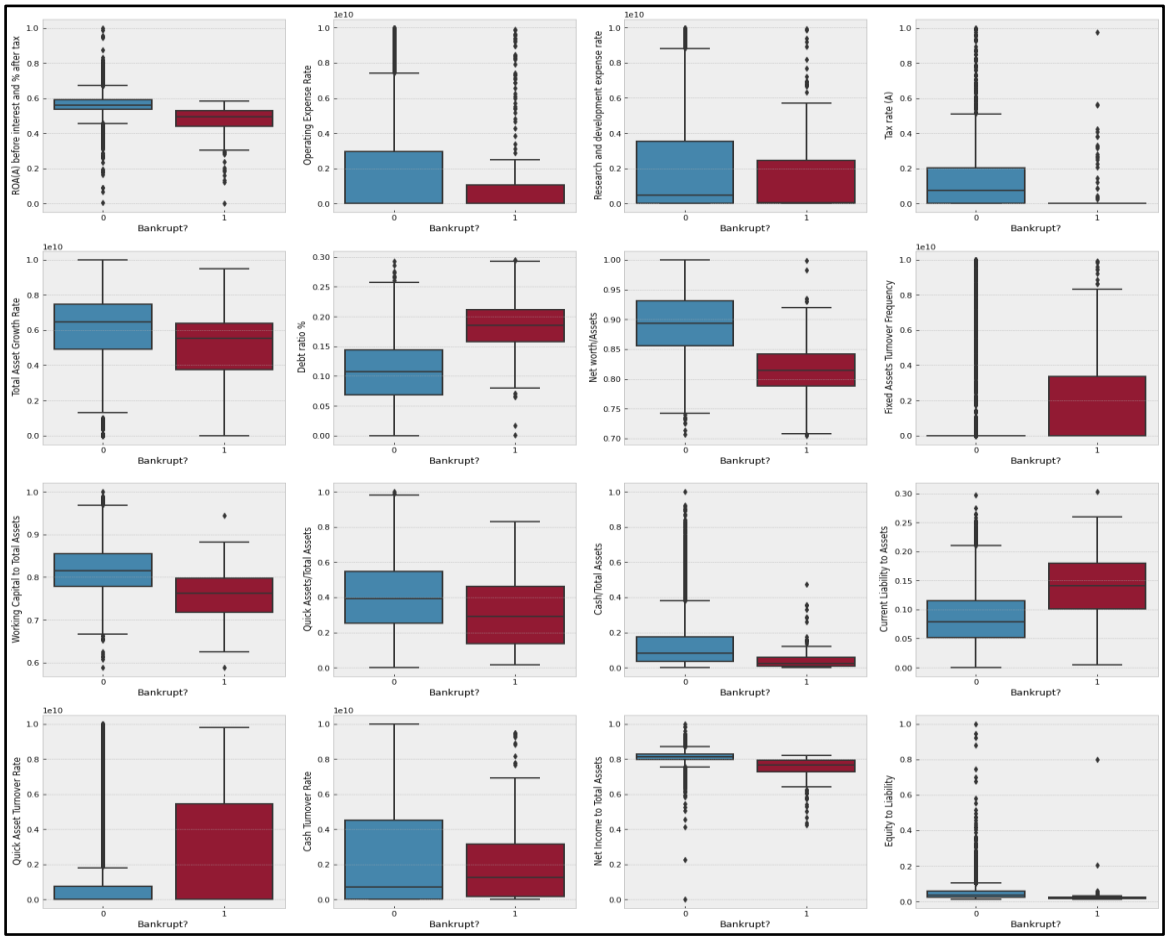


Figure 2.10: Class-Wise Boxplots of Features with Significant Difference in Class Distribution

- Now, the features and labels are separated. Histograms for all features are plotted in Figure 2.11. In some of the plots, a very high skew is observed. This skew is removed in the respective features using logarithmic transformation as defined in Figure 2.12. The condition is that if the skew value is less than -0.5 or greater than 0.5, all the values in the feature undergo logarithmic transformation [3], i.e., natural log of (1 + value). This gives the features a distribution closer to Gaussian, i.e., a bell shape. This helps improve the models' performances.

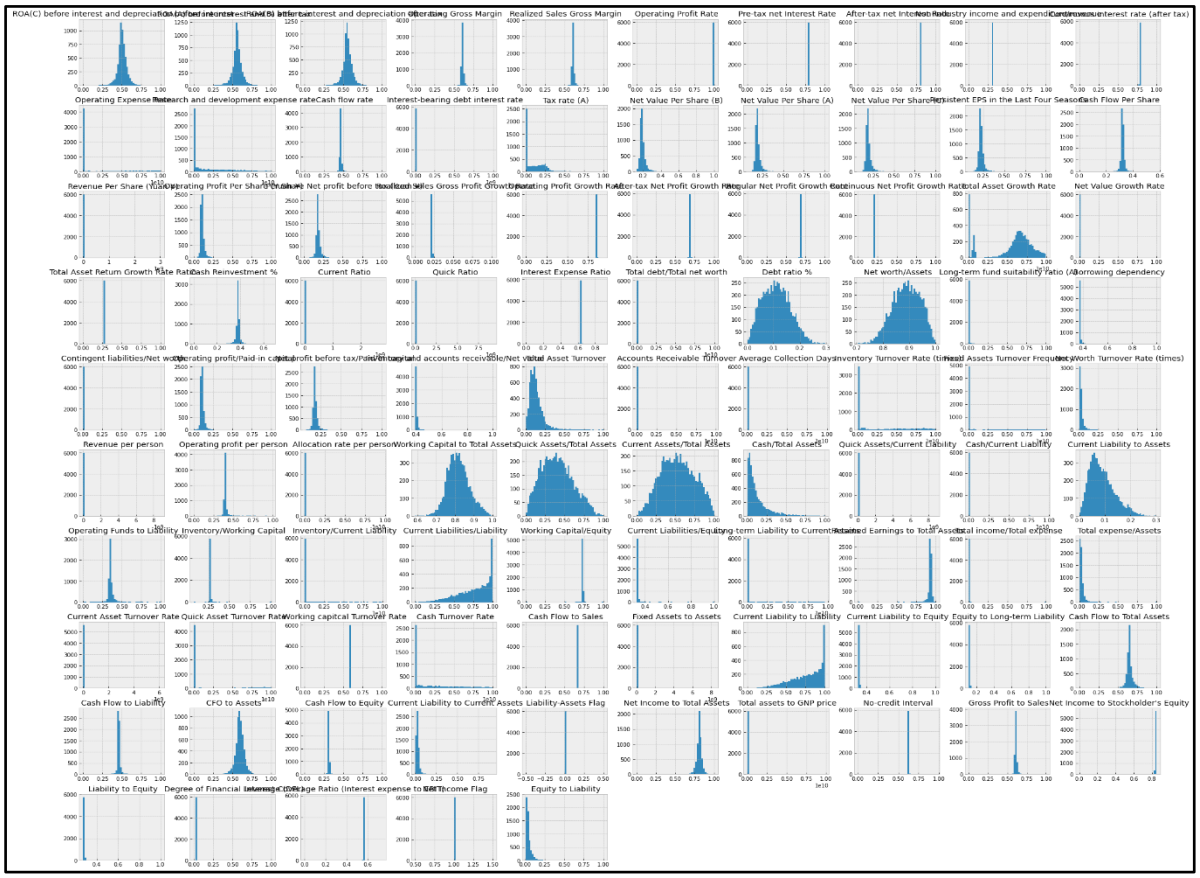


Figure 2.11: Histograms of All Features

```python
def logarithmic_transform(data):

    for column in data:
        skew = data[column].skew()
        if skew > 0.5 or skew < -0.5:
            data[column] = np.log1p(data[column])    # log1p(x) returns ln(1+x)
        else:
            continue

    return data
```

Figure 2.12: Logarithmic Transformation Function

MODELING WITH ALL 95 FEATURES:

- The data is split into training and testing sets with a respective ratio of 90:10. The splitting is stratified, so that in both the sets, the class distribution ratio is maintained. All the models will be trained and tested first using only the training set, i.e., for training and comparison, the training test will be further split into sub-training and sub-testing sets.
- Since the classes are highly imbalanced, the accuracy will show wild variations for different sub-testing sets. Hence, we make use of **Stratified K-Fold Cross Validation Splitting** [4]. This method randomly splits the training set into K sub-testing sets (in this case, K is set to 5). For each sub-testing set, the rest of the instances in the training set form the sub-training set for the respective sub-testing set. After splitting the training set into 5 sub-training and respectively 5 sub-testing sets, we perform prediction algorithms on each pair. Since each sub-testing set is different, we get accuracy metrics that depict how the model will react to new and unseen testing data. This gives a much clearer picture than training on only one split.
- Since this splitting is stratified, in each pair of sub-training and sub-testing sets, the class distribution is maintained so as to represent the data as a whole. An example of stratified 5-fold splitting is shown in Figure 2.13. The original training set is split up into 5 unique sub-testing sets and their respective sub-training sets. The numbers represent the instance indices.

The label distribution fractions shows that the sub-training and sub-testing have almost the same class distributions.

```
Training set: [   0    1    2 ... 5395 5396 5399] Testing set: [   6   12   18 ... 5394 5397 5398]
Training set: [   0    2    4 ... 5397 5398 5399] Testing set: [   1    3    7 ... 5388 5390 5396]
Training set: [   0    1    2 ... 5397 5398 5399] Testing set: [  20   25   28 ... 5384 5386 5392]
Training set: [   1    2    3 ... 5396 5397 5398] Testing set: [   0    8    9 ... 5389 5395 5399]
Training set: [   0    1    3 ... 5397 5398 5399] Testing set: [   2    4    5 ... 5381 5391 5393]

Label Distributions:

Training set: [0.96944444 0.03055556]
Testing set:  [0.96851852 0.03148148]
```

Figure 2.13: Example of Stratified 5-Fold Splitting

- For each classifier, the following procedure is followed:
    - An accuracy list is formed to store the best model accuracy for each split.
    - A dictionary of different hyperparameters is formed.
    - RandomizedSearchCV [5] (Randomized Search with Cross Validation) function is used. This function randomly picks combinations from the hyperparameter dictionary and tunes them into the respective classifier. By default, 10 combinations are picked. All these combinations are applied to the classifier for each split and the best combination is returned.
    - Now, the original training set is split into 5 pairs of unique sub-testing and respective sub-training sets.
    - Over each pair, a pipeline function is used. This function combines classifier model with a sampler of choice. Here, in each case, we use SMOTE [6] (Synthetic Minority Over-sampling Technique). This technique synthesizes samples of minority class along the line of each sample's nearest neighbors. This ensures that the synthesized samples are almost similar.
    - First, samples for minority class, i.e., class 1, in each sub-training set are synthesized so as to make the class distribution equal. Then,

the oversampled sub-training set is used to train all 10 classifiers with randomly picked hyperparameter combinations. These models are then tested against their respective sub-testing sets.

- The accuracy of the best combination of hyperparameters in each of the 5 splits is stored in the accuracy list created in the first step. The mean of this list is printed as the overall accuracy, along with the combination of hyperparameters that gives the best accuracy.
- Then, the best combination of hyperparameters is used to test against the last sub-testing set, and the classification report is printed.
- Figure 2.16 shows one of the classification reports. This report has many metrics, but the most significant ones are the recall of Class 1 (sensitivity) and the accuracy. The significance of accuracy is obvious, but the reason why sensitivity is significant as well is that Class 1 has very few instances. Sensitivity is the ratio of correct positive predictions to total number of positives. It is possible that the model has high accuracy above 95 percent, but all the test values are predicted as Class 0. Also, the main focus of this project is to predict and tell in advance if the company will go bankrupt, hence misclassification of Class 1 should be as low as possible, which means sensitivity holds more importance than accuracy.
- Below are the outputs for each classifier, that consist of the overall accuracy after stratified 5-fold cross validation integrated with SMOTE and trained with 10 different hyperparameter combinations randomly picked from the given dictionaries of hyperparameters. Consequently, the best performing combination of hyperparameters is also printed, followed by the classification report of the model with those parameters.

- **LOGISTIC REGRESSION CLASSIFIER**

  Logistic regression takes the predictor variables and pass them through the inverse exponent of a linear function, to obtain a sigmoid curve. Using a threshold value on this curve, the instance is classified.

  Figure 2.14 shows the dictionary of hyperparameters fed to the Randomized Search function. Figure 2.15 shows the overall accuracy across stratified 5-fold cross validation with the best set of hyperparameters, that are also printed. Figure 2.16 shows the classification report of the model with the best performing hyperparameters, tested against the last sub-testing set.

  Hyperparameters [7]: Penalty determines how the algorithm will penalize the model based on error from the previous iteration. C is more or less like the learning rate, or the strength of penalty. Class weight specifies the weight associated with each class while training. Solvers are used for optimization; different solvers use different algorithms.

```
parameters_LR = {"penalty": ['l2',None], 'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
                 'class_weight': ['balanced',None],
                 'solver':['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']}
```

Figure 2.14: Dictionary of Hyperparameters – Logistic Regressor

```
Overall Accuracy:                0.8680555555555557

The best set of parameters:      {'solver': 'newton-cg', 'penalty': 'l2',
'class_weight': None, 'C': 1000}
```

Figure 2.15: Overall Accuracy and Best Set of Hyperparameters – Logistic Regressor

	precision	recall	f1-score	support
Class 0	0.99	0.91	0.95	1046
Class 1	0.22	0.79	0.34	34
accuracy			0.90	1080
macro avg	0.61	0.85	0.64	1080
weighted avg	0.97	0.90	0.93	1080

Figure 2.16: Best Model Classification Report – Logistic Regressor

- **RANDOM FOREST CLASSIFIER**

  Random forest classification is based on randomly selecting features and generating decision trees over certain iterations. The output is nothing but the majority of the outputs of all the decision trees.

  Figure 2.17 shows the dictionary of hyperparameters fed to the Randomized Search function. Figure 2.18 shows the overall accuracy across stratified 5-fold cross validation with the best set of hyperparameters, that are also printed. Figure 2.19 shows the classification report of the model with the best performing hyperparameters, tested against the last sub-testing set.

  Hyperparameters [8]: Max features parameter sets the limit of parameters for each tree. Class weight specifies the weight associated with each class while training. Criterion specifies how the quality of split is calculated. Bootstrap determines whether the samples will be bootstrapped for each tree.

```
parameters_RFC = {'max_features' : ['auto', 'sqrt', 'log2'], 'random_state' : [100],
                  'class_weight' : ['balanced','balanced_subsample'],
                  'criterion' : ['gini', 'entropy'], 'bootstrap' : [True,False]}
```

Figure 2.17: Dictionary of Hyperparameters – Random Forest Classifier

```
Overall Accuracy:                1.0

The best set of parameters:      {'random_state': 100, 'max_features':
'auto', 'criterion': 'entropy', 'class_weight': 'balanced_subsample',
'bootstrap': False}
```

Figure 2.18: Overall Accuracy and Best Set of Hyperparameters – Random Forest Classifier

	precision	recall	f1-score	support
Class 0	0.98	0.98	0.98	1046
Class 1	0.31	0.32	0.31	34
accuracy			0.96	1080
macro avg	0.64	0.65	0.65	1080
weighted avg	0.96	0.96	0.96	1080

Figure 2.19: Best Model Classification Report – Random Forest Classifier

- **XGBOOST (EXTREME GRADIENT BOOSTING) CLASSIFIER**

  XGBoost [9] classification is also based on decision trees, only instead of bagging as in random forest, it follows boosting, i.e., each tree learns from the previous tree based on its error. This learning is a parallel process, making this algorithm much faster.

  Figure 2.20 shows the dictionary of hyperparameters fed to the Randomized Search function. Figure 2.21 shows the overall accuracy across stratified 5-fold cross validation with the best set of hyperparameters, that are also printed. Figure 2.22 shows the classification report of the model with the best performing hyperparameters, tested against the last sub-testing set.

  Hyperparameters [10]: Eta is essentially the learning rate. Eval metric is the evaluation metric for testing, logloss (negative log-likelihood) being used in the case of classification. Max depth specifies the maximum depth of

any tree. Lambda is the L2 regularization term on weights, whereas Alpha is for L1.

```
parameters_XGB = {'eta' : [0.1,0.01,0.001], 'eval_metric': ['logloss'],
                  'max_depth' : [3,6,9], 'lambda' : [1,1.5,2], 'alpha' : [0,0.5,1]}
```

Figure 2.20: Dictionary of Hyperparameters – XGBoost Classifier

```
Overall Accuracy:                0.9997685185185186

The best set of parameters:      {'max_depth': 9, 'lambda': 1.5,
'eval_metric': 'logloss', 'eta': 0.1, 'alpha': 0}
```

Figure 2.21: Overall Accuracy and Best Set of Hyperparameters – XGBoost Classifier

	precision	recall	f1-score	support
Class 0	0.98	0.97	0.98	1046
Class 1	0.33	0.38	0.35	34
accuracy			0.96	1080
macro avg	0.65	0.68	0.66	1080
weighted avg	0.96	0.96	0.96	1080

Figure 2.22: Best Model Classification Report – XGBoost Classifier

- **CATBOOST (CATEGORICAL BOOSTING) CLASSIFIER**

  CatBoost [11] is another decision-tree based classifier following boosting. It employs minimal variance sampling, i.e., sampling with weights at the tree level, making it faster than XGBoost.

  Figure 2.23 shows the dictionary of hyperparameters fed to the Randomized Search function. Figure 2.24 shows the overall accuracy

across stratified 5-fold cross validation with the best set of hyperparameters, that are also printed. Figure 2.25 shows the classification report of the model with the best performing hyperparameters, tested against the last sub-testing set.

Hyperparameters [12]: Eval metric is the evaluation metric for testing. Iterations specifies the maximum number of trees that can be formed. Learning rate, like in other cases, controls the gradient step. Auto class weights specifies the weight associated with each class while training.

```
parameters_CTB = {'eval_metric': ['F1'], 'iterations': [100,500,1000],
                  'learning_rate' : [0.1,0.01,0.001], 'random_seed' : [100],
                  'auto_class_weights' : ['Balanced','SqrtBalanced']}
```

Figure 2.23: Dictionary of Hyperparameters – CatBoost Classifier

```
Overall Accuracy:                0.9979166666666668

The best set of parameters:      {'random_seed': 100, 'learning_rate': 0.1,
'iterations': 1000, 'eval_metric': 'F1', 'auto_class_weights': 'Balanced'}
```

Figure 2.24: Overall Accuracy and Best Set of Hyperparameters – CatBoost Classifier

	precision	recall	f1-score	support
Class 0	0.98	0.97	0.98	1046
Class 1	0.35	0.47	0.40	34
accuracy			0.96	1080
macro avg	0.67	0.72	0.69	1080
weighted avg	0.96	0.96	0.96	1080

Figure 2.25: Best Model Classification Report – CatBoost Classifier

- After training with cross validation and testing the best model on the last sub-testing set, the confusion matrices are plotted as heat maps in Figure 2.26. The horizontal axis contains the predicted classes for the actual classes on the vertical axis. The top row shows the count of test values that actually belong Class 0, while the bottom row shows the actual Class 1 count. Similarly, the left column shows the count of test values predicted as Class 0, while the right column shows the predicted Class 1 count. The heat map helps in better visualizing the differences.

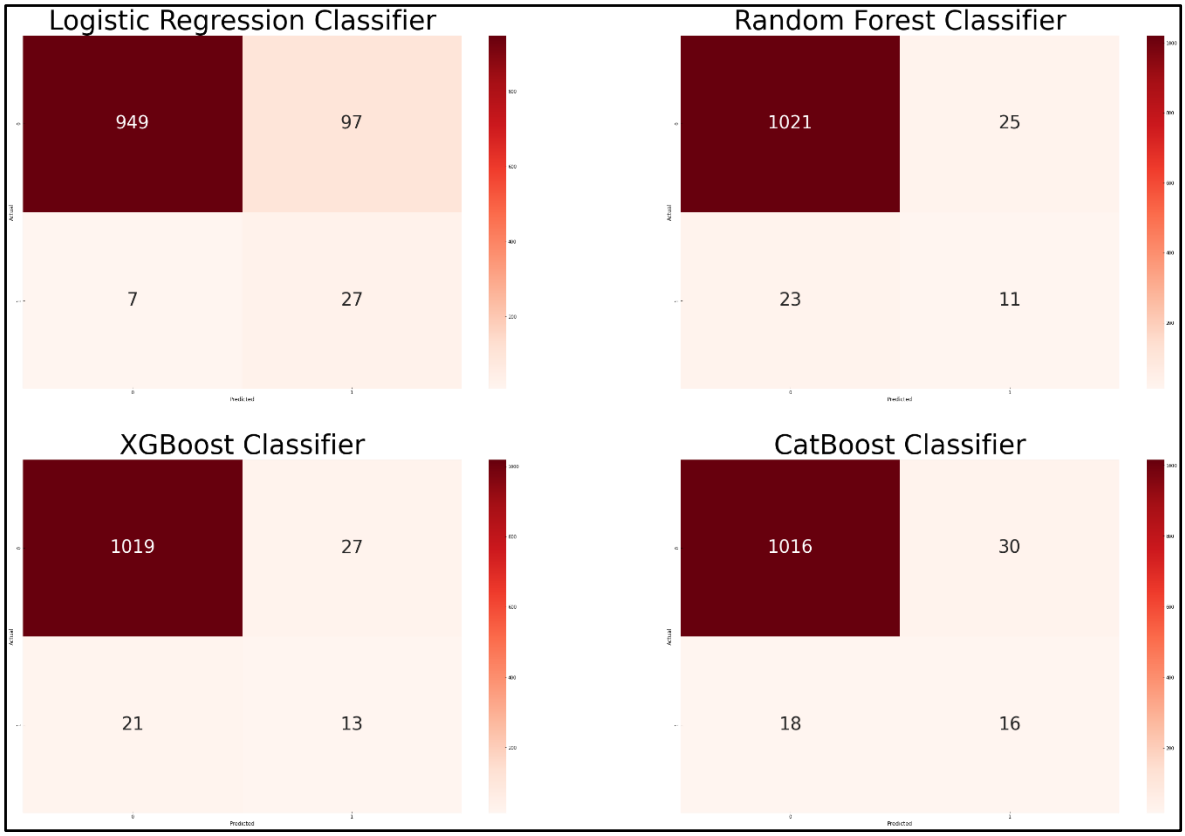


Figure 2.26: Confusion Matrix Heat Maps of Best Models

MODELING WITH FEATURE SELECTION:

- In this second section of modeling, only the 16 features that showed significant difference in their class-wise distribution, as shown in Figure 2.10, will be taken into account. These features are selected from the outlier-removed and logarithmic-transformed dataset so that comparison with the first section results becomes more meaningful. Exactly the same procedure is followed as before for each algorithm. Each algorithm is given the same dictionary of hyperparameters to pick from, and the stratified 5-fold split also gives exactly the same instances in each pair, so as to avoid any variation caused due to differing data or parameters.

- **LOGISTIC REGRESSION CLASSIFIER**

  Figure 2.27 shows the overall accuracy across stratified 5-fold cross validation with the best set of hyperparameters, that are also printed. Figure 2.28 shows the classification report of the model with the best performing hyperparameters, tested against the last sub-testing set.

```
Overall Accuracy:                0.8680555555555557

The best set of parameters:      {'solver': 'newton-cg', 'penalty': 'l2',
'class_weight': None, 'C': 1000}
```

Figure 2.27: Overall Accuracy and Best Set of Hyperparameters – Logistic Regressor

	precision	recall	f1-score	support
Class 0	0.99	0.91	0.95	1046
Class 1	0.22	0.79	0.34	34
accuracy			0.90	1080
macro avg	0.61	0.85	0.64	1080
weighted avg	0.97	0.90	0.93	1080

Figure 2.28: Best Model Classification Report – Logistic Regressor

- **RANDOM FOREST CLASSIFIER**

  Figure 2.29 shows the overall accuracy across stratified 5-fold cross validation with the best set of hyperparameters, that are also printed. Figure 2.30 shows the classification report of the model with the best performing hyperparameters, tested against the last sub-testing set.

```
Overall Accuracy:                1.0

The best set of parameters:      {'random_state': 100, 'max_features':
'auto', 'criterion': 'entropy', 'class_weight': 'balanced_subsample',
'bootstrap': False}
```

Figure 2.29: Overall Accuracy and Best Set of Hyperparameters – Random Forest Classifier

	precision	recall	f1-score	support
Class 0	0.98	0.98	0.98	1046
Class 1	0.31	0.32	0.31	34
accuracy			0.96	1080
macro avg	0.64	0.65	0.65	1080
weighted avg	0.96	0.96	0.96	1080

Figure 2.30: Best Model Classification Report – Random Forest Classifier

- **XGBOOST (EXTREME GRADIENT BOOSTING) CLASSIFIER**

  Figure 2.31 shows the overall accuracy across stratified 5-fold cross validation with the best set of hyperparameters, that are also printed. Figure 2.32 shows the classification report of the model with the best performing hyperparameters, tested against the last sub-testing set.

```
Overall Accuracy:                0.9997685185185186

The best set of parameters:      {'max_depth': 9, 'lambda': 1.5,
'eval_metric': 'logloss', 'eta': 0.1, 'alpha': 0}
```

Figure 2.31: Overall Accuracy and Best Set of Hyperparameters – XGBoost Classifier

	precision	recall	f1-score	support
Class 0	0.98	0.97	0.98	1046
Class 1	0.33	0.38	0.35	34
accuracy			0.96	1080
macro avg	0.65	0.68	0.66	1080
weighted avg	0.96	0.96	0.96	1080

Figure 2.32: Best Model Classification Report – XGBoost Classifier

- **CATBOOST (CATEGORICAL BOOSTING) CLASSIFIER**

  Figure 2.33 shows the overall accuracy across stratified 5-fold cross validation with the best set of hyperparameters, that are also printed. Figure 2.34 shows the classification report of the model with the best performing hyperparameters, tested against the last sub-testing set.

```
Overall Accuracy:                0.9979166666666668

The best set of parameters:      {'random_seed': 100, 'learning_rate': 0.1,
'iterations': 1000, 'eval_metric': 'F1', 'auto_class_weights': 'Balanced'}
```

Figure 2.33: Overall Accuracy and Best Set of Hyperparameters – CatBoost Classifier

	precision	recall	f1-score	support
Class 0	0.98	0.97	0.98	1046
Class 1	0.35	0.47	0.40	34
accuracy			0.96	1080
macro avg	0.67	0.72	0.69	1080
weighted avg	0.96	0.96	0.96	1080

Figure 2.34: Best Model Classification Report – CatBoost Classifier

- After training with cross validation and testing the best model on the last sub-testing set, the confusion matrices are plotted as heat maps in Figure 2.35.

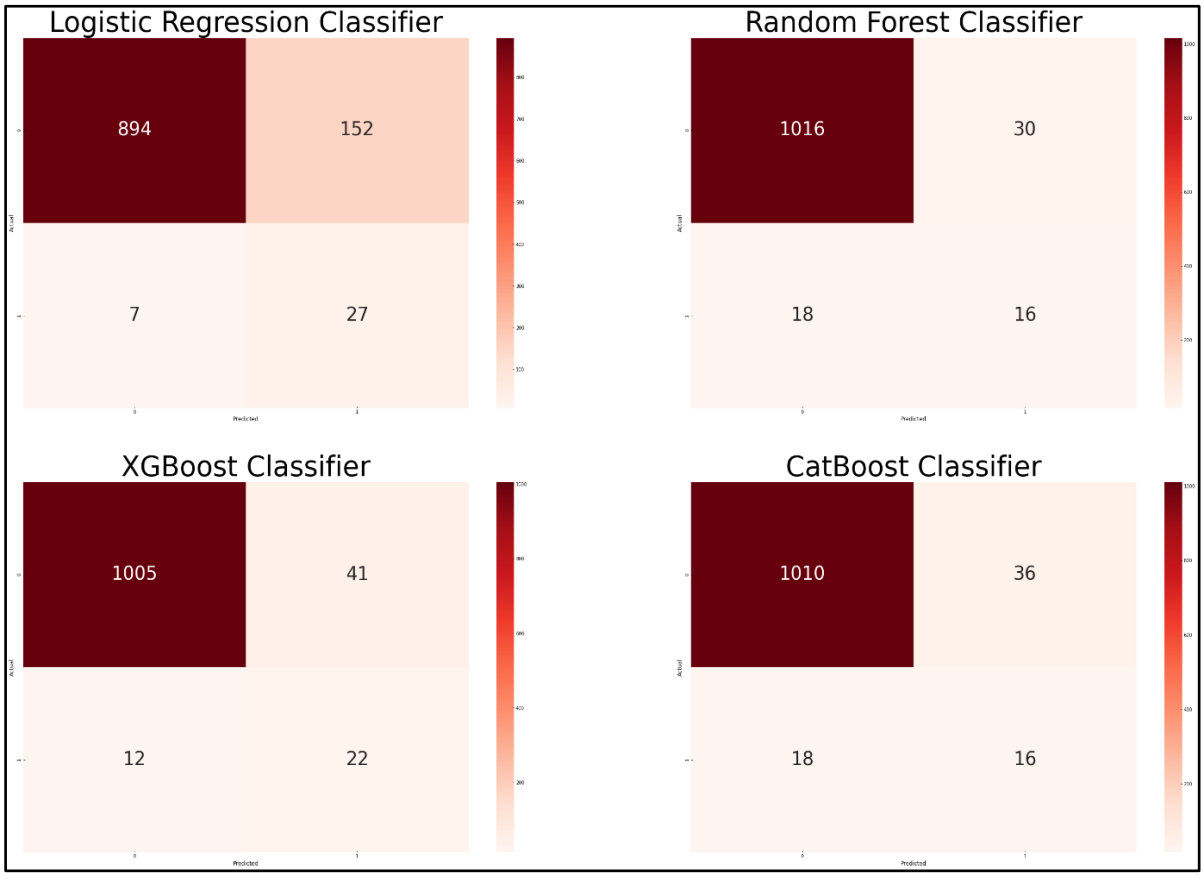


Figure 2.35: Confusion Matrix Heat Maps of Best Models

2.2 ADVANTAGES

This model for bankruptcy prediction has various advantages, including but not limited to the following.

- In a statical report about start-ups, it was found that close to 90% start-ups fail, out of which about 50% fail to recognize they are likely to go bankrupt in the near future and hence continue at the same pace, without any measures to improve their financial ratios, until finally they go bankrupt. This failure to recognize bankruptcy well before time can be overcome if start-ups use this model to predict bankruptcy.
- Not just start-ups, but even large corporates, although rarely, fail to recognize bankruptcy before obvious signs show. Hence, this model can be used by both start-ups and well-established companies alike.
- Since there is a high class-imbalance, we implement the use of SMOTE as well as stratified 5-fold cross validation splitting. SMOTE oversamples the minority class for training for its better representation, while cross validation reduces chances of error related to unseen or new test data. We also perform logarithmic transformation on highly skewed variables to further improve accuracy.
- Since there are 95 predictor variables and only binary classification, a model with the entire dataset runs the risk of overfitting. Thus, we also use feature selection based on individual feature analysis, in order to use the variables that show a significant difference in the distribution of values for either class.
- Instead of sticking to just one combination of hyperparameters, for each algorithm, we use a dictionary of hyperparameters such that 10 different combinations of them are tested on each split, and only the combination that gives the best result in the majority of splits is used for final testing.
- Most models give importance to accuracy alone, but since misclassification of Class 1 is the most undesirable for this type of classification, we give a higher priority to sensitivity. For this cause, we

have devised a metric that gives more weightage to sensitivity while also considering accuracy (refer Section 3.1).

2.3 CHALLENGES FACED

In the implementation of this program, a few challenges were faced as well, which took a lot of brainstorming to get around.

- The first challenge was that our dataset was highly class-imbalanced. Even upon using SMOTE and stratified 5-fold cross validation splitting, most of the algorithms were giving low sensitivity values even on the best set of hyperparameters. This situation was slightly improved after manual feature selection.
- Another challenge was that most of the variables seemed insignificant, since their class-wise distribution was more or less the same. To tackle this, we manually selected features by visualizing class-wise boxplots for each feature. Naturally, the results were mostly better after feature selection.
- We observed that a lot of features showed a very high skew to either side. This skew, in some cases, can affect accuracy of the model. After much research, we decided to perform logarithmic transformation on the highly skewed features to give them a shape that better resembles a bell.
- Another challenge was that the models giving a better accuracy figure would give low sensitivity. We needed to compare the models on both these metrics. Thus, we devised a new metric that gives higher weightage to sensitivity, as desired, while also taking accuracy into picture.

2.4 CODE

The python notebook for the implementation of our code can be found in the below Google drive folder, by the name ‘Bankruptcy Prediction.ipynb’. Note that this notebook can be opened using either Jupyter Notebook after downloading onto the device, or Google Colaboratory directly from Google Drive.

https://drive.google.com/drive/folders/15UNhYksZCxImeGy5cqN2cFQSBlN-czHm?usp=sharing

The CSV file for the dataset used can also be found in the same folder by the name ‘data.csv’, or in UCI Machine Learning Repository [2].

CHAPTER 3

MAIN RESULTS AND INFERENCES

3.1 MAIN RESULTS

The project compares and contrasts the performance of the model with and without feature selection. As a result of the feature selection, the total number of features was reduced from 95 to 16 (excluding the target variable). The features are chosen so that the distributions of class ‘0' and class ‘1' are significantly different. The “Operating Expense Rate”, “Total Asset Growth Rate”, “Tax Rate(A)” and “Cash Turnover Rate” are among the features that have been chosen.

The classification task of predicting whether a company will go bankrupt or not is modeled using four machine learning classification algorithms with SMOTE (due to class imbalance):

- Logistic Regression
- Random Forest Classification
- XGBoost Classification
- CatBoost Classification

The hyperparameter tuning is performed effectively using “RandomizedSearchCV” which trains the model by iterating over the list of hyperparameter values to be changed and outputs the best set of parameters and accuracy for the best-fit model.

MODELING WITH ALL 95 FEATURES:

The accuracy value of all four classifiers is shown in the Figure 3.1 below. The accuracy does not convey much information as it's the same for all classifiers except logistic regression.

```
Logistic Regression Classifier:          0.9037037037037037
Random Forest Classifier:                0.9555555555555556
XGBoost Classifier:                      0.9555555555555556
CatBoost Classifier:                     0.9555555555555556
```

Figure 3.1: Accuracy Metrics

The Figure 3.2 below shows the sensitivity values for the same. The sensitivity of the classifiers is a crucial parameter to pay attention to since it determines the true positive rate, or how often the classifier accurately predicts the outcome when the company goes bankrupt. Figure 3.2 clearly indicates that, though logistic regression's accuracy is lower than the others, its sensitivity is significantly higher. When a company is on the verge of bankruptcy, logistic regression can better forecast that it will go bankrupt.

```
Logistic Regression Classifier:          0.7941176470588235
Random Forest Classifier:                0.3235294117647059
XGBoost Classifier:                      0.38235294117647056
CatBoost Classifier:                     0.47058823529411764
```

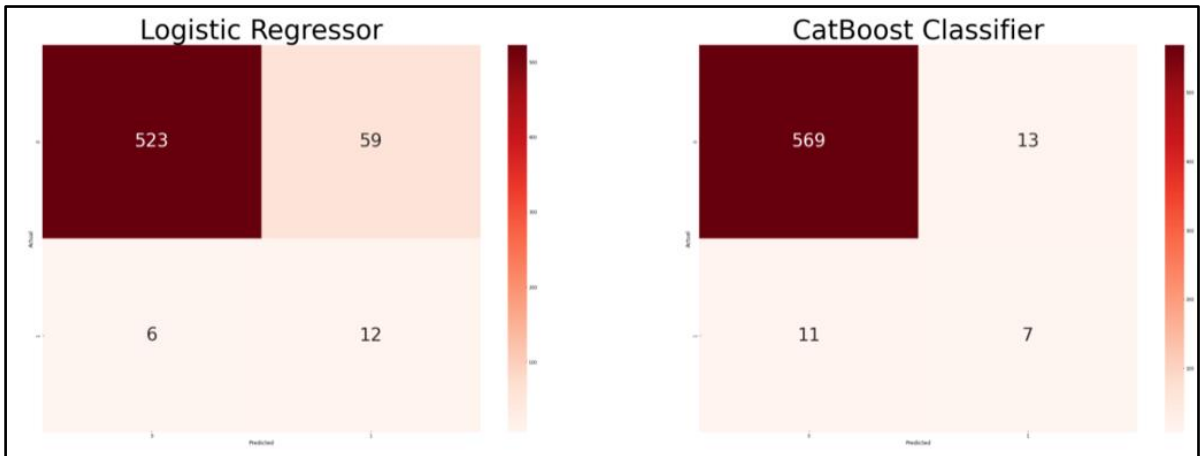
Figure 3.2: Sensitivity Metrics

A new weighted metric was designed such that it emphasizes on sensitivity and not compromise on the accuracy at the same time. The formula for the same is **Sensitivity^2 x Accuracy**. The values of the weighted metric are shown in the Figure 3.3 below. This clearly demonstrates logistic regression's dominance, as its value is substantially higher than the others. The CatBoost classifier comes after that.

Logistic Regression Classifier:	0.5698961937716263
Random Forest Classifier:	0.10001922337562477
XGBoost Classifier:	0.1396962706651288
CatBoost Classifier:	0.2116109188773549

Figure 3.3: Weighted Metrics

Because their weighted metrics were better than the rest, the model's performance on test data was evaluated using Logistic Regression and CatBoost classifier. The confusion matrix and the report for the final test data of 600 observations is seen in Figure 3.4. CatBoost classifier has a greater accuracy than Logistic regression, however misclassification is more often for class '1'. There exists a tradeoff between accuracy and sensitivity.



Logistic Regression report:

	precision	recall	f1-score	support
Class 0	0.99	0.90	0.94	582
Class 1	0.17	0.67	0.27	18
accuracy			0.89	600
macro avg	0.58	0.78	0.61	600
weighted avg	0.96	0.89	0.92	600

CatBoost report:

	precision	recall	f1-score	support
Class 0	0.98	0.98	0.98	582
Class 1	0.35	0.39	0.37	18
accuracy			0.96	600
macro avg	0.67	0.68	0.67	600
weighted avg	0.96	0.96	0.96	600

Figure 3.4: Confusion Matrix Heat Maps and Classification Reports of Best Performing Models After Final Testing

MODELING WITH FEATURE SELECTION:

After feature selection (16 features), a similar trend is noticed in the accuracy of the classifiers with logistic regression's accuracy declining slightly (Figure 3.5). When the sensitivity is analyzed for the same test data (Figure 3.6), it has improved significantly for all the classifiers, more so in the case of XGBoost.

```
Logistic Regression Classifier:          0.8527777777777777
Random Forest Classifier:                0.9555555555555556
XGBoost Classifier:                      0.950925925925926
CatBoost Classifier:                     0.95
```

Figure 3.5: Accuracy Metrics

```
Logistic Regression Classifier:          0.7941176470588235
Random Forest Classifier:                0.47058823529411764
XGBoost Classifier:                      0.6470588235294118
CatBoost Classifier:                     0.47058823529411764
```

Figure 3.6: Sensitivity Metrics

Figure 3.7 depicts the weighted metric of the classifiers. With and without feature selection, the weighted measure of logistic regression is nearly same. Since the performance of XGBoost is substantially better, these two models are utilized to further analyze the performance on the test data.

```
Logistic Regression Classifier:          0.537781141868512
Random Forest Classifier:                0.2116109188773549
XGBoost Classifier:                      0.3981385364603358
CatBoost Classifier:                     0.21038062283737025
```

Figure 3.7: Weighted Metrics

The confusion matrix and the report for the final test data of 600 observations is seen in Figure 3.8. After feature selection, the sensitivity of XGBoost has substantially increased (but not as compared to logistic regression) without a compromise in the accuracy. Even after feature selection, there is a trade-off between accuracy and sensitivity.

Logistic Regression report:

	precision	recall	f1-score	support
Class 0	0.99	0.83	0.90	582
Class 1	0.12	0.78	0.21	18
accuracy			0.83	600
macro avg	0.56	0.80	0.56	600
weighted avg	0.97	0.83	0.88	600

XGBoost report:

	precision	recall	f1-score	support
Class 0	0.98	0.96	0.97	582
Class 1	0.26	0.50	0.34	18
accuracy			0.94	600
macro avg	0.62	0.73	0.65	600
weighted avg	0.96	0.94	0.95	600

Figure 3.8: Confusion Matrix Heat Maps and Classification Reports of Best Performing Models After Final Testing

3.2 INFERENCES

Certain inferences can be drawn from the obtained outputs.

- The dataset has 6819 observations and 96 features, and the distribution plot of the ‘Bankrupt?' feature indicates that the dataset is highly class- imbalanced.

- Boxplot is used to infer the presence of outliers from the data distribution of the full dataset. An approach for outlier removal is included, ensuring that outliers are removed in a minimal but effective manner while maintaining the class distribution ratio. After removing the outliers, the dataset shrinks from 6819 to 6000 instances.

- The model is then trained using Stratified cross validation, which ensures that when the data is divided into ‘5' folds, the sampling occurs in such a way that the proportion of class ‘0' and ‘1' in each fold is similar to the actual class distribution in the entire dataset, as shown in the Figure 2.13. The same is true of the training and testing sample's class proportions.

- When feature selection is performed, it is inferred that most of the features follow the same distribution irrespective of the classes as shown in Figure 2.9. This results in the presence of almost duplicated features in the data, obviating the requirement for them. A large number of features also causes overfitting. Only those features that are notably distinct (as in Figure 2.10) for class ‘0’ and class ‘1’ are extracted and trained.

- When the histogram of all features was visualized, it was inferred that many features where skewed in either directions, this called the need for performing logarithmic transformation on those features whose skew is more than 0.5 or less than -0.5 to ensure that they follow a normal distribution.

- In the case of modeling with all the features, through the analysis of weighted metric, it is found that **Logistic Regression and CatBoost** perform better.

Though the accuracy of Logistic Regression is less than CatBoost, the sensitivity value is significantly higher and it is able to predict the true positive better. After hyperparameter tuning the best set of parameters for Logistic Regression and CatBoost respectively are shown in Figures 2.14 and 2.23.

- When the model was trained with reduced number of features, the performance of **Logistic Regression** almost remained the same. Though there was a compromise in the accuracy and increase in false positives, it was able to classify the true positives more accurately which is seen by the increase in sensitivity of class '1' from 0.67 (Figure 3.4) to 0.78 (Figure 3.8) on the test data. The sensitivity of **XGBoost** had improved further after feature selection without any compromise in the accuracy. The best set of parameters after hyperparameter tuning for Logistic Regression and XGBoost is shown in Figures 2.27 and 2.31 respectively.

- Finally, we infer that feature selection does help our model to an extent, as the sensitivity of every model is improved, maintaining the accuracy as well.

- The best performing model, when given higher weightage to sensitivity, turns out to be the **Logistic Regressor** after feature selection (Accuracy = 83%, Sensitivity = 78%), followed by **Logistic Regressor** without feature selection (Accuracy = 89%, Sensitivity = 67%). Even though the decision-tree based models observe an increased sensitivity after feature selection, their sensitivity performance is much lower than the Logistic Regressors.

CHAPTER 4

CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK

4.1 CONCLUSION

Bankruptcy prediction is one crucial aspect to be monitored in an organization and the need for adopting such models came into light after The Great Recession in 2008. Because a company's bankruptcy might have a direct impact on the financial market, a better prediction model based on many criteria is always being researched, as it will also affect the profitability of lending institutions. In this project, we have implemented supervised machine learning model which can classify whether a company will go bankrupt or not using several classification algorithms with a comparative analysis of each model to determine which is better suited. The sensitivity is a crucial performance statistic to track since it predicts that the company will go bankrupt if it actually does. The model developed should be such that it should not report “Not bankrupt” when the company is actually bankrupt. A new weighted metric is devised for selecting and testing the better model. It is inferred that a tradeoff exists between accuracy and sensitivity in most cases due to class imbalance. Even after feature selection, the logistic regression is a stronger model in terms of sensitivity, since it can predict the occurrence of bankruptcy better than the others, but there is a sacrifice in accuracy. When compared to others, XGBoost performs substantially better in terms of accuracy, however misclassification of class ‘1' is the main difficulty.

4.2 RECOMMENDATIONS FOR FUTURE WORK

The model may be deployed as a GUI or a website (using Stream-lit/ flask & Heroku) to make it more user-friendly, allowing the user to input feature values and determine whether or not the company will go bankrupt in the coming years, as well as the model's reliability.

Only those features that are notably distinct for both classes are considered in the project's feature selection. To further increase the model's performance, a different approach of feature selection can be incorporated, along with removal of highly correlated features. Instead of SMOTE as an oversampling technique, ADASYN can also be used.

To classify whether the company will go bankrupt or not, a neural network with a SoftMax function in the last layer can be employed to test the model's performance. Since it’s a numerical dataset, the architecture of a deep neural network can assist in uncovering more information than conventional machine learning techniques, potentially improving performance.

Along with the above set of variables, additional data such as the country's economic situation, stock prices, and corporate sentiment (as measured by tweets, news, and customer analytics) can be utilized to create a more robust model that accounts for all of the elements that affect an organization.

REFERENCES

[1] imbalanced-learn · PyPI

[2] UCI Machine Learning Repository: Taiwanese Bankruptcy Prediction Data Set

[3] Log Transformation: Purpose and Interpretation | by Kyaw Saw Htoon | Medium

[4] 5 Reasons why you should use Cross-Validation in your Data Science Projects | by Dima Shulga | Towards Data Science

[5] How to find optimal parameters using RandomizedSearchCV for Regression? (dezyre.com)

[6] SMOTE | Overcoming Class Imbalance Problem Using SMOTE (analyticsvidhya.com)

[7] sklearn.linear_model.LogisticRegression — scikit-learn 0.24.2 documentation

[8] sklearn.ensemble.RandomForestClassifier — scikit-learn 0.24.2 documentation

[9] XGBoost Algorithm | XGBoost In Machine Learning (analyticsvidhya.com)

[10] XGBoost Parameters — xgboost 1.5.0-dev documentation

[11] How training is performed - CatBoost. Documentation

[12] Python package training parameters - CatBoost. Documentation