Company Bankruptcy Prediction

High Dimensional Data Analysis: Classification

IT2022E - Applied Statistics and Experimental Design - Class 131680 - Sem. 20212

Nguyen Hoang Phuc
20204923

Hoang Tran Nhat Minh
20204883

Nguyen Ngoc Dung
20204905

Nguyen Hai Long
20204920

Ho Minh Khoi
20204917

I. Abstract

Bankruptcy Prediction is essential for public companies. The rationale for developing and predicting the financial distress of a company is to develop a predictive model used to forecast the financial condition of a company by combining several econometric variables of interest to the researcher. Due to the complex nature of finance, all data were high in dimensions and require certain techniques to process. In this report, we will try to predict bankruptcy through some public data online, using data analysis, dimension reduction techniques, feature selection techniques and Machine Learning models. We conclude that the best performing model we made was an ensemble of Random Forest Classifier, AdaBoost and Gradient Boosting, with Halving Grid Search Cross Validation. It managed to achieve a prediction accuracy of 61.69%. This study can be further worked into by searching for more novel dimension reduction techniques, tuning more on hyperparameters and models, or making validation against other data sets to further improve accuracy.

II. Acknowledgements

We would like to express our gratitude to Assoc. Prof. Nguyen Linh Giang for giving us an opportunity to work together on a practical problem. Without some of his advices on how to approach the problem and how to do experimental design, the research would not be as deep nor as thorough as it could be. We would like to express the same gratitude to Assoc. Prof. Than Quang Khoat as his Machine Learning course's students. The knowledge we gained from the course greatly helped us in finding the right techniques for our problem.

III. Introduction

Bankruptcy or business failure can have a negative impact both on the enterprise itself and the global economy. Some of the most famous cases include Lehman Brothers' collapse, one of the events that marked the financial crisis of 2008, or recently, China's Evergrande debt crisis, with $300 billion in liabilities at the end of 2021.

In short, bankruptcy prediction is a very important task for many related financial institutions. In general, the aim is to predict the likelihood that a firm may go bankrupt. Financial institutions are in need of effective prediction models in order to make appropriate lending decisions.

More particularly, we are currently facing the threat of an economic recession. Inflation rate has gone up recently and new COVID-19 variants are spreading and mutating fast. The Russian - Ukrainian war is also a factor that affects a lot on the supply chain, causing many problems such as food and energy crises. Thus, every firm is becoming more and more vulnerable, and the work of predicting business failure is in high demand.

In the literature, many techniques have been employed to develop bankruptcy prediction models, including statistical and machine learning techniques (Balcaen and Ooghe, 2006, Kumar and Ravi, 2007, Lin et al., 2012, Verikas et al., 2010) with machine learning techniques shown to outperform statistical techniques.

To dive into the analysis of bankruptcy, we will apply various techniques on the UCI Taiwanese Bankruptcy Prediction Data Set. The data were collected from the Taiwan Economic Journal for the years 1999 to 2009 [1]. Bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange. The dataset has been used in many studies, namely Liang, D. et al. [2] study on the importance of financial ratios and corporate governance indicators in bankruptcy prediction. This dataset contains 6819 instances and 96 attributes, including the class label, and brings up **two problems**, which we are going to provide some solutions to:

- **Dealing with imbalanced datasets:** Due to the number of bankruptcy cases being relatively small compared to the whole dataset (220 in 6819).
- **Handling high-dimensional data:** High number of attributes makes it complicated to apply statistical or machine learning methods. Thus, we would dig into some methods to reduce the complexity of the dataset to find important features.

The rest of this paper is organized as follows. Section 3 overviews literature related to our subject. Sections 4 and 5 present the research methodology and experimental results respectively. Finally, in Section 6 some conclusions are offered.

IV. Literature review

1. Terminologies & Techniques

a. Bankruptcy

In academic literature and bankruptcy prediction studies, four related terms have commonly been used: **default**, **failure**, **insolvency**, and **bankruptcy**. Whilst these terms have often been used interchangeably, they have distinct formal meanings (Jarbøl, A. and Bevort, A.)[3]. Thus, we need to choose a definition to ensure homogeneity. In comparison to the other terms, **bankruptcy** rather refers to a **legal process** imposed by a court order so that the debtor get relief from some or all of their debts.

b. Corporate Governance Indicators

The definition of “Corporate governance” describes the processes, structures, and mechanisms that influence the control and direction of corporations'. [4] Many corporate governance indicators (CGIs) defined (see Appendix) have been used for solving bankruptcy or financial crisis

problems. However, not all the CGIs used for predicting bankruptcy in related works are the same. In other words, different categories of CGIs have been considered in different studies. We would discuss more details in the next part of our literature.

c. Handling imbalanced data

One approach to addressing the problem of class imbalance is to randomly resample the training dataset. There are two main approaches to random resampling for imbalanced classification: **Undersampling** and **Oversampling**.

- **Undersampling: Random undersampling** randomly selects examples from the majority class to delete from the training dataset. The main issue with random undersampling is that we run the risk that our classification models will not perform as accurate as we would like to since there is a great deal of **information loss.**

Figure 1. Comparison between undersampling and oversampling

- Oversampling: Random oversampling involves randomly duplicating examples from the minority class and adding them to the training dataset.

- Oversampling: SMOTE (*Synthetic Minority Oversampling TEchnique*) SMOTE works by selecting examples that are close in the feature space (*k* nearest neighbors), drawing a line between the examples in the feature space and drawing a new sample at a point along that line. Chawla, N. et al.[5] concluded that “The combination of SMOTE and under-sampling performs better than plain under-sampling.”

Multiple variations of SMOTE (as below) can be used to avoid generation of noise.

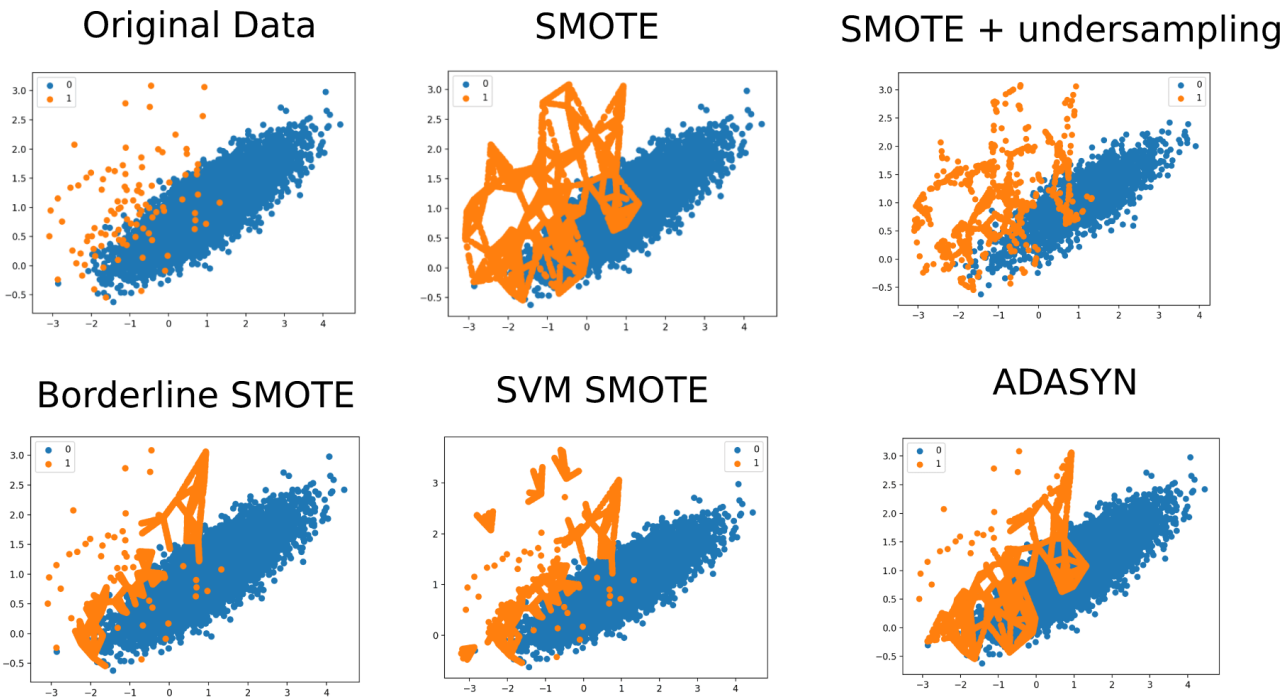

Figure 2. Variations of SVM and example results (plotted by Machine Learning Mastery[6])

d. Handling high-dimensional data

One of the most common ways of dealing with high-dimensional data is **dimensional reduction**, which can be classified into **feature selection** and **feature projection.**

- Feature selection: select a subset of relevant attributes for modelling. Some of feature selection methods include **subset selection**, where search methods such as *genetic algorithm* or *simulated annealing* are used and then a model is applied to the subset to evaluate the result, or **correlation feature selection**, based on the following hypothesis: "Good feature subsets contain features highly correlated with the classification, yet uncorrelated to each other" (Hall, M.)[7]. **Random forest** is also a possible approach.

- **Feature projection: PCA** (*Principal Component Analysis*) tries to find a new basis such that the majority of information lies in a few axes while the minority lies in the rest.

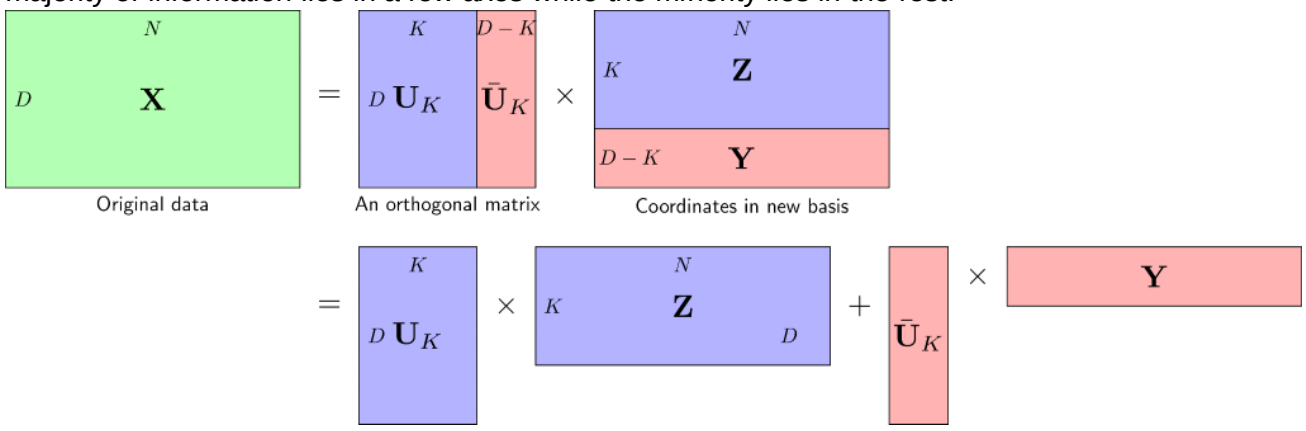

Figure 3. How PCA works (source: Machinelearningcoban[8])

Procedure of PCA can be described in the following steps and in **Figure 4**:

Step 1: Find mean vector of the data $$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

Step 2: Subtract mean, or shifting data to origin $$\hat{\mathbf{x}}_n = \mathbf{x}_n - \bar{\mathbf{x}}$$

Step 3: Calculate covariance matrix $$\mathbf{S} = \frac{1}{N}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$$

Step 4: Find eigenvalues and eigenvectors of S, note that eigenvectors are now **orthogonal**
Step 5: Select K eigenvectors with highest eigenvalues
Step 6: Project the data to selected eigenvectors

Step 7: Obtain new dataset on new space $$\mathbf{x} \approx \mathbf{U}_K\mathbf{Z} + \bar{\mathbf{x}}$$

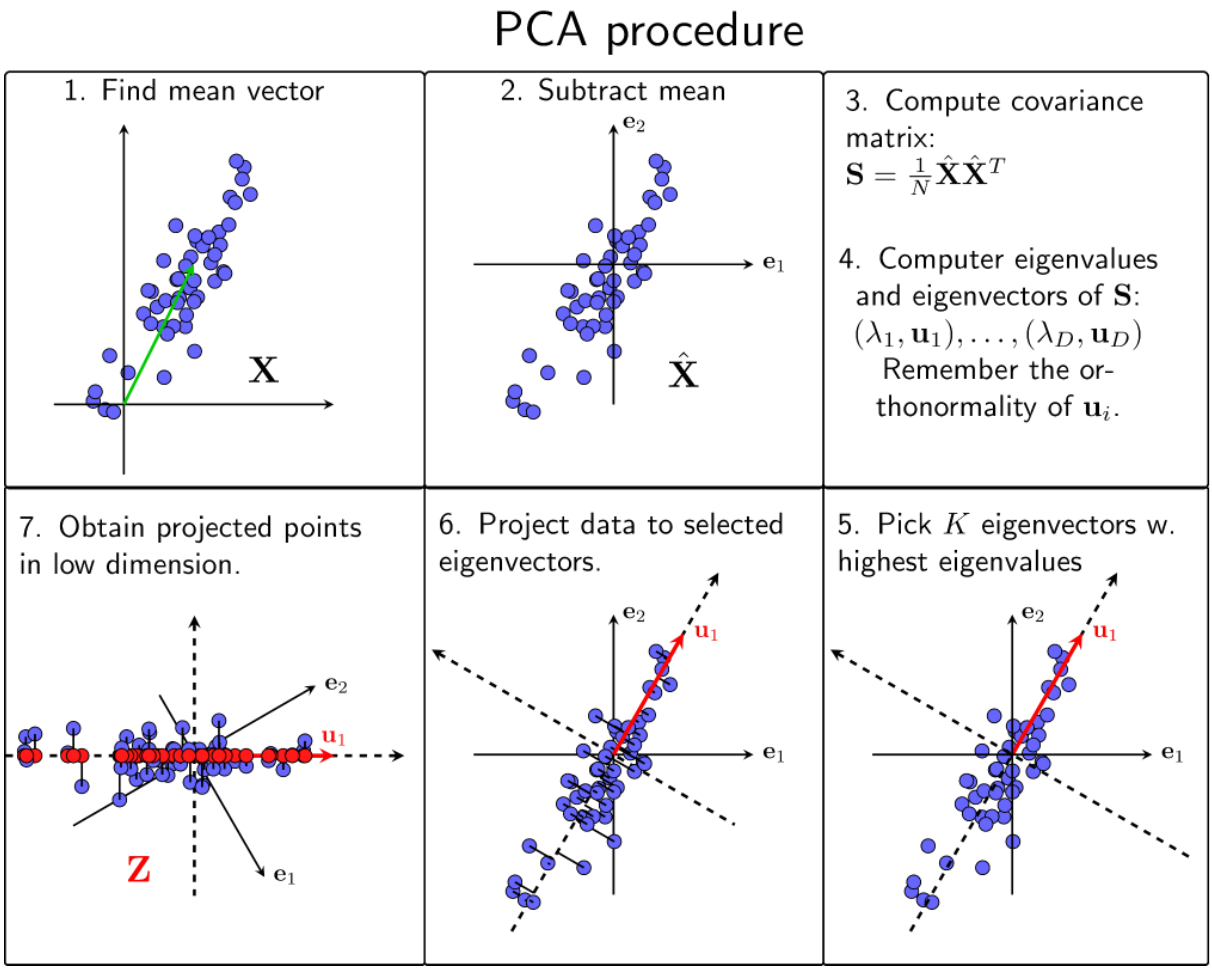

Figure 4: PCA procedure (source: Machinelearningcoban)

- **Feature projection: MDA** (*Multiple Discriminant Analysis*) is also useful in handling high dimensional data, and in particular, bankruptcy prediction. Two models widely used are **Altman Z-**

score and **Ohlson O-score,** both of which, after using MDA, deduced simple formulas using very few CGIs.
The original Z-score formula was as follows:

$$Z = 1.2A + 1.4B + 3.3C + 0.6D + 1.0D, \text{ where:}$$

A = working capital / total assets
B = retained earnings / total assets
C = earnings before interest and tax / total assets
D = market value of equity / total liabilities
E = sales / total assets

From the calculated Z-score, firms can be classified into:

Z > 2.99 – **safe** zone; 1.81 < Z < 2.99 – **grey** zone; Z < 1.81 – **distress** zone

The Ohlson O-score is the result of a 9-factor linear combination of CGIs from consecutives financial disclosure statements:

$$T = -1.32 - 0.407\log(TA_t/GNP) + 6.03\frac{TL_t}{TA_t} - 1.43\frac{WC_t}{TA_t} + 0.0757\frac{CL_t}{CA_t}$$

$$- 1.72X - 2.37\frac{NI_t}{TA_t} - 1.83\frac{FFO_t}{TL_t} + 0.285Y - 0.521\frac{NI_t - NI_{t-1}}{|NI_t| + |NI_{t-1}|}$$

where TA = total assets
GNP = gross national product price index level (in USD, 1968 = 100)
TL = total liabilities
WC = working capital
CL = current liabilities
CA = current assets
X = 1 if TL > TA, 0 otherwise
NI = net income
FFO = funds from operations
Y = 1 if a net loss for the last two years, 0 otherwise.

Probability of bankruptcy appears as $P = \frac{e^T}{1+e^T}$, a sigmoid function.

For the O-score, any results larger than 0.5 suggest that the firm will default within two years.

2. Previous studies

Bankruptcy prediction has been a subject of formal analysis since at least 1932, when FitzPatrick published a study of 20 pairs of firms, one failed and one surviving, matched by date, size and industry, in The Certified Public Accountant. He did not perform statistical analysis as is now common, but he thoughtfully interpreted the ratios and trends in the ratios. His interpretation was effectively a complex, multiple variable analysis.
In 1967, William Beaver applied t-tests to evaluate the importance of individual accounting ratios within a similar pair-matched sample.
In 1968, in the first formal multiple variable analysis, Edward I. Altman applied multiple

discriminant analysis within a pair-matched sample.[9] One of the most prominent early models of bankruptcy prediction is the Altman Z-score, which is still applied today.
In 1980, James Ohlson applied logistic regression in a much larger sample that did not involve pair-matching.[10]

The latest research within the field of Bankruptcy and Insolvency Prediction compares various differing approaches, modelling techniques, and individual models to ascertain whether any one technique is superior to its counterparts.
Jackson and Wood (2013) is one of many reviews of the literature to date, and included an empirical evaluation of 15 popular models from the existing literature. These models range from the univariate models of Beaver through the multidimensional models of Altman and Ohlson, and continuing to more recent techniques which include option valuation approaches. They find that models based on market data - such as an option valuation approach - outperform those earlier models which rely heavily on accounting numbers.[11]
Zhang, Wang, and Ji (2013) proposed a novel rule-based system to solve bankruptcy prediction problem. The whole procedure consists of the following four stages: first, sequential forward selection was used to extract the most important features; second, a rule-based model was chosen to fit the given dataset since it can present physical meaning; third, a genetic ant colony algorithm (GACA) was introduced; the fitness scaling strategy and the chaotic operator were incorporated with GACA, forming a new algorithm—fitness-scaling chaotic GACA (FSCGACA), which was used to seek the optimal parameters of the rule-based model; and finally, the stratified K-fold cross-validation technique was used to enhance the generalization of the model.[12]

V. Methodology

1. Data Collection

Most of the CGIs can be found or else, calculated through financial reports, which are publicly open on the companies' websites. The data can be found in the following types: **balance, income, equity changes** and **cash flows**.

	Note	30 June 2021 (Unaudited) RMB million	31 December 2020 (Audited) RMB million
ASSETS			
Non-current assets			
Property, plant and equipment	7	**81,027**	75,731
Right-of-use assets	7	**19,617**	18,561
Investment properties	7	**155,407**	165,850
Goodwill	7	**8,185**	7,822
Intangible assets	7	**13,516**	10,696
Trade and other receivables	9	**6,955**	7,249
Prepayments	10	**2,334**	2,461
Investments accounted for using the equity method	11	**115,736**	92,270
Financial assets at fair value through other comprehensive income	12	**3,562**	1,412
Financial assets at fair value through profit or loss	13	**9,866**	8,230
Deferred income tax assets		**8,777**	5,943
		424,982	396,225
Current assets			
Inventories		**357**	358
Properties under development	8	**1,278,965**	1,257,908
Completed properties held for sale	8	**144,514**	148,473
Trade and other receivables	9	**175,946**	141,706
Contract acquisition costs		**4,198**	5,190
Prepayments	10	**164,684**	151,026
Income tax recoverable		**20,698**	16,334
Financial assets at fair value through profit or loss	13	**1,604**	3,195
Restricted cash	14	**74,855**	21,992
Cash and cash equivalents	14	**86,772**	158,752
		1,952,593	1,904,934
Total assets		**2,377,575**	2,301,159

Figure 5: An example of an interim report. (Source: Evergrande[13])

2. Exploratory Data Analysis

2.1. Check data information

```
bank_data.info()
```

```
RangeIndex: 6819 entries, 0 to 6818
Data columns (total 96 columns)
```

```
dtypes: float64(93), int64(3)
memory usage: 5.0 MB
```

Through data.info(), we observed that we have a majority of "float64" data. The categorical data is distinguished as binary 1 and 0, thus stored as "int64". So, we first analyze three features which datatype are "int64".

2.2. Feature overview

```
strange_features = bank_data.dtypes[bank_data.dtypes == "int64"].index

numeric_features = bank_data.dtypes[bank_data.dtypes == "float64"].index



print(bank_data[strange_features].columns.tolist())

print(len(strange_features))

print(len(numeric_features))
```

```
['Bankrupt?', ' Liability-Assets Flag', ' Net Income Flag']
3
93
```

We have here 3 “strange” features, one of which is the objective. Therefore, we will take a look into the other 2 features.

a. Liability-Assets Flag compared to Bankrupt

- Explanation of `Liability-Assets`:

  The `Liability-Assets` flag denotes the status of an organization, where if the `total liability` exceeds `total assets`, the flagged value will be 1, else the value is 0. A majority of times, organizations/company's assets are more than their liabilities.

- Brief:

  A small portion of organization suffers bankruptcy, although possessing more assets than their liabilities.

- Exploring flag:

```
print(bank_data[[" Liability-Assets Flag", "Bankrupt?"]].value_counts())

print("======================")

plt.figure(figsize=(10, 7))
```

```
sns.countplot(x=" Liability-Assets Flag", hue="Bankrupt?", data=bank_data)
```

```
 Liability-Assets Flag  Bankrupt?

0                       0          6597
                        1           214
1                       1             6
                        0             2

dtype: int64
======================
```

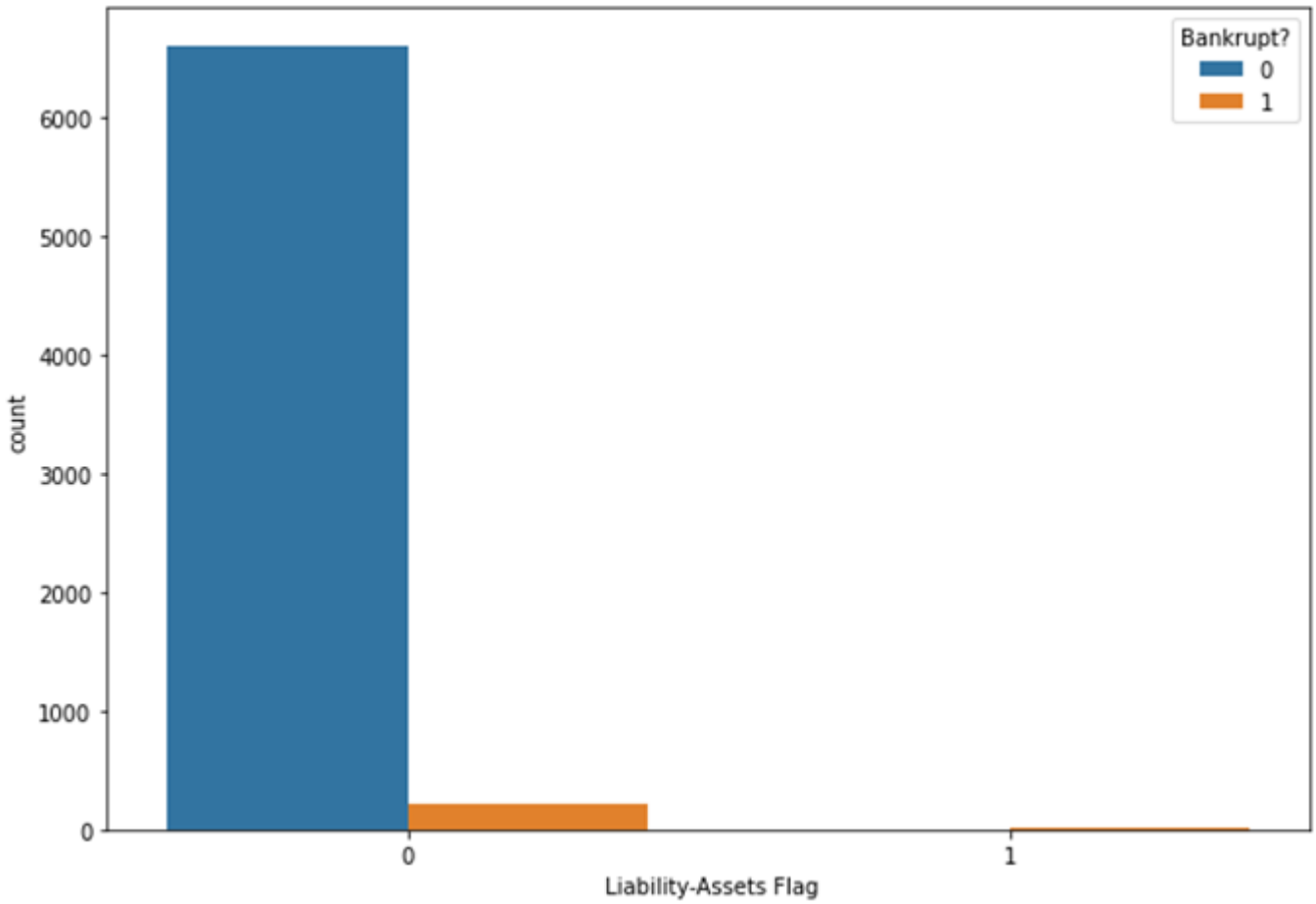

Figure 6: Liability-Assets Flag compared to Bankrupt

b. Net Income Flag compared to Bankrupt

- Explanation of `Net Income`:

The `Net Income` flag denotes the `status of an organization's income` in the last two years, where if the net income is negative for the past two years, the flagged value will be 1, else the value is 0. We observe that all the records have been exhibiting a loss for the past two years.

- Brief:

Many organizations that have suffered losses for the past two years have stabilized their business, thus avoiding bankruptcy.

- Exploring Flag

```
print(bank_data[[" Net Income Flag", "Bankrupt?"]].value_counts())

print("======================")

plt.figure(figsize=(10, 7))

sns.countplot(x=" Net Income Flag", hue="Bankrupt?", data=bank_data)
```

```
Net Income Flag  Bankrupt?
1                0           6599
                 1           220

dtype: int64
======================
```

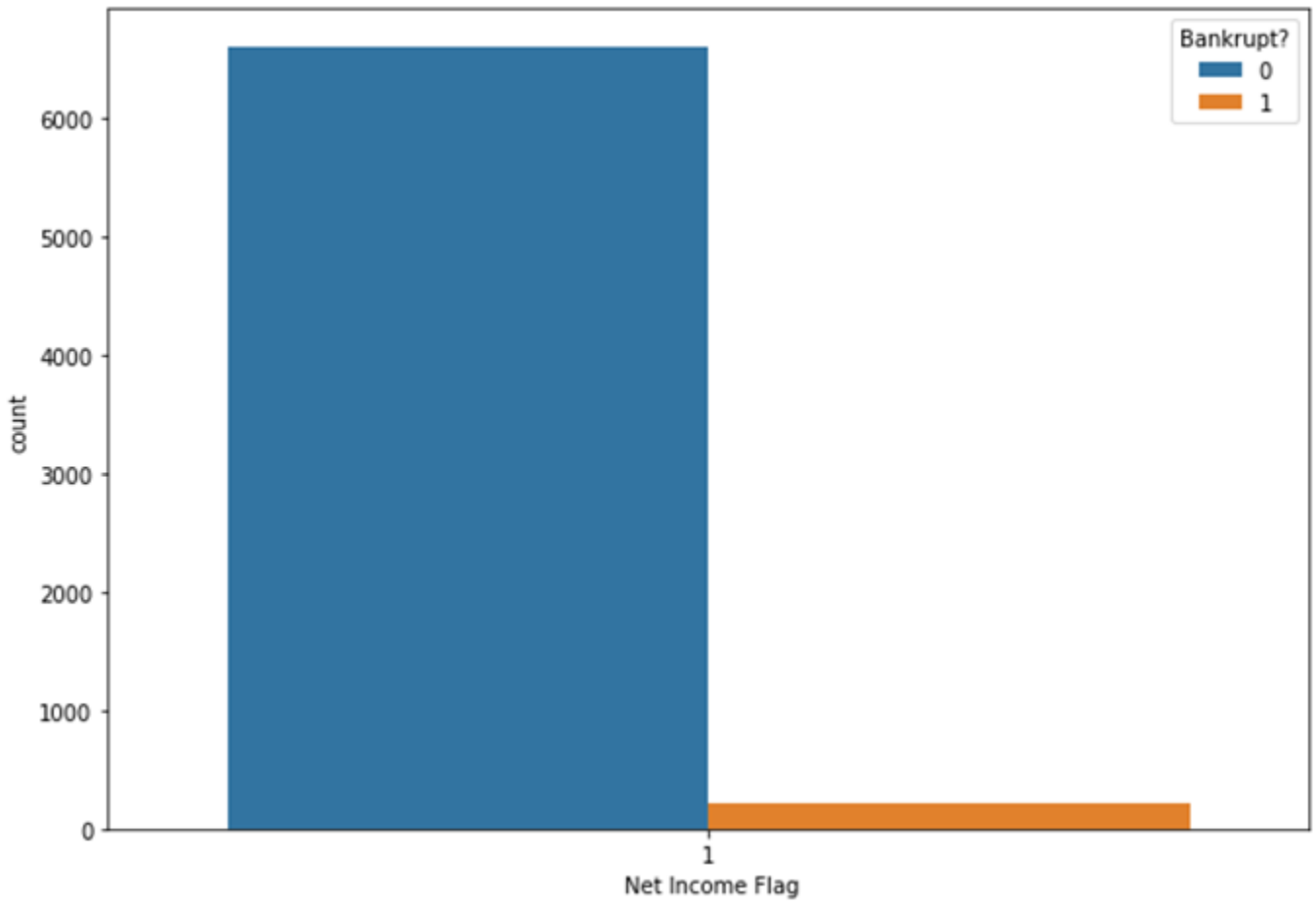

Figure 7: Net Income Flag compared to Bankrupt

2.3. Details Analysis

a. Checking NaN

```
[col for col in bank_data if bank_data[col].isnull().sum() > 0]
```

```
[]
```

There is no missing data.

b. Checking Duplicate

```
bank_data.duplicated().sum()
```

0

There are no duplicated rows.

c. Checking Objective column

```
bank_data['Bankrupt?'].value_counts()
```

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

```
print(f"Financially stable:
{round(bank_data['Bankrupt?'].value_counts()[0]/len(bank_data) * 100,2)} %")

print(f"Financially unstable:
{round(bank_data['Bankrupt?'].value_counts()[1]/len(bank_data) * 100,2)} %")
```

```
Financially stable: 96.77 %
Financially unstable: 3.23 %
```

```
plt.figure(figsize=(10, 7))

sns.countplot(bank_data['Bankrupt?'])
```

```
plt.title('Class Distributions \n (0: Financially Stable || 1: Financially Unstable)',
fontsize=14)
```

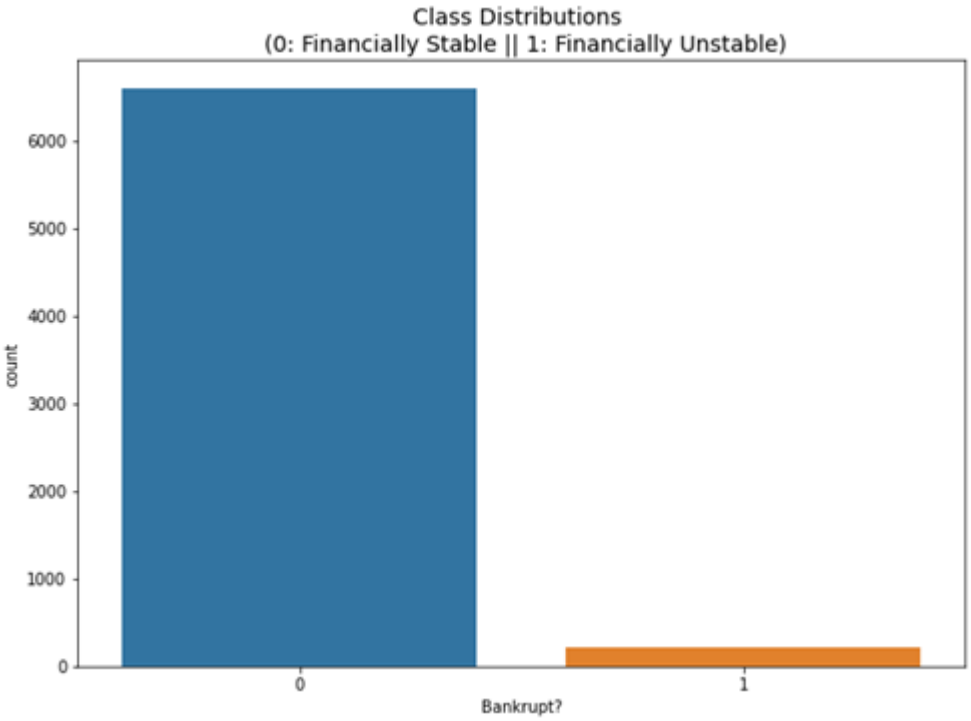

Figure 8: Financial stability compared to Bankrupt

We can see that the dataset is highly imbalanced.

2.4. Seeing correlations between columns

- Correlations between features and target column

```
plt.figure(figsize=(25, 7))

print(bank_data.corr()['Bankrupt?'].sort_values().drop('Bankrupt?'))

bank_data.corr()['Bankrupt?'].sort_values().drop('Bankrupt?').plot(kind='bar',
cmap='viridis')
```

```
Net Income to Total Assets                              -0.315457
ROA(A) before interest and % after tax                  -0.282941
ROA(B) before interest and depreciation after tax       -0.273051
ROA(C) before interest and depreciation before interest -0.260807
Net worth/Assets                                        -0.250161
```

```
                                                          ...
 Current Liability to Current Assets                    0.171306
 Borrowing dependency                                   0.176543
 Current Liability to Assets                            0.194494
 Debt ratio %                                           0.250161
 Net Income Flag                                             NaN
Name: Bankrupt?, Length: 95, dtype: float64
```

We can conclude that:

- All features have low correlation with target variable (absolute value < 0.35) .

- Many features, for instance 'Operating Profit Rate', 'Realized Sales Gross Profit Growth Rate' and 'Cash Flow To Sales', have nearly 0 correlation with target variable.

- Feature 'Net Income Flag' has no correlation (NaN) with target variable

- Correlation between features

```
plt.figure(figsize=(20, 15))
mask = np.zeros_like(bank_data.corr())
mask[np.triu_indices_from(mask)] = True
sns.heatmap(bank_data.corr(), mask=mask, cmap='viridis')
```

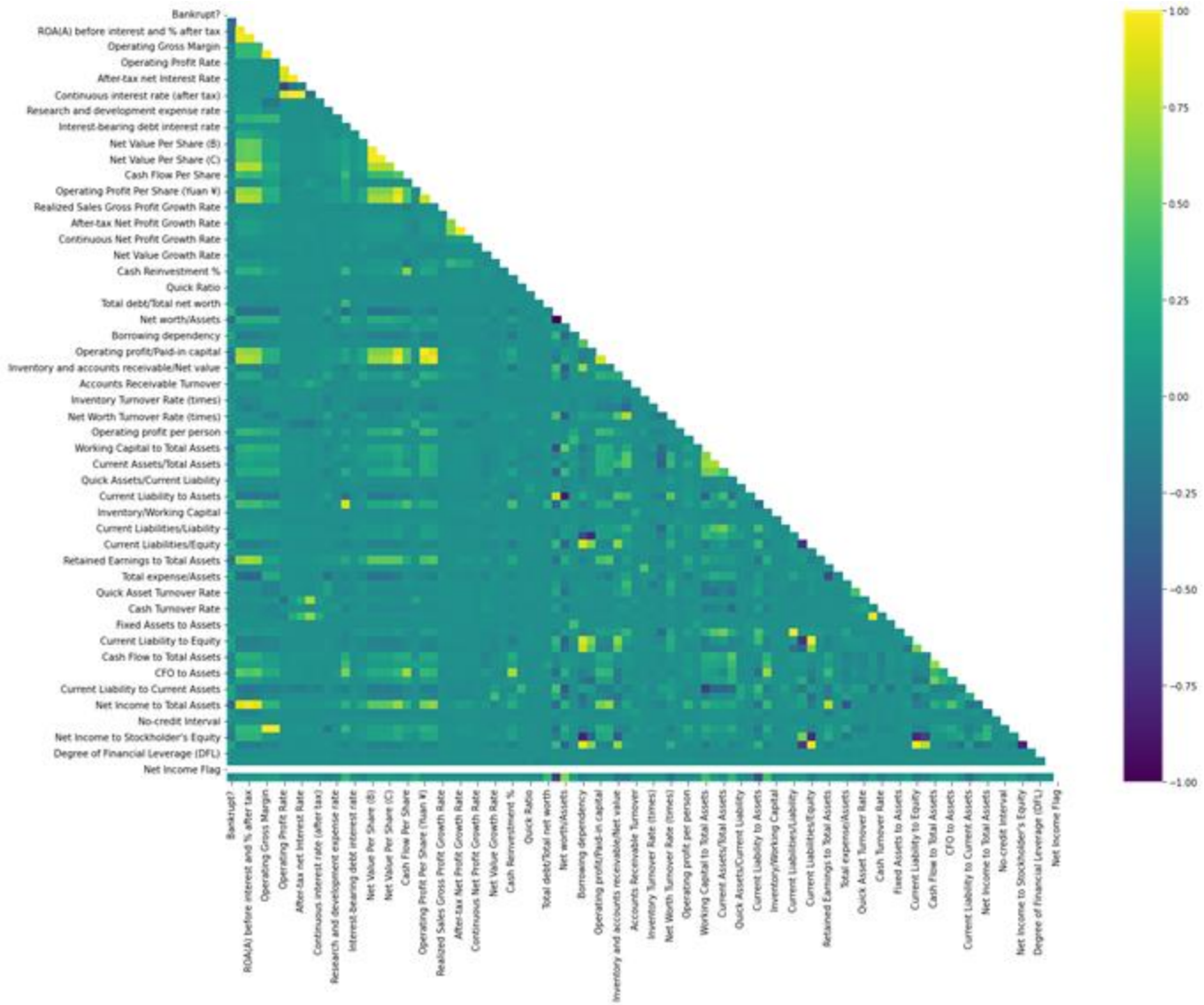

Figure 9: Correlation between all 96 features

It seems like there are many correlated features. We will single out any pair of features that has high correlation. We found that with correlation rate 0.8 and above, there are 39 pairs, which means there are much duplicated information among features.

We then analyze the top ten positively and negatively correlated attributes.

```
positive_corr = bank_data[numeric_features].corrwith(bank_data["Bankrupt?"])
                                           .sort_values(ascending=False)[:10]
                                           .index
                                           .tolist()

negative_corr = bank_data[numeric_features].corrwith(bank_data["Bankrupt?"])
                                           .sort_values()[:10]
                                           .index
                                           .tolist()
```

```
positive_corr_df = bank_data[positive_corr + ["Bankrupt?"]]
negative_corr_df = bank_data[negative_corr + ["Bankrupt?"]]

x_value = positive_corr_df.columns.tolist()[-1]
y_value = positive_corr_df.columns.tolist()[:-1]
```

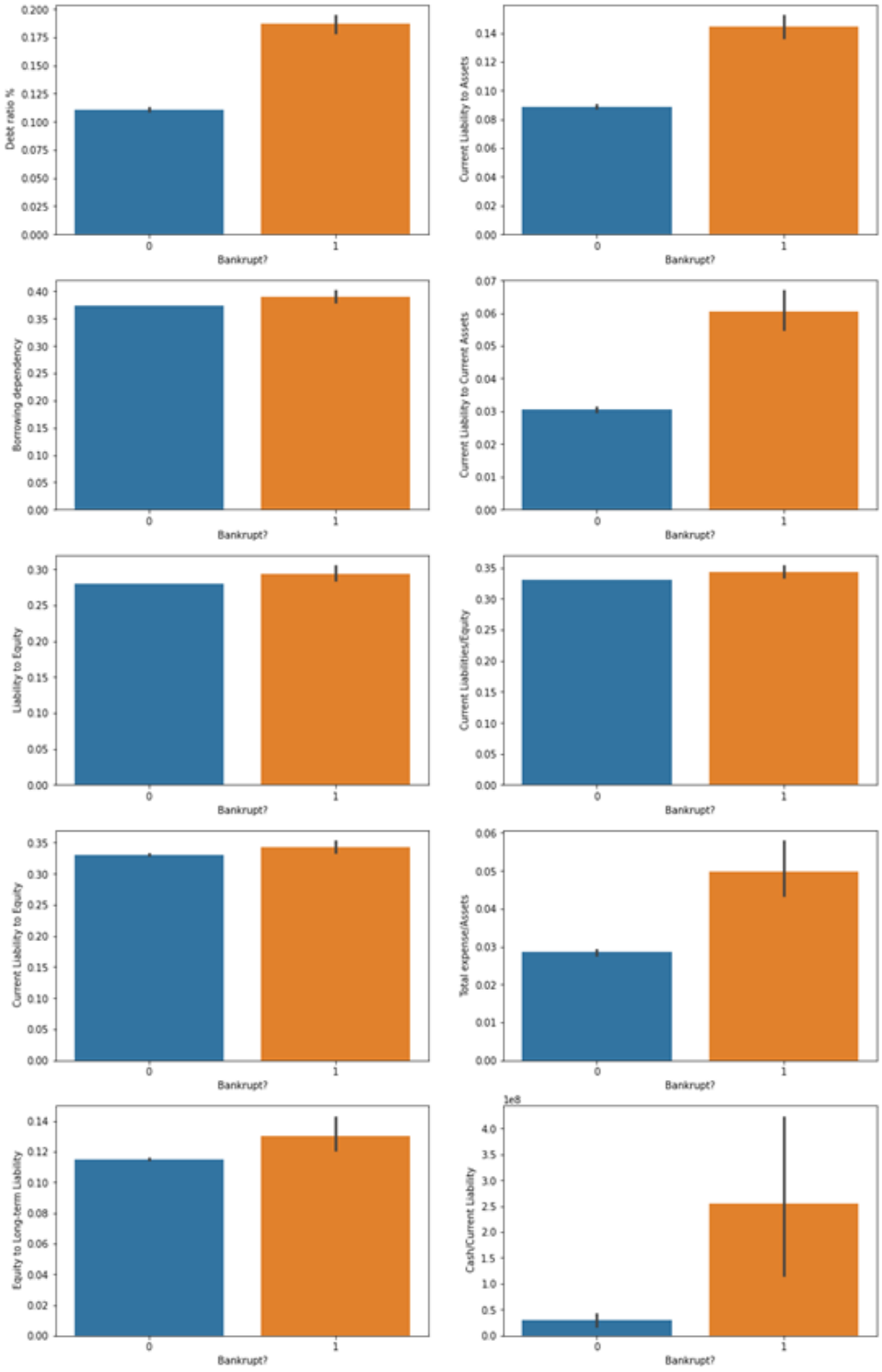

Figure 10: Correlation between top 10 positively correlated features

We see that three attributes Debt Ratio %, Current Liability To Assets, Current Liability To Current Assets are commonly high in bankrupt organizations.

```
x_value = negative_corr_df.columns.tolist()[-1]

y_value = negative_corr_df.columns.tolist()[:-1]
```

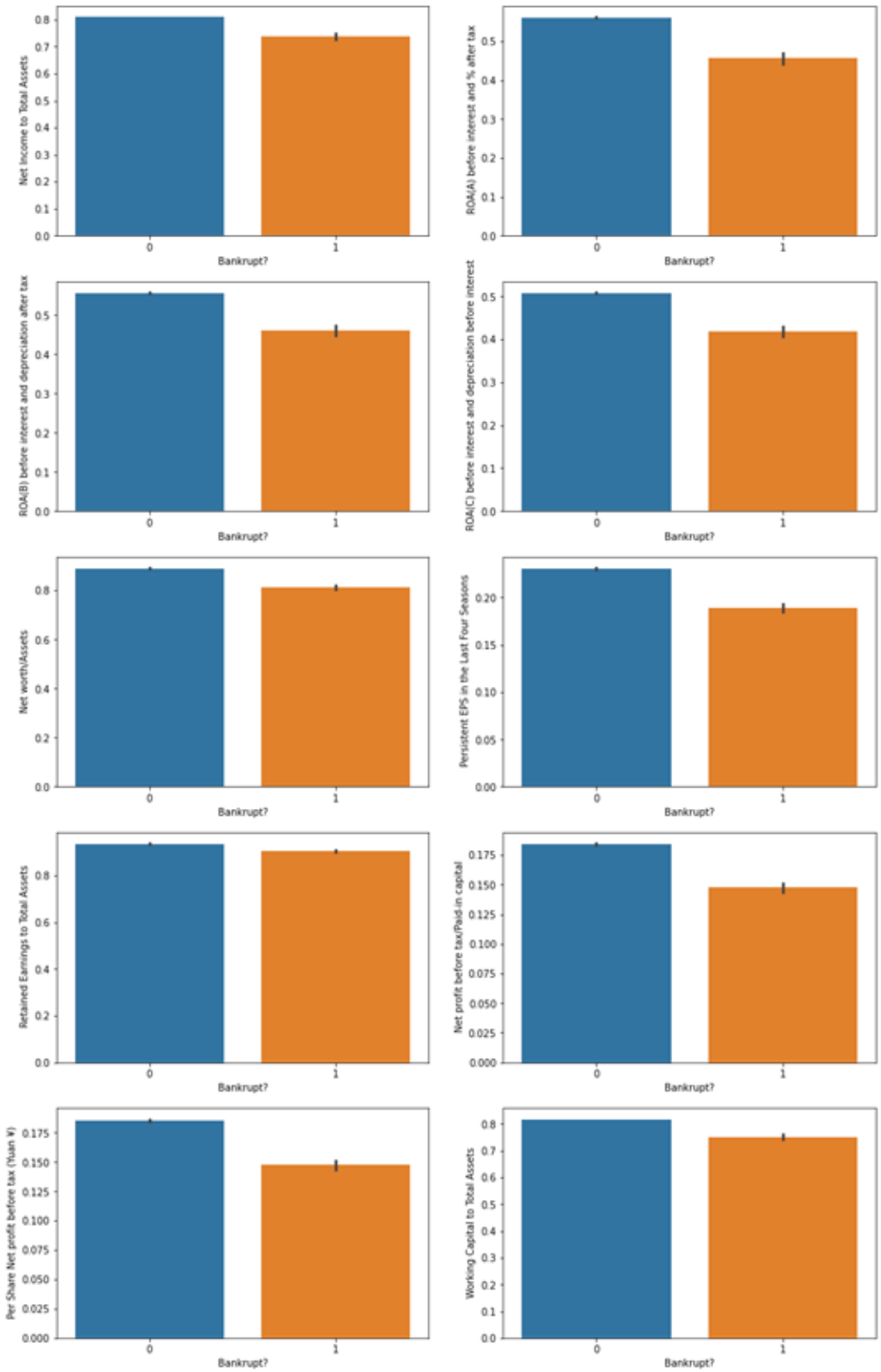

Figure 11: Correlation between top 10 negatively correlated features

These attributes show us that the more the assets and earning of a company, the less likely is the organization to be bankrupt.

We then check the relation of top ten positive and negative correlation attributes among each other

```
relation = positive_corr + negative_corr
plt.figure(figsize=(20, 14))
mask = np.zeros_like(bank_data[relation].corr())
mask[np.triu_indices_from(mask)] = True
sns.heatmap(bank_data[relation].corr(), mask=mask, annot=True, cmap='viridis')
```

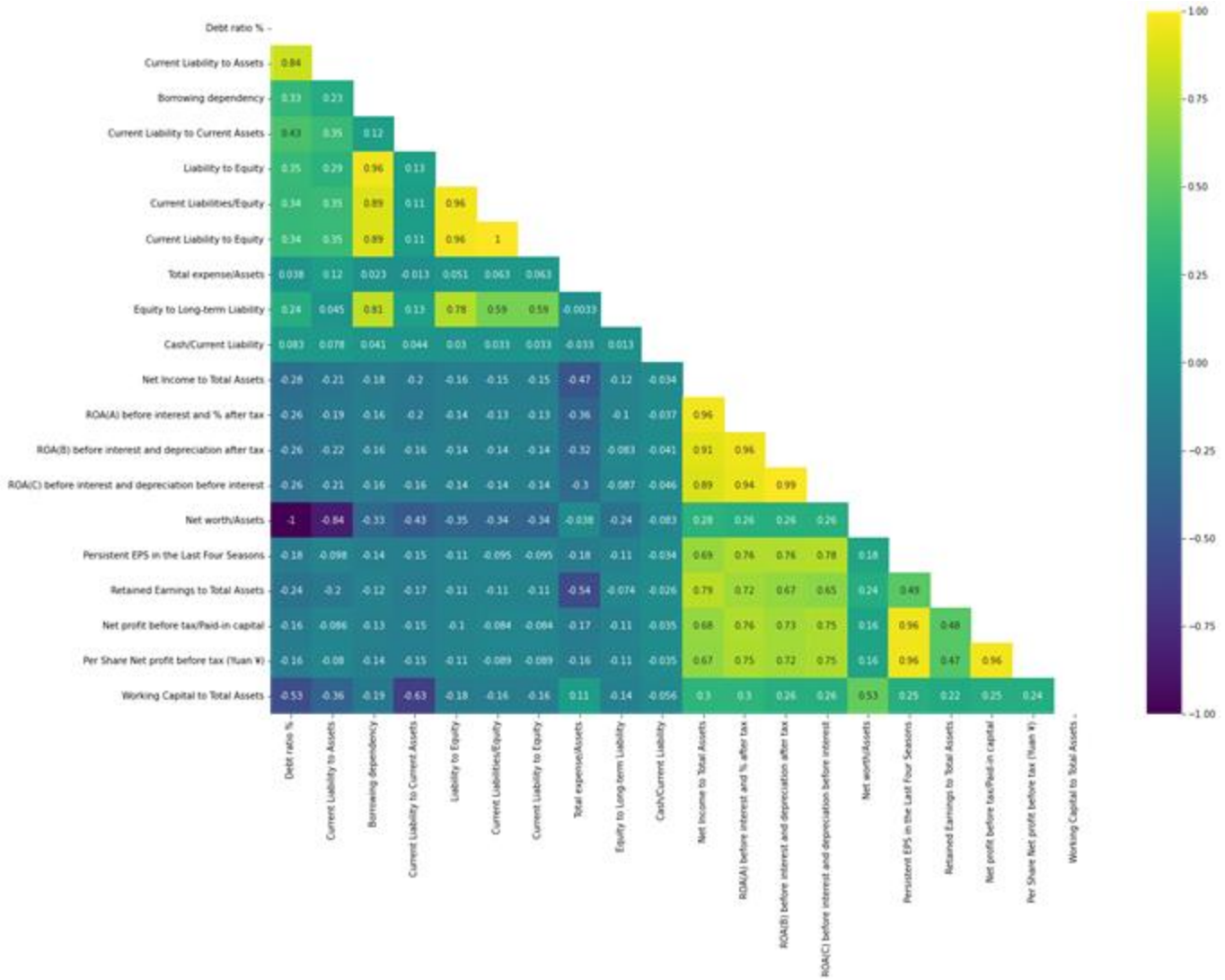

Figure 12: relation of top ten positive and negative correlation attributes among each other

2.5. Checking single valued features

```
feature_value_counts = []

for feature in bank_data.columns[1:]:
```

```
    feature_value_counts.append([feature, bank_data[feature].value_counts().shape[0]])

feature_value_counts = sorted(feature_value_counts, key=lambda x: x[1])
```

```
single_valued_features = [feature[0] for feature in feature_value_counts if feature[1]
== 1]

print(single_valued_features)
```

```
[' Net Income Flag']
```

We can safely drop this feature because it has no effect on our model.

```
bank_data = bank_data.drop(columns=single_valued_features)
```

2.6. Checking distribution and outliers in each feature

```
bank_data.hist(figsize=(80, 100))
plt.show()

for i in range(10):
    plt.figure(figsize=(15, 10))
    sns.boxplot(data=bank_data.iloc[:, i: i + 10], orient='h')
    plt.show()
```

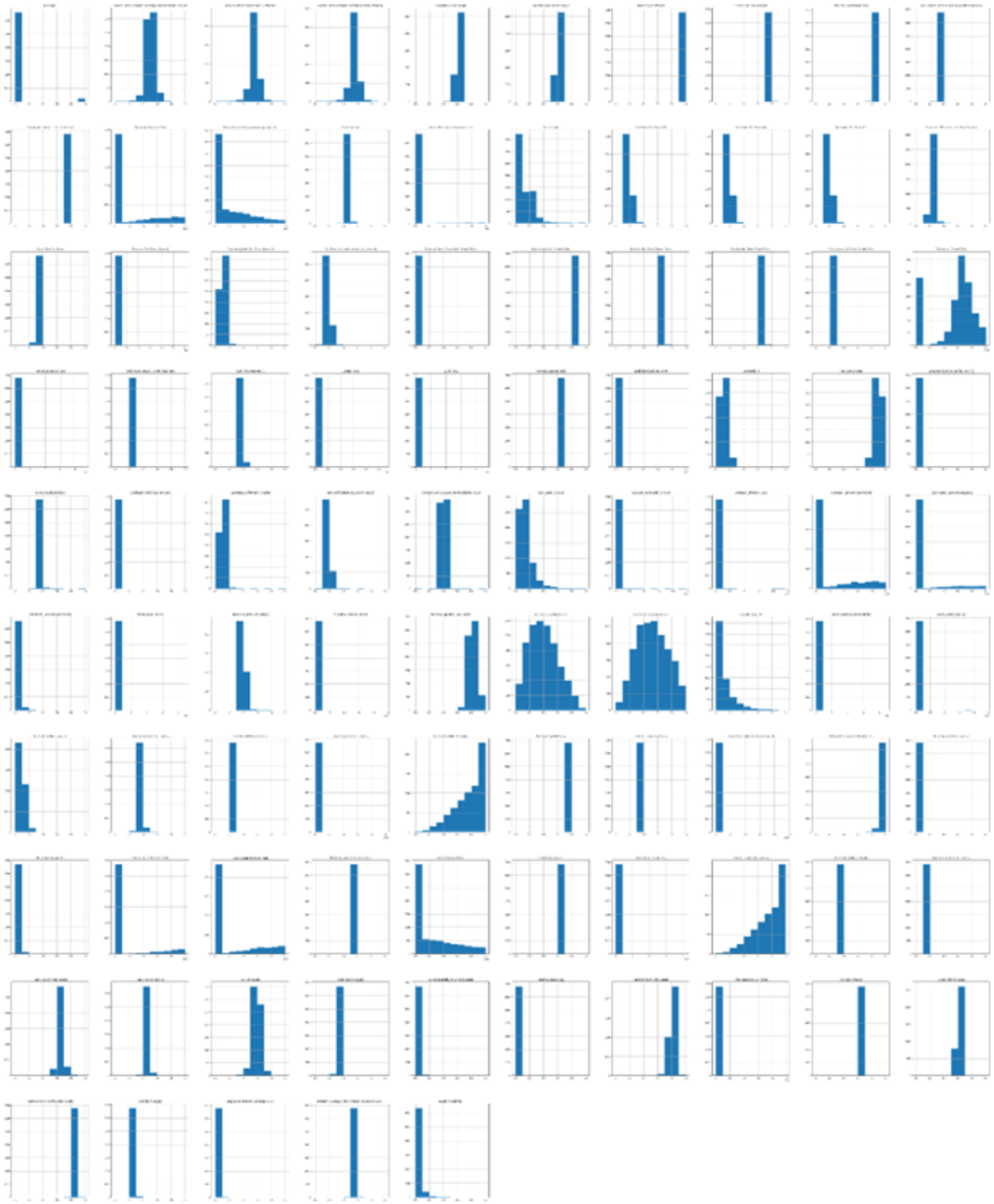

Figure 13: Distribution of each feature

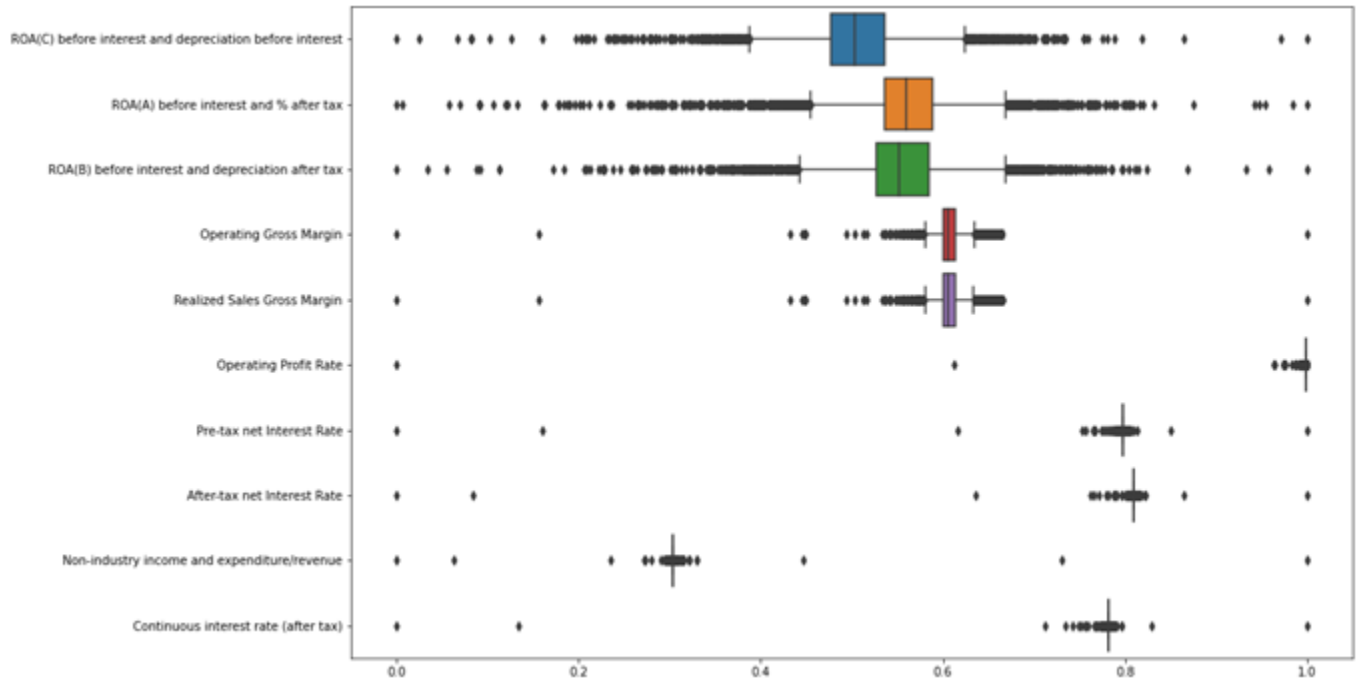

Figure 14: Outliers of each feature

We can conclude that most data is rich on outliers, and in some the values are located in just one bin! However, our topic does not cover outlier handling, so we do not go further from here.

2.7. Summary of EDA

a. About data

- This dataset is fairly small (6819 observations) and has been scaled.

- There is no missing value and duplicated rows.

- Features are highly correlated with each other while we find no remarkable correlations between them and the target variable.

- Data is highly imbalanced in target variable with nearly 97% of class 0 (non-bankrupt).

- Most of the data is rich on outliers, and in other the values are located in just one bin.

b. Some conclusions

- The number of organizations that have gone bankrupt in the years between 1999 – 2000 is few.

- Several companies possess many assets, which is always a good sign for an organization.

- An organization cannot guarantee not being bankrupt, although owning several assets.

- The organizations in the dataset are running into losses for the past two years as their net income poses to be negative.

- Very few of the organizations that have had negative income in the past two years suffer from bankruptcy.

- It is observed that “Debt Ratio %, Current Liability To Assets, Current Liability To Current Assets" attributes are a few of the attributes that have a high correlation with the target attribute.

- An increase in the values of the attributes “Debt Ratio %, Current Liability To Assets, Current Liability To Current Assets” causes an organization to suffer heavy - losses, thus resulting in bankruptcy.

- An increase in the values of the attributes that have a negative correlation with the target attribute helps an organization avoid bankruptcy.

- There seems to be a relation between attributes that have a high correlation with the target attribute and a low correlation with the target attribute.

- We observed several correlations among the top 20 attributes, one of which being “Net Worth/Assets and Debt Ratio %” that is negatively correlated with one another.

3. Data Processing

3.1. Handling imbalanced data

We oversampled the data using 4 techniques:

- SMOTE

```
smote = ('smote', SMOTE(sampling_strategy=1.0, random_state=42))
```

- Borderline SMOTE

```
borderline_smote = ('borderline_smote', BorderlineSMOTE(sampling_strategy=1.0,
random_state=42))
```

- SVMSMOTE

```
svm_smote = ('svm_smote', SVMSMOTE(sampling_strategy=1.0, random_state=42))
```

- ADASYN

```
adasyn = ('adasyn', ADASYN(sampling_strategy=1.0, random_state=42))
```

3.2. Dimension reduction

We visualized effects of dimensionality reduction techniques by reducing the dimension of data to 2D and 3D. We handled the data through linear methods and Manifold methods.

For Linear methods, we have chosen Principal Component Analysis (PCA), Factor Analysis (FA) and Independent Component Analysis (ICA). All 3 ran on 2 components and 3 components, with PCA additionally ran with 99.99% variability.

For Manifold methods, we have chosen Kernel PCA, Isometric Mapping, Distributed Stochastic Neighbor Embedding (t-SNE), Uniform Manifold Approximation and Projection (UMAP) and Locally Linear Embedding (LLE). Each was run on 2 components and 3 components.

Through PCA, we keep 99.99% variability of data with just 11 features

3.3. Feature selection

Feature selection was done using Backward Elimination, Forward Elimination and Random Forest. After feature selection was done, we got this figure of important features.

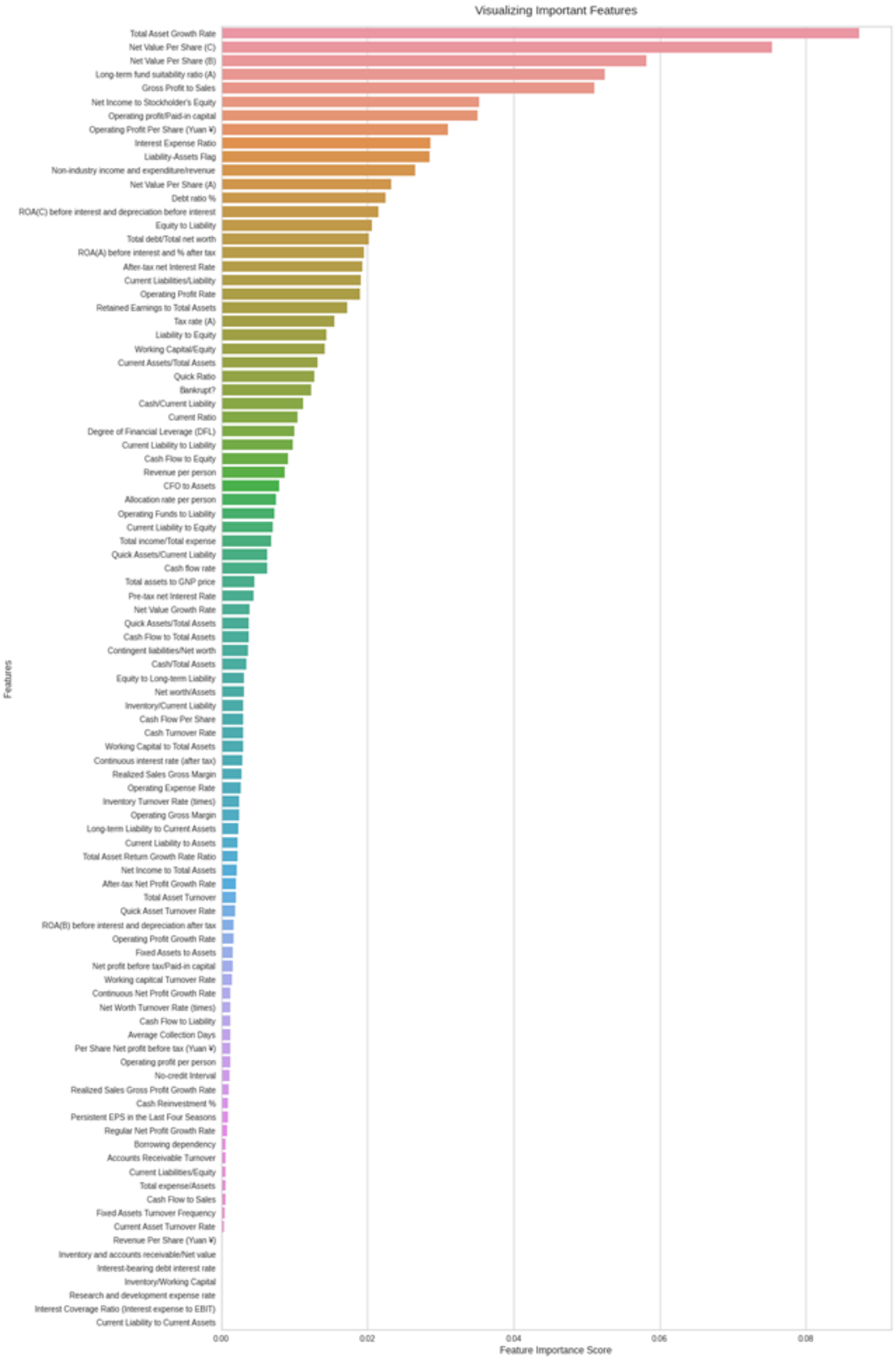

Figure 15: Feature importance

4. Training and Evaluation

The baseline model was trained using SVM_SMOTE and XGBoost, then was cross-validated with Grid Search CV. Then, we applied a dimension reduction method before training the model again. The methods are:

- PCA
- FA
- ICA
- Kernel-PCA
- Isometric Mapping (IsoMap)
- LLE

For the Machine Learning approaches, we first built a prediction model. We splitted the original dataset into two datasets, named "training and validation" and "test". We created a new dataset named "over-sampled training and validation" from the "training and validation" dataset using SMOTE. Afterwards, we split the "training and validation" dataset into two datasets, which are the training and the validation datasets. Finally, we SMOTE over-sampled the training dataset. The two datasets, which are the over-sampled training dataset and the "over-sampled training and validation" dataset, would be used for training purposes. The other two datasets, which are the validation dataset and the test dataset, would be used for testing and evaluating purposes. Figure 16 should summarize the process.

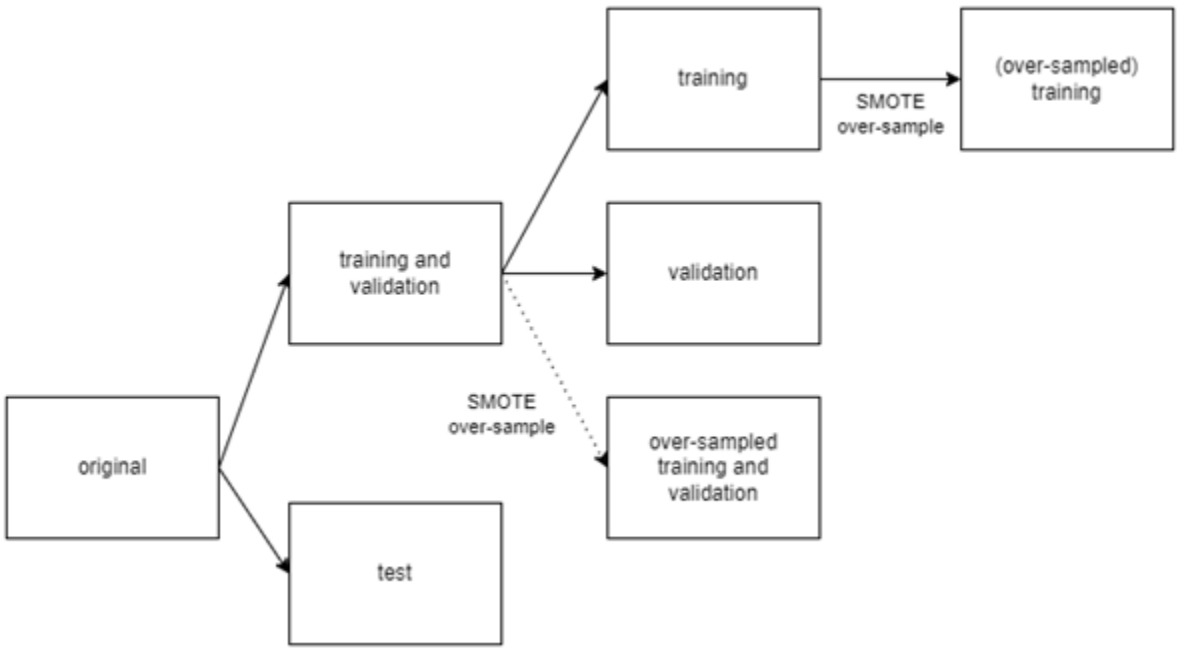

Figure 16. Datasets used in Machine Learning approaches

Then, we trained the model using Bagging, AdaBoost and Gradient Boosting.

VI. Result and Discussion

1. SVM_SMOTE + XGBoost model

name	params	test_f2_score	mean_train_score
SVM_SMOTE + XGBoost	{}	0.5462184873949579	0.7965208239559292
FA + SVM_SMOTE + XGBoost	{'fa__n_components': 4}	0.5409836065573771	0.6185728774369207
ICA + SVM_SMOTE + XGBoost	{'ica__algorithm': 'parallel', 'ica__fun': 'logcosh', 'ica__n_components': 36}	0.5058365758754865	0.7172742886257952
PCA + SVM_SMOTE + XGBoost	{'pca__n_components': 77}	0.4713114754098361	0.7652492350548601
LLE + SVM_SMOTE + XGBoost	{'lle__n_components': 9, 'n_neighbors'=20}	0.23890784982935154	0.37309501239867965
Isomap + SVM_SMOTE + XGBoost	{'iso__n_components': 15, 'iso__n_neighbors':10}	0.16363636363636364	0.4314972530580296
KernelPCA + SVM_SMOTE + XGBoost	{'kernel_pca__n_components': 80}		0.525946111938067

Figure 17. Training result on SVM_SMOTE + XGBoost model

We use F2 score instead of F1 score because the F2 score is more suitable in our application where it's more important to classify correctly as many positive samples as possible, rather than maximizing the number of correct classifications, In short, we want to lower the importance of precision and increase the importance of recall.

For dimensional reduction tasks, we did some research and found these models: FA, ICA, PCA, LLE, IsoMap Embedding and KernelPCA. Applying those models, together with the base model (no features reduction), to our data set, we collect some indicators in the parameter-tuning duration, which is shown in the table above.

The three models LLE, IsoMap and KernelPCA performed really poor since their F2 scores are pretty low, about 0.24 and 0.16 for the first two, respectively, and especially with the KernelPCA, we did not see any promising result in training and validation, so we did not tune the parameters further to find the best possible result of this method.

With the other three models, the PCA produced really good outputs for the training task (at about 0.77, which is the highest among them) but was awful in the testing task, (absolutely overfitting). Compared to

FA and ICA, they gave much better generalization results, especially the FA with the F2 score is nearly approximate to that of the base model. It seems really promising if we spend more time tuning parameters for this model.

2. Machine Learning models

2.1. Finding the best component models

Halving grid search cross-validation (halving grid search CV) is a method to search for the best combination of hyperparameters. "The search strategy starts evaluating all the candidates with a small amount of resources and iteratively selects the best candidates, using more and more resources" [14].

All the classifier classes of scikit-learn have their own default hyperparameters' values. Those default values are usually good in many cases. Therefore, in order to find the best values for hyperparameters possible without costing too much time, for each machine learning model, we choose one particular hyperparameter to tune, and then we search for every possible integer from half to double the default value of that hyperparameter using halving grid search CV. For instance, in RF, we tuned the "n_estimators" hyperparameter by trying all integers from 50 to 200 since the default value of "n_estimators" is 100. Finally, we evaluate the generalization f2 score of the best model that the algorithm found on the validation dataset.

For neural networks, we found that they were really inconsistent in performance, therefore they aren't applicable for this problem.

For random forest classifiers, after searching for the values of "n_estimators" from 50 to 200, the best estimator we found has a number of trees that equals 157. Its generalization score is 0.6862745098039215. The results of the searching process are represented in Figure 18.

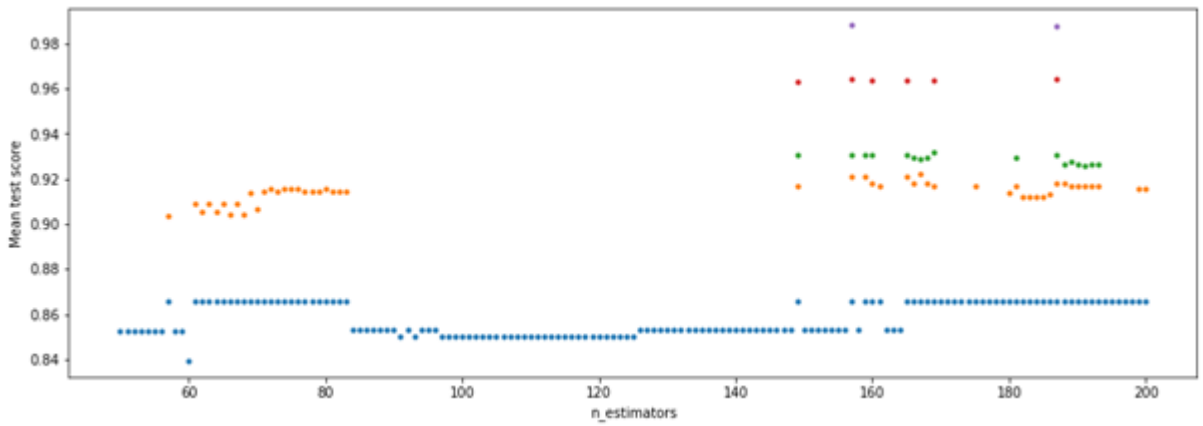

Figure 18. Results of all the halving grid search CV iterations while tuning "n_estimators" of RF.

For random forest only, we took a look at the "feature_importances_" attribute of the best model (with "n_estimators=157") that the algorithm found above (Figure 19), and drew the conclusion that nearly all the attributes of the dataset are important, so that might be the reason why filtering the data features reduced the scores in the "machine learning models" section.

```
In [28]:
y_val_pred_best_rf = best_rf.predict(X_val)
f2_score(y_val, y_val_pred_best_rf)

Out[28]:
0.6862745098039215

In [29]:
best_rf.feature_importances_

Out[29]:
array([0.01862026, 0.03020667, 0.02505078, 0.00365536, 0.00357177,
       0.01004923, 0.02502298, 0.05430697, 0.01377716, 0.03491197,
       0.0035351 , 0.00314161, 0.00300699, 0.00886403, 0.00245806,
       0.00503899, 0.00843891, 0.00793887, 0.05024937, 0.0031459 ,
       0.00365134, 0.00428008, 0.03161663, 0.00265778, 0.00295246,
       0.00509644, 0.00425495, 0.00393679, 0.0058943 , 0.01888417,
       0.00268771, 0.00262984, 0.00245653, 0.00898183, 0.0107566 ,
       0.05172119, 0.02041576, 0.01951123, 0.00246444, 0.02546082,
       0.00427747, 0.00852287, 0.03407256, 0.00281954, 0.00343692,
       0.00368019, 0.00530178, 0.00433191, 0.00583851, 0.00259647,
       0.00568187, 0.00404241, 0.0081111 , 0.00235277, 0.00334439,
       0.00307103, 0.0042536 , 0.00437451, 0.00305103, 0.00623993,
       0.00483375, 0.00890061, 0.0036752 , 0.00516508, 0.00277397,
       0.01148774, 0.00585817, 0.03540003, 0.03749784, 0.00357154,
       0.00604843, 0.00668659, 0.00229488, 0.00808419, 0.00331315,
       0.0026741 , 0.00410118, 0.01026889, 0.00940613, 0.00339837,
       0.00283435, 0.00294198, 0.00225487, 0.00650945, 0.00014869,
       0.057908  , 0.00501729, 0.00279171, 0.00326181, 0.02321043,
       0.02540753, 0.0089158 , 0.00773526, 0.        , 0.01092031])
```

Figure 19. The "feature_importances_" attribute of the best random forest found.

For bagging classifiers, after searching for the values of "n_estimators" from 5 to 20, the best estimator we found has a number of trees that equals 17. Its generalization score is 0.46391752577319584. The results of the searching process are represented in Figure 20.

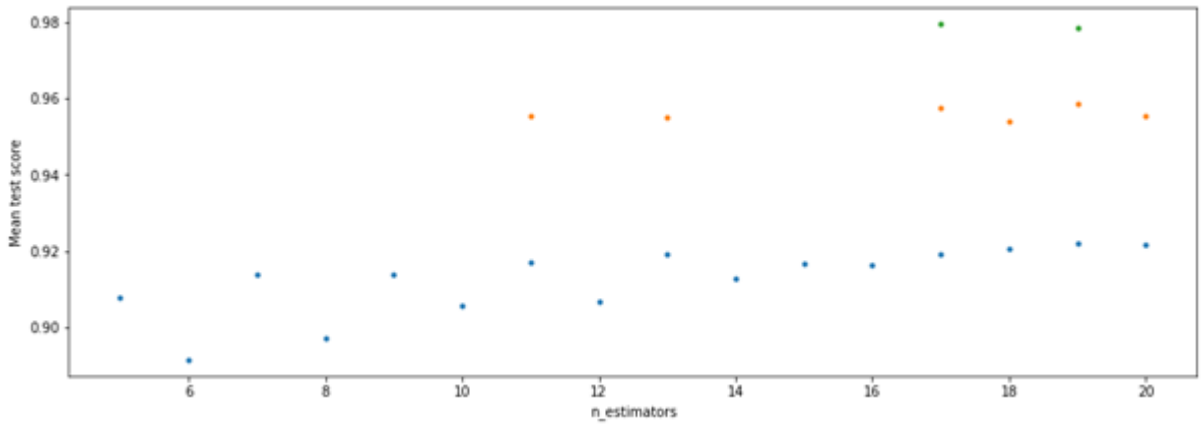

Figure 20. Results of all the halving grid search CV iterations while tuning "n_estimators" of bagging.

For AdaBoost classifiers, after searching for the values of "n_estimators" from 25 to 100, the best estimator we found has a number of trees that equals 97. Its generalization score is 0.5701754385964912. The results of the searching process are represented in Figure 21.

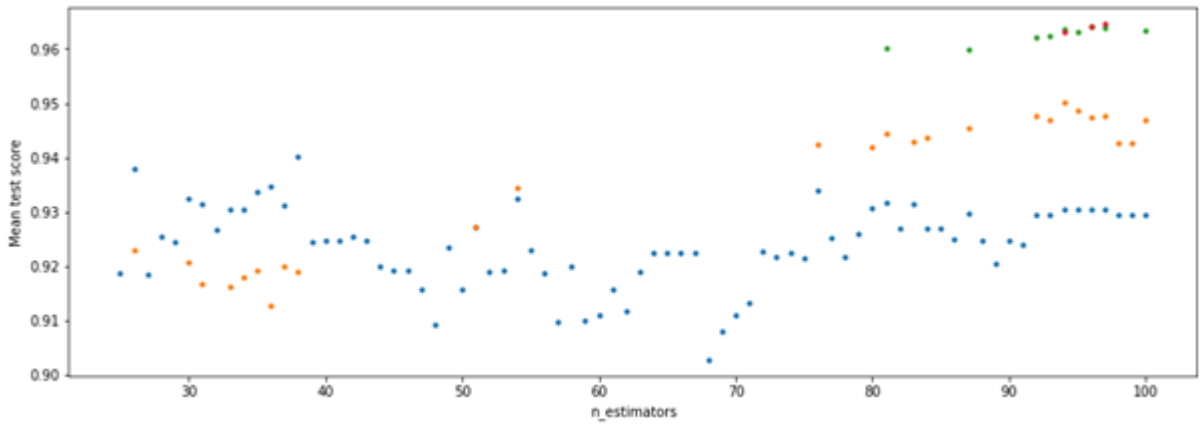

Figure 21. Results of all the halving grid search CV iterations while tuning "n_estimators" of AdaBoost.

For gradient boosting classifiers, after searching for the values of "n_estimators" from 50 to 200, the best estimator we found has a number of trees that equals 170. Its generalization score is 0.7017543859649122. The results of the searching process are represented in Figure 22.

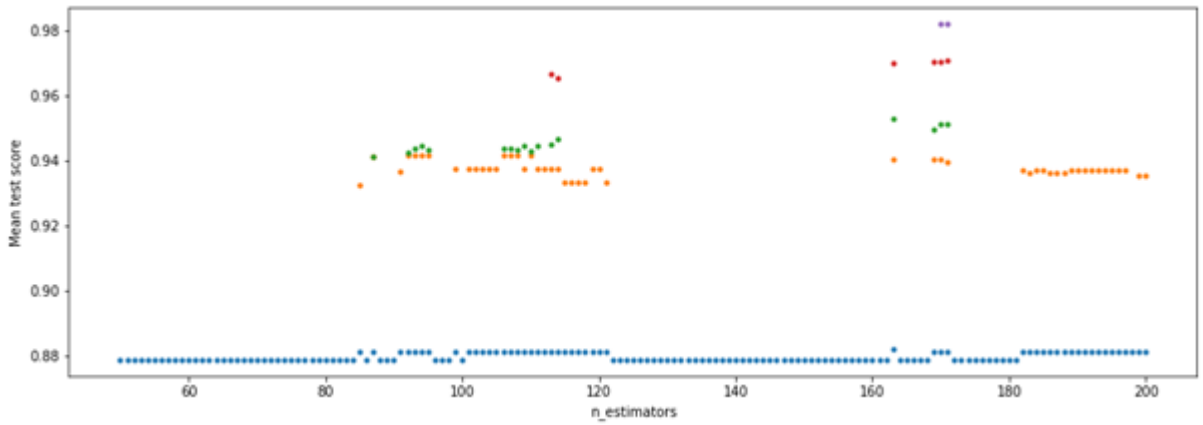

Figure 22. Results of all the halving grid search CV iterations while tuning "n_estimators" of gradient boosting.

2.2. Ensembling the best models into one model

To prevent biased predictions, we only use a simple voting classifier [3]. The four best models above are put into the classifier and retrained using the "over-sampled training and validation" dataset to get the final voting classifier. Finally, this classifier is tested on the test dataset to check its generalizability (in other words, check if its predictions will be good in the future). Its generalization f2 score is 0.6168831168831169 (Figure 23), which is pretty good for a hard-to-predict problem like this company bankruptcy prediction.

```
In [59]: voting.fit(X_train_val_over, y_train_val_over)

Out[59]: VotingClassifier(estimators=[('rf',
                                       RandomForestClassifier(n_estimators=157,
                                                              random_state=42)),
                                      ('bagging',
                                       BaggingClassifier(n_estimators=17,
                                                         random_state=42)),
                                      ('ada',
                                       AdaBoostClassifier(n_estimators=97,
                                                          random_state=42)),
                                      ('gb',
                                       GradientBoostingClassifier(n_estimators=170,
                                                                  random_state=42))])

In [60]: y_test_pred_voting = voting.predict(X_test)
         f2_score(y_test, y_test_pred_voting)

Out[60]: 0.6168831168831169
```

Figure 23. Testing result of the final voting classifier.

VII. Conclusion

Company Bankruptcy prediction is a tricky topic. There are too many features to take into account. We have applied methods learned while attending courses Applied Statistics and Experimental Design, and Machine Learning. Tried as we may with different analytical methods and models, we only have found the result to be just slightly better when done dimension reduction. We did not expect an excellent result either, since as mentioned this is a tricky topic.

In summary of our work, the best model can predict, with about 61% accuracy, which company will go bankrupt. This could be further improved should there be more time to tune the parameters.

Due to time limitations, we could not explore all avenues surrounding bankruptcy prediction inaccuracies. For future works, we would like to consider the following:
- Exploring other dimensionality reduction methods and their result when applied into the data.
- More hyperparameter tuning and model tuning.
- Examining our results on other related datasets, to make verifications and conclusions.

VIII. References

[1] Deron Liang and Chih-Fong Tsai, National Central University, Taiwan, **UCI Taiwanese Bankruptcy Prediction Data Set,** https://archive.ics.uci.edu/ml/datasets/Taiwanese+Bankruptcy+Prediction

[2] Deron Liang, Chia-Chi Lu, Chih-Fong Tsai, Guan-An Shih, **Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study**, European Journal of

Operational Research, Volume 252, Issue 2, 2016, Pages 561-572, ISSN 0377-2217, https://www.sciencedirect.com/science/article/pii/S0377221716000412

[3] Adam Jarbøl, Axel Bevort, **Bankruptcy Prediction and Corporate Governance: Revisiting the Z-score in the post-Global Financial Crisis period**, Thesis for MSc in Applied Economics and Finance, May 2020, https://research.cbs.dk/en/studentProjects/52e0f1bb-c2c1-43dd-bab8-ee64cfb081d0

[4] Shailer, Greg. Corporate Governance, in D. C. Poff, A. C. Michalos (eds.), **Encyclopedia of Business and Professional Ethics,** Springer International Publishing AG, 2018, https://doi.org/10.1007/978-3-319-23514-1_155-1

[5] Chawla, N., Bowyer, K., Hall, L. and Kegelmeyer, W., **SMOTE: Synthetic Minority Over-sampling Technique**, Journal of Artificial Intelligence Research, 2002, Volume 16 https://www.jair.org/index.php/jair/article/view/10302

[6] https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

[7] Mark A. Hall, **Correlation-based Feature Selection for Machine Learning** (PhD thesis). University of Waikato, April 1999. https://www.cs.waikato.ac.nz/~mhall/thesis.pdf

[8] https://machinelearningcoban.com/2017/06/15/pca/#3-principal-component-analysis

[9] Altman, Edward I., **Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy**. Journal of Finance. 23 (4), September 1968, Pages 189–209. doi:10.1111/j.1540-6261.1968.tb00843.x.

[10] James A. Ohlson, **Financial Ratios and the Probabilistic Prediction of Bankruptcy**, Journal of Accounting Research Vol. 18, No. 1 ,Spring, 1980, Pages 109-131. https://doi.org/10.2307/2490395

[11] Jackson, Richard H.G.; Wood, Anthony. **The performance of insolvency prediction and credit risk models in the UK: A comparative study**, The British Accounting Review. 45 (3), 2013, Pages 183–202. https://www.sciencedirect.com/science/article/abs/pii/S0890838913000498

[12] Zhang, Yudong; Shuihua Wang; Genlin Ji . **A Rule-Based Model for Bankruptcy Prediction Based on an Improved Genetic Ant Colony Algorithm**. Mathematical Problems in Engineering. 2013: 753251. https://www.hindawi.com/journals/mpe/2013/753251/

[13] Evergrande, **2021 Interim Report,** https://doc.irasia.com/listco/hk/evergrande/interim/2021/intrep.pdf

[14] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., ... & Varoquaux, G. (2013). ECML PKDD Workshop: languages for data mining and machine learning. API Design for Machine Learning Software: Experiences from the Scikit-Learn Project, 108-122.

Hung, Dang & Thi Thanh Binh, Vu. (2022). **Data Mining for Bankruptcy Prediction: An Experiment in Vietnam**., 2002, Pages 175-184. 10.15439/2021KM30. https://www.researchgate.net/publication/358739597_Data_Mining_for_Bankruptcy_Prediction_An_Experiment_in_Vietnam

https://en.wikipedia.org/wiki/Altman_Z-score

https://en.wikipedia.org/wiki/Ohlson_O-score

Last, F., Douzas, G., & Bacao, F. (2017). Oversampling for imbalanced learning based on k-means and smote. arXiv preprint arXiv:1711.00837.

Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. The Journal of Machine Learning Research, 18(1), 559-563.

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.

Wikipedia contributors. (2022d, July 7). Lasso (statistics). Wikipedia. https://en.wikipedia.org/wiki/Lasso_(statistics)

Brownlee, J. (2021, January 4). Bagging and random forest for imbalanced classification. Machine Learning Mastery. Retrieved July 12, 2022, from https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/

IX. Appendices

Information of attributes in the dataset

Col.	Corporate Governance Indicators	Further explanation (if needed)
Y	Bankrupt?: Class label	
X1	ROA(C) before interest and depreciation before interest: Return On Total Assets(C)	**ROA**: Net income / Average total assets measures how well a firm uses its money **Depreciation**: the action of re-evaluating asset due to loss in value by expiration or novel technology
X2	ROA(A) before interest and % after tax: Return On Total Assets(A)	
X3	ROA(B) before interest and depreciation after tax: Return On Total Assets(B)	
X4	Operating Gross Margin: Gross Profit/Net Sales	**Net sales** is different from **gross profit** in the sense that **net sales** is gross sales but deducting sales returns, allowances, and discount; while **gross profit** is purely amount of money after deducting cost of products
X5	Realized Sales Gross Margin: Realized Gross Profit/Net Sales	**Realized profit** means the profit gained **after completing transactions** *e.g. A bank invest in some stocks and haven't sold them. Then if the stock price increases, the investment profit is counted as **unrealized profit***
X6	Operating Profit Rate: Operating Income/Net Sales	
X7	Pre-tax net Interest Rate: Pre-Tax Income/Net Sales	
X8	After-tax net Interest Rate: Net Income/Net Sales	

X9	Non-industry income and expenditure/revenue: Net Non-operating Income Ratio	
X10	Continuous interest rate (after tax): Net Income-Exclude Disposal Gain or Loss/Net Sales	
X11	Operating Expense Rate: Operating Expenses/Net Sales	
X12	Research and development expense rate: (Research and Development Expenses)/Net Sales	
X13	Cash flow rate: Cash Flow from Operating/Current Liabilities	**Liabilities**: quite similar to debt, but exists in more form than just only money, such as bank notes, salaries, payment for suppliers are liabilities but **not debt**
X14	Interest-bearing debt interest rate: Interest-bearing Debt/Equity	Indicates how **independently** a company works. However if the value is too low, it means that the company doesn't use its capability to loan, thus cannot scale well.
X15	Tax rate (A): Effective Tax Rate	Calculated by (Total tax expand/Taxable income). In general, higher income means higher effective tax rate.
X16	Net Value Per Share (B): Book Value Per Share(B)	Can also be written as **NAVPS** or **BVPS**
X17	Net Value Per Share (A): Book Value Per Share(A)	
X18	Net Value Per Share (C): Book Value Per Share(C)	
X19	Persistent EPS in the Last Four Seasons: EPS-Net Income	EPS: **Earning per share**
X20	Cash Flow Per Share	
X21	Revenue Per Share (Yuan ¥): Sales Per Share	
X22	Operating Profit Per Share (Yuan ¥): Operating Income Per Share	
X23	Per Share Net profit before tax (Yuan ¥): Pretax Income Per Share	
X24	Realized Sales Gross Profit Growth Rate	
X25	Operating Profit Growth Rate: Operating Income Growth	
X26	After-tax Net Profit Growth Rate: Net Income Growth	
X27	Regular Net Profit Growth Rate: Continuing Operating Income after Tax Growth	
X28	Continuous Net Profit Growth Rate: Net Income-Excluding Disposal Gain or Loss Growth	
X29	Total Asset Growth Rate: Total Asset Growth	
X30	Net Value Growth Rate: Total Equity Growth	
X31	Total Asset Return Growth Rate Ratio: Return on Total Asset Growth	
X32	Cash Reinvestment %: Cash Reinvestment Ratio	
X33	Current Ratio	Current Asset/Current Liability, measures whether a firm has enough resources to meet its **short-term** obligations

X34	Quick Ratio: Acid Test	(Current Asset - Inventory)/Current Liability, **relatively similar to current ratio**, but excluding Inventory from Asset
X35	Interest Expense Ratio: Interest Expenses/Total Revenue	
X36	Total debt/Total net worth: Total Liability/Equity Ratio	
X37	Debt ratio %: Liability/Total Assets	
X38	Net worth/Assets: Equity/Total Assets	
X39	Long-term fund suitability ratio (A): (Long-term Liability+Equity)/Fixed Assets	
X40	Borrowing dependency: Cost of Interest-bearing Debt	
X41	Contingent liabilities/Net worth: Contingent Liability/Equity	Contingent liability is liability which is **stochastic** *e.g. Warranty cost, because you cannot calculate how many products need maintenance*
X42	Operating profit/Paid-in capital: Operating Income/Capital	Paid-in capital
X43	Net profit before tax/Paid-in capital: Pretax Income/Capital	
X44	Inventory and accounts receivable/Net value: (Inventory+Accounts Receivables)/Equity	
X45	Total Asset Turnover	**Asset Turnover:** Net sales/Total Asset, measures the efficiency of a company's use of its assets in generating sales revenue or sales income.
X46	Accounts Receivable Turnover	
X47	Average Collection Days: Days Receivable Outstanding	Measure how long it takes a company to collect its account receivables
X48	Inventory Turnover Rate (times)	
X49	Fixed Assets Turnover Frequency	*Asset Turnover mentioned above*
X50	Net Worth Turnover Rate (times): Equity Turnover	
X51	Revenue per person: Sales Per Employee	
X52	Operating profit per person: Operation Income Per Employee	
X53	Allocation rate per person: Fixed Assets Per Employee	
X54	Working Capital to Total Assets	
X55	Quick Assets/Total Assets	
X56	Current Assets/Total Assets	
X57	Cash/Total Assets	
X58	Quick Assets/Current Liability	
X59	Cash/Current Liability	
X60	Current Liability to Assets	
X61	Operating Funds to Liability	

X62	Inventory/Working Capital	
X63	Inventory/Current Liability	
X64	Current Liabilities/Liability	
X65	Working Capital/Equity	
X66	Current Liabilities/Equity	
X67	Long-term Liability to Current Assets	
X68	Retained Earnings to Total Assets	
X69	Total income/Total expense	
X70	Total expense/Assets	
X71	Current Asset Turnover Rate: Current Assets to Sales	Sales: **Gross Sales** in particular
X72	Quick Asset Turnover Rate: Quick Assets to Sales	
X73	Working capital Turnover Rate: Working Capital to Sales	
X74	Cash Turnover Rate: Cash to Sales	
X75	Cash Flow to Sales	
X76	Fixed Assets to Assets	
X77	Current Liability to Liability	
X78	Current Liability to Equity	
X79	Equity to Long-term Liability	
X80	Cash Flow to Total Assets	
X81	Cash Flow to Liability	
X82	Cash Flow to Assets	
X83	Cash Flow to Equity	
X84	Current Liability to Current Assets	
X85	Liability-Assets Flag	**1** if Total Liability exceeds Total Assets **0** otherwise
X86	Net Income to Total Assets	
X87	Total assets to GNP price	GNP: **Gross National Product**
X88	No-credit Interval	(Asset-Liability)/Daily Operational Cost
X89	Gross Profit to Sales	
X90	Net Income to Stockholder's Equity	
X91	Liability to Equity	
X92	Degree of Financial Leverage (DFL)	
X93	Interest Coverage Ratio (Interest expense to EBIT)	Interest Coverage Ratio (*Tỷ số khả năng trả*

		lãi): EBIT/Interest expense **EBIT**: Earnings Before Interest and Taxes
X94	Net Income Flag	**1** if Net Income is Negative for the last two years **0** otherwise
X95	Equity to Liability	