

**Corporate** **Bankruptcy**

Prediction

Using

**Machine** **Learning** **Techniques**

ADMN5006

Arnab Barua

Luke Laylo

Jad Naitali

Varun Nallamothu

Sreehari Ramadas

Ghita Tazi

# Abstract

Since the last few decades, corporate bankruptcy has been of interest to researchers around the world and a source of worry for business stakeholders including management, investors, etc. Due to the numerous elements that contribute to bankruptcy, it is not enough to rely on a single predictive model; instead, the difficulty is in identifying only the most important factors. The high-class imbalance in the data is another significant barrier that impairs the performance of the model.

# Section 1: Introduction

The team has designed and developed a machine-learning model for corporate bankruptcy prediction. In this study, we evaluate the performance of machine learning models (Linear Support Vector Classifier, Naïve Bayes, and random forest) to findings from cross-validation, and logistic regression in predicting bankruptcy one year before the occurrence.

## Data Exploration Analysis

Using visuals is one of the quickest ways to study and comprehend the data. When it comes to understanding the data's structure, the distribution of its values, and the existence of any correlations within the dataset, the findings of visual data exploration can be quite effective. Therefore, the findings of our dataset are as below:

* Identifying the datatype of all the variables:

Text, table

Description automatically generated

* Checking the presence of null values in the dataset:

Diagram

Description automatically generated with medium confidence

Table

Description automatically generated

* Correlation between the dataframe

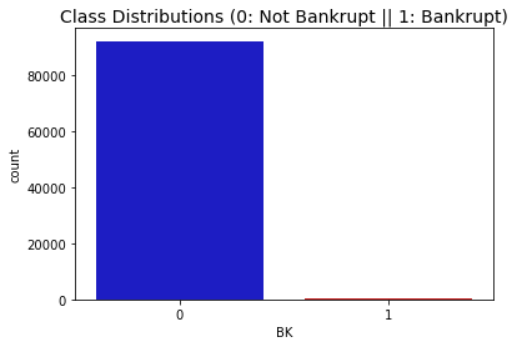
**Table

Description automatically generated**

Chart

Description automatically generated

* Plotting bar-graph to determine the distribution of bankrupt and non-bankrupt cases:



After viewing and analyzing the dataset, we have confirmed that there are a lot of problems involving the dataset. First problem we could see in the data set is that there are a lot of missing values to evaluate the data. Second, the class distribution between bankrupt and not bankrupt is highly imbalanced. Lastly, there are also a lot of extreme outliers present in the dataset presented.

# Section 2: Data Processing

To begin with, we had to design and develop a machine learning model for bankruptcy prediction 2 years into the future using the dataset provided to us. The column “BK” in the data provided denotes whether the company goes bankrupt (indicated by 1) 2 years in the future, or not (indicated by 0). - Some of the input variables included are “Assets Growth”, “Sales Growth”, “Earnings-per-share (EPS)”, “Return-on-equity (ROE)”, etc.

Our goals were to understand the data that was provided to us and create a 50/50 sub-data frame ratio of “Bankrupt” and “Non-Bankrupt”.

Determine the classifiers that we are going to use and decide which one has the higher accuracy and UAC and understand the mistakes made with imbalanced datasets.

We imported the libraries that are going to help us solve the dataset. We, then, imported the data source into the notebook and saw the basic statistics of the report.

We witnessed the dataset consisted of 13 columns ('EPS', 'Liquidity', 'Profitability', 'Productivity', 'Leverage Ratio', 'Asset Turnover', 'Operational Margin', 'Return on Equity’, 'Market Book Ratio', 'Assets Growth', 'Sales Growth', 'Employee Growth', 'BK') and 92872 rows. To check our data validity; we needed to check if there are missing values in the table, check for uniqueness, check for normality, and checked for extreme outliers in the dataset that could distort the algorithm.

We then checked for missing values in our dataset and got to know there were 12 columns that had missing values. If the columns had more than 50% missing values, we would have dropped the columns but looking at the percentage, we didn’t need to drop any of the columns. Following this, we checked for unique values in the columns and looking at the dataset all columns had more than 2 distinctions therefore, there wasn’t a need for dropping any column.

Delete Rows which have more than 60% missing values. Out of 92872 rows, we deleted 64 rows that had more than 7 columns that had no value in them. By seeing the distributions, we can have an idea of how skewed these features are, we can also see further distributions of the other features.

**Missing Values**

When it comes to the missing values, we used KNNImputer with n\_neighbours = 5 as a result there were no missing values remaining in our dataset.

**Imbalanced Data**

One of the frequent issues with classification modelling is the existence of uneven class distribution (Elrahman and Abraham; 2013). This indicates that there is a significant disparity in the number of observations made by one class and the other class. Because most machine learning algorithms are built to function best when both classes are equally balanced, this is a difficult problem. The prediction model that would be created in the event of a class imbalance could prove to be biased and inaccurate.

Therefore, we applied SMOTE (Synthetic Minority Oversampling Technique). Smote is

one of the more commonly used oversampling techniques to deal with the imbalance problem. It helps to balance the class distribution by randomly increasing minority classes by replicating them. It further helps in:

* Location of the synthetic points: SMOTE picks the distance between the closest neighbors of the minority class, in between these distances it creates synthetic points.
* Final Effect: More information is retained since we didn't have to delete any rows unlike in random under sampling.
* Accuracy || Time Trade-off: Although it is likely that SMOTE will be more accurate than random under-sampling, it will take more time to train since no rows are eliminated as previously stated.

**Outliers**

We use RobustScaler to deal with the outliers. It scales features using statistics that are robust to outliers. This method removes the median and scales the data in the range between 1st the quartile and 3rd quartile. i.e., in between 25th quantile and 75th quantile range. This range is also called an Interquartile range.

The median and the interquartile range are then stored so that it could be used upon future data using the transform method. If outliers are present in the dataset, then the median and the interquartile range provide better results and outperform the sample mean and variance.

RobustScaler uses the interquartile range so that it is robust to outliers.

# Section 3: Modelling Development

This step in the machine learning process is regarded as crucial and necessary. The proposed models are to be put into practice after the data preparation procedure, which included feature selection and resampling. The effects of preprocessing methods on the various models can then be assessed and contrasted. Information on the model that was used, and this part discusses the model's operation.

* **Random Forest:**

An ensemble of several decision trees, known as a random forest model, is frequently applied to classification issues. It builds each tree using methods like feature randomness and bagging, resulting in an uncorrelated forest of trees (Khoshgoftaar, 2007). Every tree is supported by a distinct random sample.

This group of trees' prediction performance is more precise than that of any single tree. A few of the characteristics that make it a good fit for the selected dataset are the model's quick training speed, outlier resistance, and capacity for handling unbalanced data.

* **Naïve Bayes Classifier Algorithm:**

Based on the Bayes theorem, the Naive Bayes classifier categorises each value as independent of every other value. It enables us to forecast a class or category using probability utilising a given set of features.

Despite its simplicity, the classifier performs admirably and is frequently used because it outperforms more complex classification techniques.

* **Linear Support Vector Classifier (SVC):**

With a high number of data, the Linear Support Vector Classifier (SVC) approach performs well. It uses a linear kernel function to perform classification. When compared to the SVC model, the Linear SVC adds more parameters including the loss function and penalty normalization, which applies "L1" or "L2." Because linear SVC is based on the kernel linear technique, the kernel method cannot be modified.

**Implementation**

We chose three models to work with, which were Random Forest, Linear SVC and Naïve Bayes. We ran the three models and compared their outputs for the first round. The performance of models has been compared later in this literature. In the first round we only applied Standard Scaler to see how the model would look like if it is not optimized.

After finishing the first round, we applied the different data processing technique explained earlier to develop our models. Aside from the above-mentioned techniques applied we also used Hyperparameter tuning to further improve the output of our data.

**Hyperparameter tuning**

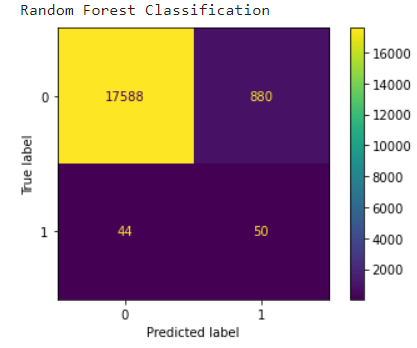
In this part, we tried to apply the best parameters to be used in each of the model to further improve our desired output. We used HalvingGridSearchCV for the hyperparameter tuning which we have set the numbers to be applied for the different parameter.

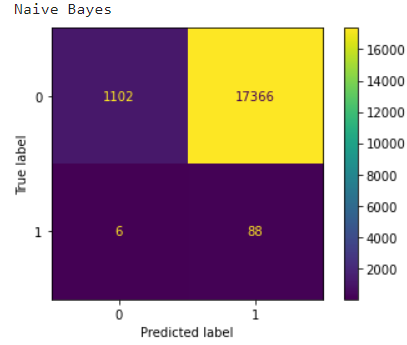
# Section 4: Model Comparison

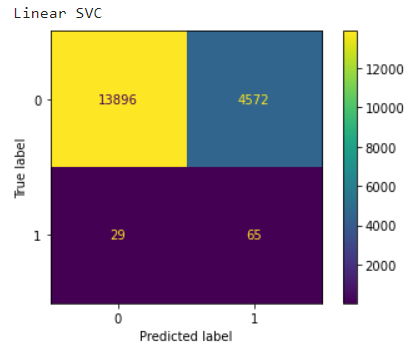
Before we go to the output of each model, let us first understand the concepts on how the different output would be valid for the scenario. The tools that are shown can help understand the precision, accuracy, recall, and F score of each model.

#### **Confusion Matrix:**

* Positive/Negative: Type of Class (BK) ["No", "Yes"] True/False: Correctly or Incorrectly classified by the model.
* True Negatives (Top-Left Square): This is the number of correctly classifications of the "No" (No Bankruptcy Detected) class
* False Negatives (Top-Right Square): This is the number of incorrectly classifications of the "No"(No Bankruptcy Detected) class.
* False Positives (Bottom-Left Square): This is the number of incorrectly classifications of the "Yes" (Bankruptcy Detected) class
* True Positives (Bottom-Right Square): This is the number of correctly classifications of the "Yes" (Bankruptcy Detected) class.







**Model Comparison (Round 1)**

In this model comparison, we did not try to normalize and clean the data with the appropriate method to verify the data. The data was only transformed using a standard scaler. We can see that due to the imbalanced data set the accuracy is too high because most of the predictors where mostly Not bankrupt and was not able to fully utilize the Bankrupt predictor.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** |
| **Random Forest** | **99%** | **0.50** | **0.00** |
| **Naïve Bayes** | **6%** | **0.01** | **0.95** |
| **Linear SVC** | **99%** | **0.0** | **0.0** |

**Model Comparison (Round 2):**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Accuracy** | **Precision** | **Recall** |
| **Random Forest** | **95%** | **.05** | **0.53** |
| **Naïve Bayes** | **6%** | **0.01** | **0.94** |
| **Linear SVC** | **75%** | **0.01** | **0.69** |

**Note:** Accuracy is a good measure of model performance if the dataset we are dealing with doesn’t have class imbalance whereas, if the classes are imbalanced, it is recommended to look for precision and recall.

**Evaluation**

We concluded that after our final data processing, the performance, particularly recall and precision, improved significantly for many if not all.

Furthermore, AUC provides an aggregate measure of performance across all possible classification thresholds.

For Random Forest:

The accuracy and precision kind of remained almost the same, however, the recall improved from 0.00 to 0.53.

Chart, line chart

Description automatically generated

For Naïve Bayes:

The accuracy and precision remained almost the same.

Chart, line chart

Description automatically generated

For Linear SVC:

The accuracy saw a downfall yet the precision and recall improved.

Chart, line chart

Description automatically generated

**Conclusion & Future Work**

To get the best forecast performance, many strategies have been used, as can be seen from the prior literature. The topic of bankruptcy prediction has garnered interest in recent decades. The difficult element, as mentioned, is choosing the best financial factors that play a significant role in a company's bankruptcy. The issue of class disparity is another barrier to this investigation.

In conclusion, we get to the result that among the three models compared,

* we can see that linear SVC is better at determining Bankruptcy (True Positives).
* We can see that Random Forest is better at determining No Bankruptcy (True Negatives).
* Naive Bayes is better suited for categorical input variables than numerical variables.

These methods can be investigated further using alternative classifiers and sets of financial attributes. In this study, we focused only on the dataset presented to us for our assignment. One may afterwards investigate bankruptcy information for any other dataset in the financial setting.

**Future Steps & Recommendations**

Additional techniques and tools to advance the performance of the model.

For future implementation we wanted to use XGBoost which provides better scaling, performance and being a tree-based classifier or ensemble, it is not influence by outliers. It is also very robust because it has the ability to perform well under diverse conditions. **XGBoost** stands for e**X**treme **G**radient **B**oostingand is an implementation of gradient-boosted decision trees designed for speed and performance. The library is laser-focused on computational speed and model performance, as such there are few frills. Nevertheless, it does offer a number of advanced features.

For improving model accuracy, it is important how you deal with class imbalance. Most Machine learning algorithms work best when the number of samples in each class is almost equal. Consequently, in such cases, there is a high probability of getting a high accuracy just by predicting the majority class, but the model fails to capture the minority class, which is most often the point of creating the model in the first place. The simplest implementation of over-sampling is to duplicate random records from the minority class, whereas under-sampling involves removing random observations from the majority class. However, these techniques also have their weaknesses. In simple terms, over-sampling can cause overfishing and under-sampling can cause loss of information.