CASE STUDY 4: PREDICTING BANKRUPTCY

BY: ANDREEA CAROLINA CRAUS

# 1 | INTRODUCTION

Predicting bankruptcy for economic decision-making is valuable as it impacts local communities, industry participants, investors, policymakers, and the global economy. The goal of this analysis is to assess a company’s financial condition and future prospects in the long-term and develop a predictive model that combines various econometric measures to foresee the financial condition of the company. The data was provided as five separate files for the yearly financial records for the company. Each file contained data for 64 attributes in the form of synthetic features such as different measures of net profit, working capital, equity, sales, any many other different combinations. Each instance of the data has a class label of either 0 (Bankrupt) or 1 (Non-Bankrupt), making this a binary classification problem. For this analysis, two of the most common and successful predictive models will be used based on the Random Forest and Gradient Boosting (XGBoost) algorithms.

# 2 | METHODS

## 2.1 DATA EXPLORATION & PREPROCESSING

The features provided in the data are called synthetic features, which are a combination of various econometric measures using arithmetic operations, such as addition, subtraction, multiplication, and division, developing more complex features. The first step is exploring if and how many missing values are present in the data. For example, the nullity matrix plot for Year 1 is shown in Figure 1, which provides a data-dense display to visualizing missing data patterns. The nullity matrices for all years were distributed similarity and as can be seen from the plot, Attribute 21 and Attribute 37 have a significant amount of missing values, which needs to be further explored. For the purpose of this analysis, these two features will be removed as they contain more than 50% missing values. In addition, the rows of the missing values for the other features will be removed. Furthermore, the correlation of features with missing values present is explored for each year. An example of the correlation heatmap for Year 1 can be seen in Figure 2. With many of the features containing missing values being heavily correlated, eliminating the missing values could skew the results of the analysis, therefore inputing the missing values could improve the model. Next, the class distribution for each label is explored to determine if action needs to be taken to deal with data imbalance. As can be seen in Figure 3, the distribution of labels for Bankrupt or Non-Bankrupt classes is extremely imbalanced, with many more values for the Bankrupt class. The imbalanced data will need to undergo oversampling or undersampling.

A purple and yellow chart

Description automatically generated

Figure : Nullity Matrix for Year 1

A diagram of numbers and a triangle

Description automatically generated with medium confidence

Figure : Correlation Heatmap for Year 1

A group of blue bars

Description automatically generated

Figure : Class Label Distributions for Each Year

## 2.2 SYNTHETIC MINORITY OVERSAMPLING TECHNIQUE (SMOTE)

To handle the issue of the class imbalance in the data, the SMOTE technique was used which specifically targets the minority class, Non-Bankrupt in this case, and aims to balance the class distribution by oversampling. For each minority class instance, SMOTE generates synthetic instances by interpolating between that instance and its nearest neighbors in the feature space. This process is important as imbalanced data can lead to biased models that perform poorly on the minority class.

## 2.3 RANDOM FOREST

Random Forest is an ensemble learning method that combines the predictions of multiple decision trees to improve overall predictive performance and robustness. It employs a technique called Bootstrap Aggregating, or Bagging, which creates multiple subsets (bootstrap samples) of the training set by randomly sampling with replacement. Each subset is used to train an individual decision tree. For each decision tree in the forest, a random subset of features is considered at each split point. This introduces diversity among the trees, preventing them from becoming highly correlated and improving the overall performance of the ensemble. The tree is built by recursively partitioning the data based on the chosen features until a certain stopping criterion is met, such as reaching a maximum depth or minimum number of samples per leaf. For classification tasks, each tree in the forest "votes" for a class, and the class with the majority of votes becomes the predicted class. For regression tasks, the predictions from each tree are averaged to obtain the final prediction. The final output of the Random Forest is the aggregation of the individual tree outputs. This ensemble approach often results in more robust and accurate predictions compared to individual decision trees, as it mitigates overfitting and reduces sensitivity to noise in the data. The following hyperparameters were tuned for the Random Forest Model to attain the highest accuracy:

* n\_estimators: The number of trees.
* max\_depth: The maximum depth of the trees.
* min\_samples\_split: The minimum number of samples required to split an internal node.
* max\_features: The number of features to consider when looking at the best split
* Bootstrap: Whether bootstrap samples are used when building trees.

## 2.4 XGBOOST (XTREME GRADIENT BOOSTING)

XGBoost is a powerful and efficient machine learning algorithm used for both regression and classification tasks. It belongs to a family of gradient boosting algorithms, which sequentially combine weak learners, typically decision trees, to create a strong predictive model that has become popular due to its speed, scalability, and high performance in machine learning tasks. XGBoost builds an ensemble of decision trees, or the weak learners, and each tree is trained sequentially, with the upcoming trees trying to correct errors made by the previous trees in an effort to minimize the objective function which includes a loss function , which quantifies the different between predicted and actual values, and a regularization term, which helps control the complexity of the model to prevent overfitting. It uses a gradient boosting framework to optimize the objective function and the gradient of the loss function is computed to determine the direction in which the model should update its predictions. The learning rate is a parameter that control the contribution of each tree to the final prediction. The following hyperparameters were tuned for the XGBoost model to attain the highest accuracy:

* learning\_rate: Controls the contribution of each tree to the final prediction.
* subsample: The fraction of the training dataset for growing each tree, smaller values introduce randomness and can prevent overfitting.
* max\_depth: Represents the maximum depth of a tree and controls the complexity of the individual trees in the ensemble.
* n\_estimators: The number of trees

# 3 | RESULTS

## 3.1 XGBOOST RESULTS

After hyperparameter tuning, the values for the parameters chosen to use in the model can be seen in Table 1. A confusion matrix on the tuned model is shown in Figure 4 which displays the correctly labeled predicted values against the actual values.

|  |  |  |  |
| --- | --- | --- | --- |
| learning\_rate | subsample | max\_depth | n\_estimators |
| 0.1 | 0.6 | 9 | 50 |

Table 1: XGBoost Model HyperParameter Tuning Results

# A blue squares with white text Description automatically generated

Figure : Confusion Matrix for XGBoost Model

## 3.2 RANDOM FOREST RESULTS

After hyperparameter tuning, the values for the parameters chosen to use in the model can be seen in Table 2. A confusion matrix on the tuned model is shown in Figure 5 which displays the correctly labeled predicted values against the actual values.

|  |  |  |  |
| --- | --- | --- | --- |
| bootstrap | max\_depth | max\_features | n\_estimators |
| 0.1 | 0.6 | 9 | 200 |

Table 2: Random Forest Model HyperParameter Tuning Results

A blue squares with white text

Description automatically generated

Figure : Confusion Matrix for Random Forest Model

# 4 | CONCLUSION

While the Random Forest model had a very slightly better accuracy score, the confusion matrices for both models and the precision and recall scores all show evidence that the XGBoost model performed better. In Table 3, the evaluation metrics for the two models can be seen including accuracy, precision, and recall. The precision and recall for the Random Forest model were both 0% and none of the Non-Bankrupt class labels were predicted correctly. While only three of the Non-Bankrupt class labels were predicted correctly in the XGBoost model, this is due to the imbalanced data and further hyper-parameter tuning could improve these results.

|  |  |  |
| --- | --- | --- |
| **Metric** | **XGBoost** | **Random Forest** |
| Accuracy | 98.02% | 98.34% |
| Precision | 23.08% | 0% |
| Recall | 16.67% | 0% |

Table 6: Evaluation Metrics for

In conclusion, the XGBoost model overall performed better on predicting the classification of Non-Bankrupt or Bankrupt companies with an accuracy of 98.02% likely due to its use of regularization techniques such as L1 and L2 regularization, which help prevent overfitting. On the other hand, Random Forest often relies on bootstrapping and feature randomness for diversity. In addition, XGBoost implements gradient boosting which builds trees sequentially, with each tree correcting the errors of the previous ones.

# 5 | CODE

Relevant code is attached in CarolinaCraus\_CS4.ipynb