Corporate Bankruptcy Prediction on Imbalanced Financial Data

Matthieu Hanna Gerguis Renaud de l'Epine Ilian Segoin

Academic year 2025–2026

Programme:	A4 IF3
Course:	Machine Learning
Instructor:	**Walid KHERIJI**
Institution:	**ESILV**
Project title:	Corporate Bankruptcy Prediction on Imbalanced Financial Data
Dataset:	Company Bankruptcy Prediction (Kaggle)

Contents

Abstract

Corporate bankruptcy prediction is a key problem in risk management, as the cost of failing to identify distressed firms is typically much higher than issuing false alarms. In this project, we study bankruptcy prediction as a highly imbalanced binary classification task on a real-world financial dataset derived from the Taiwan Economic Journal (TEJ), made available on Kaggle. We construct a complete pipeline including data cleaning, feature selection, principal component analysis (PCA), baseline models, imbalance-handling strategies (undersampling, random oversampling, SMOTE), ensemble methods, and deep neural networks. Our analysis focuses on maximizing recall for the minority class (bankrupt firms) while preserving interpretability and robustness. We show that *Logistic Regression combined with random oversampling without PCA* provides an excellent compromise between sensitivity, AUC, stability, and simplicity, achieving a recall on bankruptcies of approximately 0.86, an accuracy of about 0.88, and an AUC-ROC around 0.93 on the held-out test set.

1 Introduction

Corporate bankruptcy prediction has been an active research topic in finance and risk management for several decades. Early approaches relied on linear discriminant analysis and simple financial ratios, such as Altman's Z-score model [1] and Ohlson's logit-based model [2]. More recently, the availability of large-scale financial datasets and advances in machine learning have motivated the use of more flexible models, including tree-based ensembles, kernel methods, and deep neural networks [4, 5, 6, 8].

In this project, we address the problem of predicting corporate bankruptcy on a real-world dataset that is both *high dimensional* and *strongly imbalanced*. The data originate from the Taiwan Economic Journal (TEJ) and cover financial statements of Taiwanese firms between 1999 and 2009. A cleaned version of the dataset is available on Kaggle under the name *Company Bankruptcy Prediction* [19], which we use as the basis for our experiments. The dataset contains 6 819 firms, 95 financial features (various profitability, liquidity, leverage, cash-flow, turnover and growth ratios), and a binary label `Bankrupt?` indicating whether the firm went bankrupt in the subsequent period. The minority class (bankrupt firms) accounts for roughly 3% of observations, which creates a highly imbalanced classification setting.

1.1 Problem statement

We formulate bankruptcy prediction as a supervised binary classification problem. Given input vectors $x \in \mathbb{R}^d$ representing financial ratios and a target label $y \in \{0, 1\}$ ($y = 1$ for bankrupt, $y = 0$ otherwise), the goal is to learn a classifier $f : \mathbb{R}^d \to \{0, 1\}$ that predicts y from x. Because bankruptcies are rare but very costly when missed, we prioritize *recall on the minority class* (also called sensitivity or true positive rate for bankrupt firms) over global accuracy.

Formally, recall on class 1 (bankrupt) is defined as

$$\text{Recall}_1 = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{1}$$

where TP denotes true positives (correctly predicted bankruptcies) and FN false negatives (missed bankruptcies). In contrast, accuracy is defined as

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{2}$$

which can be misleading in highly imbalanced settings.

1.2 Contributions

The main contributions of this project are as follows:

- We build a reproducible data processing pipeline, including zero-pattern analysis, variance-based and correlation-based feature selection, and a dynamic per-feature variance threshold.
- We systematically compare baseline models (Logistic Regression, Naïve Bayes, Random Forest) with and without PCA, showing that PCA can help some linear models but may harm tree-based ensembles.
- We conduct an extensive study of imbalance-handling techniques applied to Logistic Regression: no resampling, undersampling, random oversampling, and SMOTE, each with and without PCA, quantified in terms of recall and AUC.
- We evaluate ensemble methods (Random Forest, AdaBoost, Bagging, XGBoost) and deep neural networks, including hyperparameter optimization for a feed-forward network using Keras Tuner.
- We identify *Logistic Regression with random oversampling and no PCA* as the most suitable final model, balancing high recall, good AUC, low complexity, and interpretability.

The rest of the report is organized as follows. Section 2 reviews related work. Section 3 describes the data and preprocessing steps. Section 4 details the models, loss functions, and metrics. Section 5 presents the experimental protocol. Section 6 discusses the results. Section 7 concludes and outlines future work.

2 Related Work

2.1 Classical bankruptcy prediction models

The literature on bankruptcy prediction goes back at least to the 1960s. Altman proposed the famous Z-score model based on multiple discriminant analysis and a linear combination of financial ratios to separate bankrupt from non-bankrupt firms [1]. Ohlson later introduced a logit model using financial ratios and indicator variables to compute bankruptcy probabilities [2]. These models remain popular in practice due to their simplicity and interpretability.

2.2 Machine learning approaches to bankruptcy prediction

The last two decades have seen a systematic adoption of machine learning techniques for bankruptcy prediction. Surveys such as Devi and Radhika [3] and Dasilas and Rigani [4] review statistical and machine learning methods, including decision trees, support vector machines (SVM), artificial neural networks (ANN), and ensemble methods. More recent studies have evaluated advanced tree-based ensembles and gradient boosting methods, showing strong predictive performance on corporate failure datasets [5, 8, 6, 7]. Other works investigate automated machine learning or specialized architectures for bankruptcy risk [20, 21].

Machine learning techniques are also applied to related problems such as bank failure prediction and insolvency in insurance companies [9, 10]. Although promising, these models must be carefully evaluated under class imbalance and cost-sensitive criteria to avoid misleadingly high accuracy.

2.3 Imbalanced learning and sampling techniques

Imbalanced classification has motivated a rich line of research on algorithmic modifications and data-level strategies. One of the most influential techniques is SMOTE (Synthetic Minority Over-sampling Technique) [11], which generates synthetic minority examples using interpolation in feature space. Many variants and comparative studies of imbalanced learning algorithms exist in the literature.

In our work, we focus on *data-level* rebalancing methods—random undersampling, random oversampling, and SMOTE—combined with relatively simple classifiers, in line with recommendations from empirical studies on credit and bankruptcy risk [5, 6, 8].

2.4 Modeling and evaluation foundations

From a methodological perspective, our model choices and evaluation procedures are grounded in standard machine learning textbooks and tools. We rely on logistic regression, linear models, and neural networks as described in Bishop [12], Hastie et al. [13], and Murphy [14]. Tree-based ensembles and gradient boosting are implemented using scikit-learn and XGBoost [16, 15], while multi-layer perceptrons follow the classical supervised neural network formulations [17]. We use scikit-learn's implementation as the main software framework for supervised learning and model selection [18].

3 Data

3.1 Source and structure

The dataset used in this project is the *Company Bankruptcy Prediction* dataset published on Kaggle by F. Soriano [19]. It is derived from financial statements collected by the Taiwan Economic Journal (TEJ) between 1999 and 2009 and labeled according to the Taiwan Stock Exchange bankruptcy rules.

The dataset contains:

- 6 819 firms (rows),
- 95 financial features (columns) including profitability, liquidity, leverage, activity, and growth indicators,
- 1 binary target variable `Bankrupt?` with values 0 (non-bankrupt) and 1 (bankrupt).

All features are numerical (`float64` or `int64`). The Kaggle version we use is already cleaned with no missing values. We further verified that all columns are numerical and that the data types are consistent.

The original Kaggle page is available at:

https://www.kaggle.com/datasets/fedesoriano/company-bankruptcy-prediction

3.2 Initial exploratory analysis

We first performed a descriptive analysis of the dataset:

- We inspected data types and confirmed that all variables are numeric.
- We computed descriptive statistics (mean, standard deviation, quartiles, min, max) for all features.
- We examined the distribution of the target variable and found that only about 3% of firms are labeled bankrupt, confirming a strong class imbalance.

- We counted the number of zero entries per column and per row. Most columns contain at least one zero; a small subset has many zeros. At the row level, some firms exhibit an unusually high number of zero-valued ratios.

3.3 Cleaning zero-heavy rows

Rows with a very large number of zero values may correspond to atypical or low-information firms (e.g., missing business activity or extreme accounting patterns). We computed, for each row, the number of zero-valued features and studied how many rows exceed a given threshold. We observed that:

- all rows contain at least one zero,
- the maximum number of zeros in a row is 19,
- rows with more than 6–7 zeros are rare.

Figure 1 illustrates how the number of rows decreases as we increase the minimum number of zero-valued features required per row. This confirms that rows with more than 6–7 zeros are outliers.

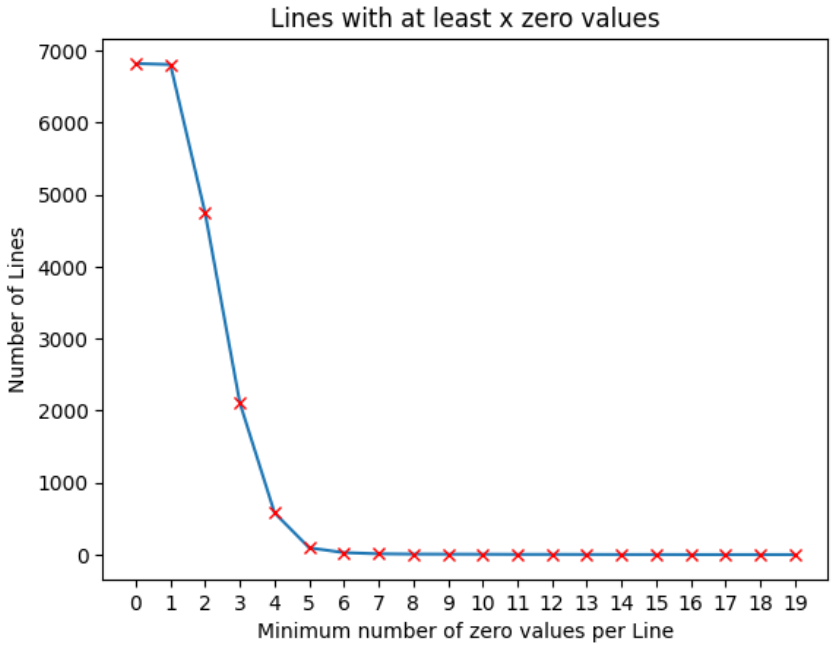

Figure 1: Number of rows with at least x zero-valued features. Most firms have only a few zero-valued ratios; rows with many zeros are rare.

To quantify the effect of removing zero-heavy rows, we also tracked the remaining number of firms as a function of the zero threshold (Figure 2).

We therefore removed rows with 7 or more zero-valued features, retaining most of the dataset while discarding a small number of extreme outliers. The resulting cleaned dataset is denoted as $\mathcal{D}_{\text{trim}}$.

3.4 Variance analysis and feature selection

We computed the empirical variance of each feature on $\mathcal{D}_{\text{trim}}$ and observed that many variables exhibit extremely low variance, close to constant values. Figure 3 shows the raw distribution

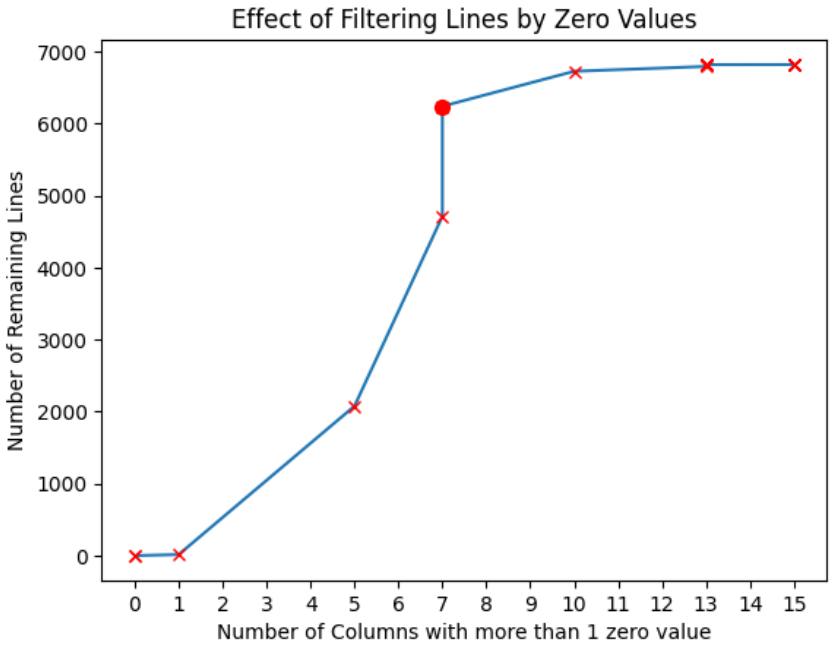

Figure 2: Effect of filtering rows by zero count. The highlighted point corresponds to the chosen threshold of 7 zeros per row, which removes only a small fraction of the dataset.

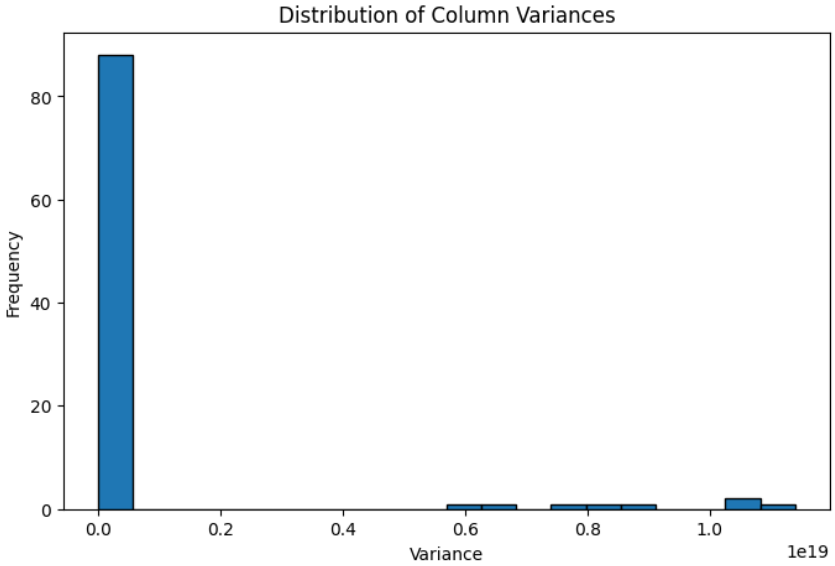

Figure 3: Distribution of column variances. A few features have extremely large variance, while many others have very small variance.

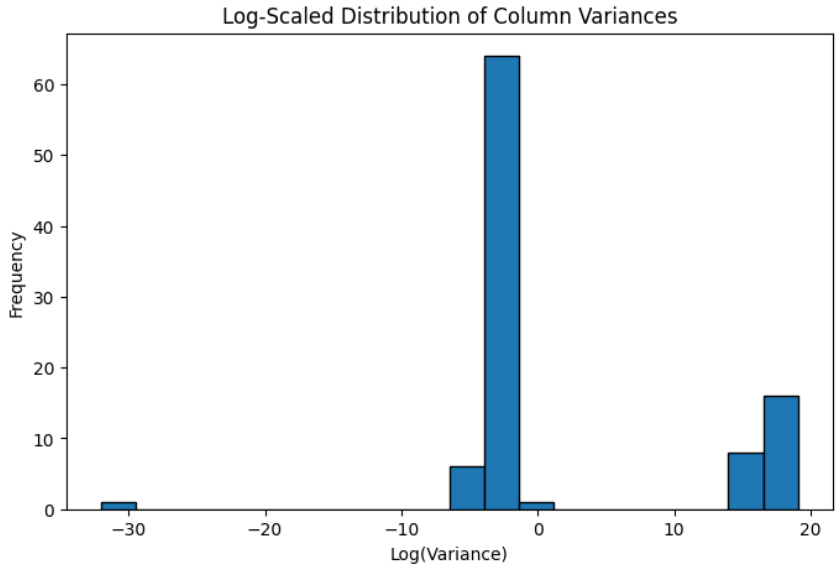

Figure 4: Log-scaled distribution of column variances. The logarithmic scale reveals the presence of several groups of features with similar variance levels.

of variances across all columns, while Figure 4 displays the same information on a logarithmic scale to better highlight very small and very large variances.

To avoid including uninformative features, we explored several variance-based selection strategies:

- A fixed variance threshold of 0.01, which reduced the feature space to 32 variables.
- A lower threshold of 0.001, retaining about 52 variables.
- A *dynamic* per-feature threshold of the form

$$\tau_j = \max\left(\tau_0, \alpha \cdot \bar{x}_j\right),$$

  where $\bar{x}_j$ is the mean of feature j, τ_0 is a base threshold (e.g. 0.0001), and α is a scaling factor (e.g. 0.001).

The dynamic threshold is less aggressive than a fixed global threshold and preserves more features with moderate variance, which is beneficial for models that exploit interactions between variables.

We also computed the correlation matrix and applied correlation filtering:

- We first removed low-variance features.
- We then identified feature pairs with high absolute Pearson correlation (e.g. $|\rho| > 0.7$) and removed features involved in many redundant relationships.
- We also considered dropping features with very low absolute correlation with the target ($|\rho_{j,\text{target}}| < 0.05$).

Figure 5 shows the full correlation matrix of the original feature set, highlighting blocks of highly correlated ratios. After applying variance and correlation filtering, the correlation structure among the selected features becomes sparser, as illustrated in Figure 6.

The final cleaned feature set used for the main experiments (denoted $\mathcal{D}_{\text{clean}}$) is based primarily on the dynamic variance threshold and conservative correlation filtering, which strikes a balance between dimensionality reduction and information preservation.

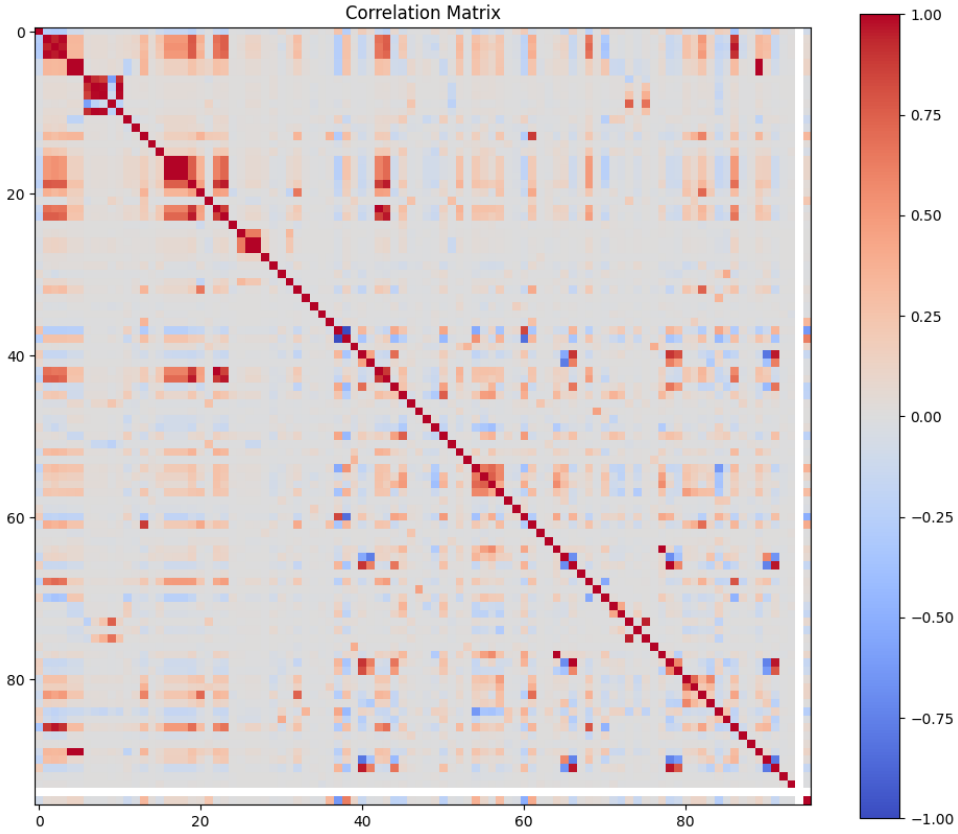

Figure 5: Correlation matrix of the original feature set. Many features exhibit strong positive or negative correlations, motivating correlation-based feature selection.

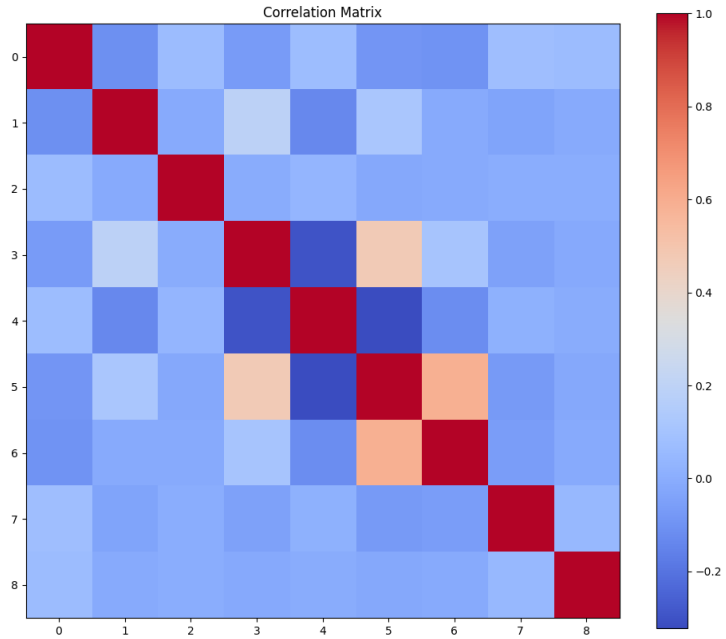

Figure 6: Correlation matrix of a subset of selected features after variance and correlation filtering. Redundant correlations are reduced while informative structure is preserved.

3.5 Standardization and PCA

For models that require standardized inputs (especially Logistic Regression and Naïve Bayes), we applied standardization:

$$x_{ij}^{\text{std}} = \frac{x_{ij} - \mu_j}{\sigma_j}, \tag{3}$$

where μ_j and σ_j are the empirical mean and standard deviation of feature j in the training set.

We also built an alternative representation using Principal Component Analysis (PCA) [13], keeping enough components to explain 95% of the total variance. This resulted in a reduced-dimensional dataset X_{PCA} that we used to compare performance with and without dimensionality reduction.

3.6 Train/validation/test splits

We split the data into training and test sets using a 70/30 split, stratified by the target label to preserve the global class distribution in both sets. Formally:

- 70% of the data are used for training and hyperparameter tuning;
- 30% are held out as a test set for final evaluation.

Within the training set, we performed cross-validation (typically 5-fold) for hyperparameter optimization. The test set remained untouched until the final evaluation phase to ensure an unbiased assessment of generalization performance.

4 Methodology

4.1 Problem formulation

Let $\{(x_i, y_i)\}_{i=1}^n$ denote the dataset, with $x_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$. Our goal is to learn a classifier f_θ parameterized by θ that approximates the conditional probability $P(Y = 1 \mid X = x)$ and yields binary predictions:

$$\hat{y}_i = \mathbf{1}\left[f_\theta(x_i) \geq \tau\right],$$

where τ is a decision threshold (default $\tau = 0.5$ unless otherwise stated).

4.2 Logistic Regression

Logistic Regression (LR) is our main baseline model and final choice. Given an input x, the model computes a linear score

$$z = w^\top x + b, \tag{4}$$

and maps it to a probability using the logistic function

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{5}$$

The predicted bankruptcy probability is therefore

$$\hat{p}(y = 1 \mid x) = \sigma(w^\top x + b). \tag{6}$$

The parameters (w, b) are estimated by minimizing the regularized negative log-likelihood (cross-entropy loss):

$$\mathcal{L}(w, b) = -\sum_{i=1}^{n} \left[y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)\right] + \lambda \|w\|_2^2, \tag{7}$$

where $\hat{p}_i = \hat{p}(y = 1 \mid x_i)$ and $\lambda > 0$ is an ℓ_2 regularization parameter related to the hyperparameter C in scikit-learn [16, 14].

4.3 Other baseline models

To establish baselines, we also considered:

- **Gaussian Naïve Bayes (GNB)**: a generative classifier that assumes conditional independence among features and Gaussian class-conditional distributions [12].
- **Random Forest (RF)**: an ensemble of decision trees trained on bootstrap samples with random feature subsets; RF is known to handle non-linear interactions well [13].

All models are implemented using scikit-learn [16].

4.4 Resampling strategies for class imbalance

Given the strong class imbalance (about 3% bankruptcies), we studied several data-level resampling strategies applied to the training set:

- **No resampling**: training directly on the imbalanced data.
- **Random undersampling**: randomly removing majority class examples until both classes are balanced.
- **Random oversampling**: randomly duplicating minority class examples until balance is reached.
- **SMOTE**: Synthetic Minority Over-sampling Technique [11], which generates new minority examples by interpolating between nearest neighbours in feature space.

Let n_0 and n_1 denote the number of majority and minority examples, respectively, in the training set. Random oversampling yields a new training set with size $2n_0$ (balanced), whereas random undersampling yields a training set with size $2n_1$. SMOTE also yields a balanced dataset but with synthetic minority samples.

4.5 Ensemble methods

In Stage 3 we evaluated several ensemble methods:

- **Random Forest** (RF) as a bagging approach for decision trees.
- **Bagging** with Logistic Regression as base estimator.
- **AdaBoost** with Logistic Regression as weak learner, focusing on misclassified examples.
- **XGBoost** [15], a gradient boosting framework that optimizes a regularized objective over decision trees.

We used standard implementations (scikit-learn for RF, AdaBoost, Bagging; XGBoost library for XGB).

4.6 Deep Neural Networks

We implemented a feed-forward neural network (multi-layer perceptron) with one or several hidden layers. In the simplest case (without PCA), the architecture is:

$$h^{(1)} = \phi(W^{(1)}x + b^{(1)}), \tag{8}$$

$$h^{(2)} = \phi(W^{(2)}h^{(1)} + b^{(2)}), \tag{9}$$

$$\hat{p}(y = 1 \mid x) = \sigma(w^{\top}h^{(2)} + b), \tag{10}$$

where $\phi(\cdot)$ is a non-linear activation function (ReLU or tanh), and σ is the logistic function. The network is trained by minimizing binary cross-entropy with class weights to penalize misclassification of bankrupt firms. We used TensorFlow/Keras with early stopping on validation loss and Keras Tuner for hyperparameter search.

4.7 Evaluation metrics

We use the following metrics:

- **Accuracy**: overall proportion of correctly classified observations.
- **Precision (class 1)**:
$$\text{Precision}_1 = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$
- **Recall (class 1)**: as defined in (1).
- **F1-score (class 1)**: harmonic mean of precision and recall.
- **ROC curve**: plots true positive rate vs. false positive rate at varying thresholds.
- **AUC-ROC**: area under the ROC curve, measuring the ranking quality of the classifier.

Because the dataset is highly imbalanced and our main interest lies in not missing bankruptcies, we treat Recall_1 and AUC-ROC as the primary metrics.

5 Experiments

5.1 Overview

We structured the experiments in three stages:

- **Stage 1: Baseline models.** We trained LR, GNB, and RF on cleaned data, with and without PCA, without any resampling. The goal was to understand the impact of model choice and PCA under strong imbalance.
- **Stage 2: Imbalance handling.** We fixed the model to Logistic Regression and compared four resampling strategies (none, undersampling, random oversampling, SMOTE), again with and without PCA. We also performed hyperparameter tuning for LR under each scenario using cross-validated AUC-ROC.
- **Stage 3: Advanced models.** We evaluated ensemble methods (RF, Bagging, AdaBoost, XGBoost) and feed-forward DNNs. For neural networks, we combined SMOTE, class weights, early stopping, and hyperparameter search.

5.2 Implementation details

All experiments were implemented in Python using:

- `numpy` and `pandas` for data manipulation,
- `scikit-learn` for classical ML models, preprocessing, and model selection [16, 18],
- `imblearn` for SMOTE and resampling utilities,
- `xgboost` for gradient boosted trees [15],
- `tensorflow/keras` and `keras-tuner` for neural networks.

We adopted the following general settings:

- Train/test split: 70/30, stratified by the target label.
- Cross-validation: 5-fold stratified CV for hyperparameter tuning.
- Logistic Regression: maximum iterations set to 1000; solvers `liblinear` or `saga` depending on the scenario.
- Random Forest: moderate depth and number of trees (e.g. 100 trees, depth up to 10) in baseline; tuned models allowed larger trees.
- RandomizedSearchCV or GridSearchCV used where exhaustive search was too costly.

We report approximate performance values to reflect the general trends; exact results depend on random seeds and train/test splits.

5.3 Stage 1: Baseline models

In Stage 1 we compared:

- Logistic Regression (LR),
- Gaussian Naïve Bayes (GNB),
- Random Forest (RF).

Each model was trained (i) on standardized raw features, and (ii) on PCA-transformed features (95% variance retained). No resampling was applied; the training data remained imbalanced.

Figure 7 reports the ROC curves of the three baseline models without PCA. Random Forest achieves the highest AUC, followed closely by Logistic Regression, while Naïve Bayes performs significantly worse.

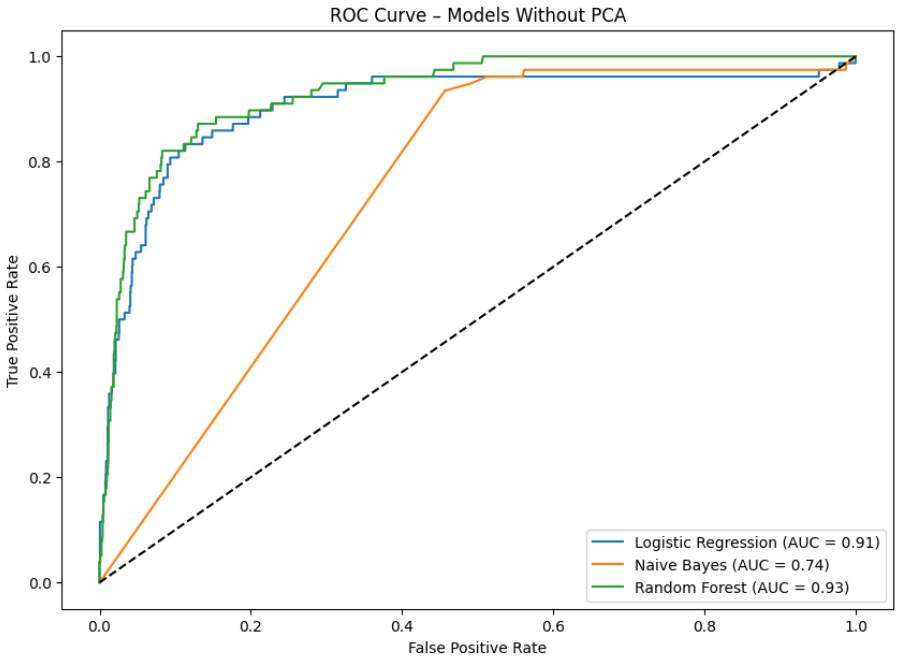

Figure 7: ROC curves of baseline models (Logistic Regression, Naïve Bayes, Random Forest) trained without PCA.

The main observations are:

- LR achieved strong AUC (around 0.91 without PCA, 0.93 with PCA), but recall for bankruptcies was very low (around 0.2–0.3).
- GNB obtained high accuracy but almost zero recall on the minority class, rendering it unsuitable for our objective.
- RF achieved high AUC (up to 0.93 without PCA), but tended to predict almost all firms as non-bankrupt, leading to recall close to zero on the minority class in some configurations.

These results confirm that accuracy and AUC alone are insufficient in imbalanced settings: models may appear strong globally while failing to detect bankruptcies.

5.4 Stage 2: Resampling with Logistic Regression

Stage 2 focuses on Logistic Regression and compares four resampling strategies:

- No resampling (original),
- Random undersampling,
- Random oversampling,
- SMOTE.

We evaluated each strategy with and without PCA and recorded recall on the minority class. Table 1 summarizes the approximate recall values observed for class 1 (bankrupt firms).

Table 1: Approximate recall on bankruptcies (class 1) for Logistic Regression under different resampling strategies.

Strategy	PCA	Recall (class 1)
Original (imbalanced)	Yes	≈ 0.19
Original (imbalanced)	No	≈ 0.28
Undersampling	Yes	≈ 0.45
Undersampling	No	≈ 0.86
Random oversampling	Yes	≈ 0.85
Random oversampling	No	≈ 0.86
SMOTE	Yes	≈ 0.85
SMOTE	No	≈ 0.80

Key findings:

- All resampling methods substantially improve recall compared to the original imbalanced training.
- Random undersampling without PCA reaches the highest recall (≈ 0.86), but at the cost of discarding a large number of majority-class examples and producing many false positives.
- Random oversampling and SMOTE also achieve high recall (0.80–0.86) while preserving all original data; SMOTE often yields the best AUC-ROC.
- PCA tends to slightly reduce recall in most resampling strategies, except in some undersampling scenarios where it stabilizes the model.

To better understand the effect of PCA on discriminative ability, we also plotted ROC curves for the original model and for the resampling strategies. Figure 8 compares the original Logistic Regression model with and without PCA; Figures 9 and 10 show the corresponding curves for random oversampling and SMOTE.

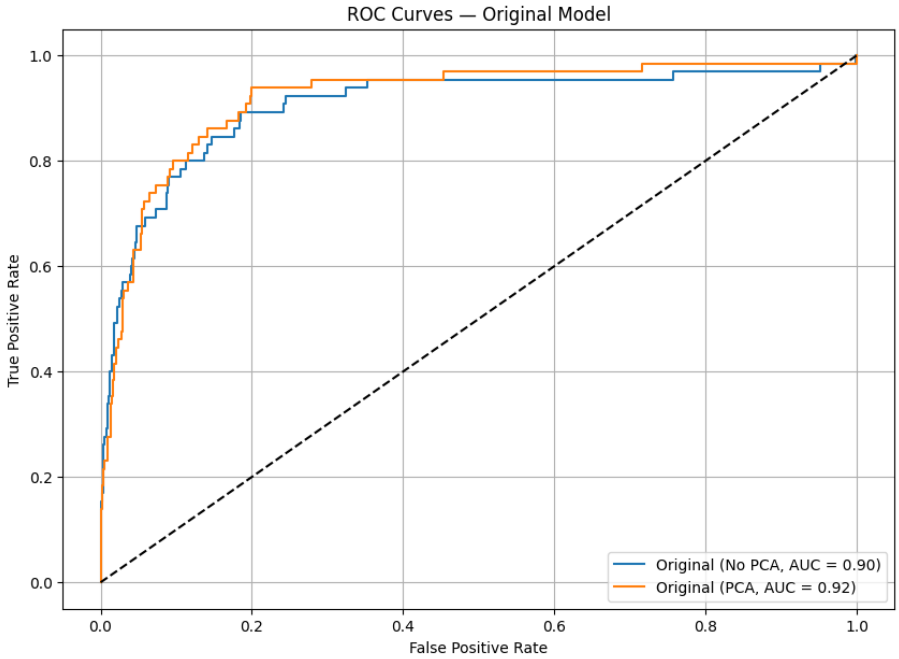

Figure 8: ROC curves of the original Logistic Regression model with and without PCA. PCA slightly improves AUC while leaving overall shape similar.

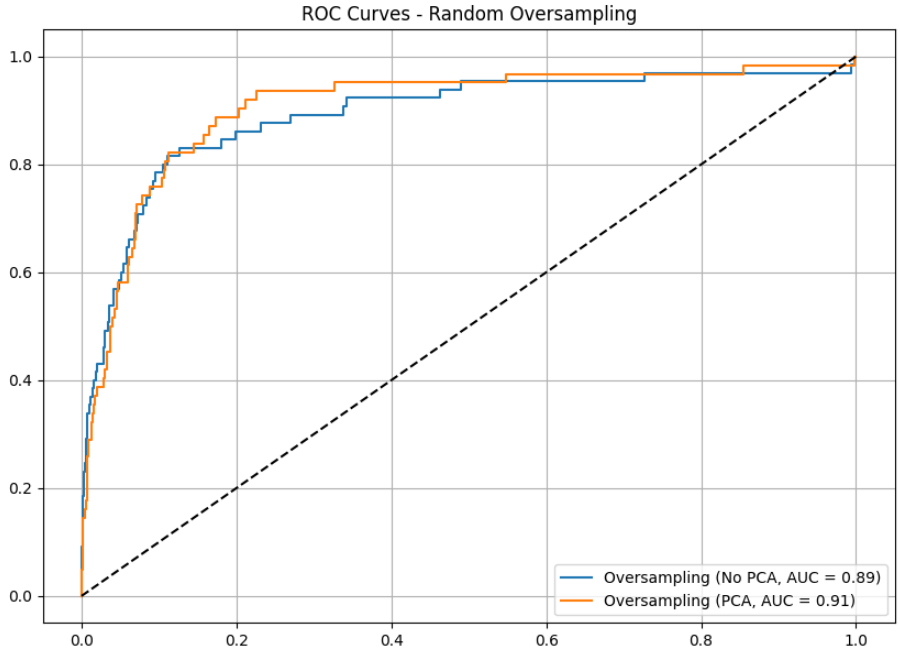

Figure 9: ROC curves of Logistic Regression with random oversampling, with and without PCA. The PCA version yields a marginally higher AUC.

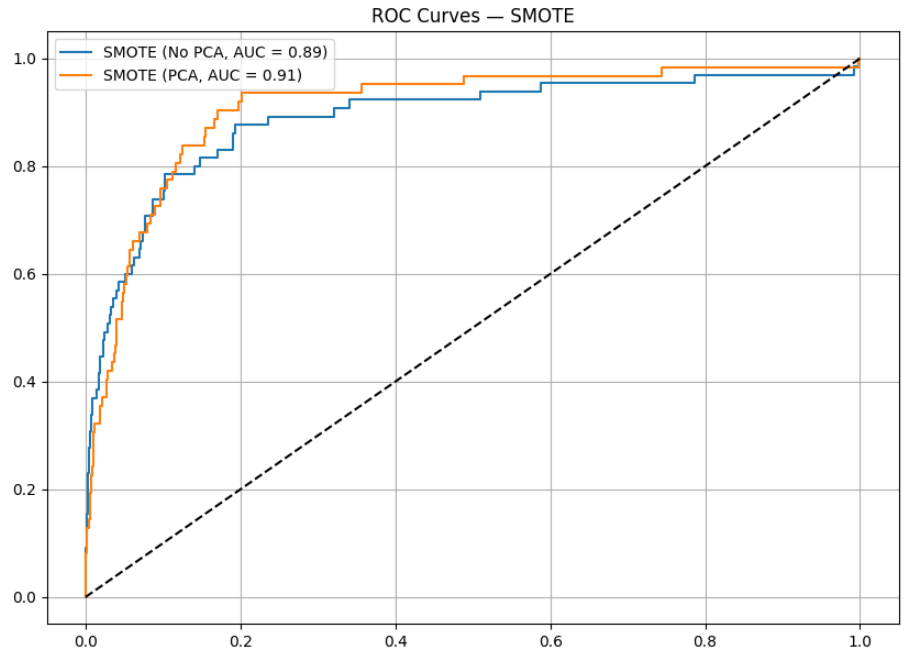

Figure 10: ROC curves of Logistic Regression with SMOTE, with and without PCA. SMOTE substantially improves sensitivity, and PCA slightly increases AUC.

5.5 Stage 2: Hyperparameter tuning for Logistic Regression

We then optimized the regularization parameter C and solver choice for Logistic Regression under each resampling strategy using 5-fold cross-validated AUC-ROC as the objective. The main outcomes are:

- On the original (imbalanced) data, a small C (strong regularization) performs best, with AUC around 0.93.
- Under undersampling, overall performance remains weaker (AUC ≈ 0.64), reflecting information loss from removing majority examples.
- With random oversampling and SMOTE, larger values of C (e.g. $C = 100$) are beneficial, and the best AUC is obtained with SMOTE without PCA (AUC ≈ 0.96).

Although SMOTE without PCA yields the highest AUC in this stage, practical considerations about interpretability and reliance on synthetic data lead us to also consider random oversampling as a strong and simpler alternative.

5.6 Stage 3: Ensemble models and DNNs

In Stage 3, we evaluated the following models on a randomly oversampled training set without PCA:

- Logistic Regression (LR),
- Random Forest (RF),
- AdaBoost (with LR as base estimator),
- Bagging (with LR as base estimator),
- XGBoost (XGB).

All models were evaluated on the original imbalanced test set. Approximate results are summarized in Table 2.

Table 2: Approximate performance of ensemble models trained on oversampled data (test set).

Model	Accuracy	Recall (class 1)	Comment
Logistic Regression	≈ 0.88	≈ 0.86	Simple, stable
Bagging (LR)	≈ 0.88	≈ 0.86	Similar to LR
AdaBoost (LR)	≈ 0.88	≈ 0.86	Balanced trade-off
Random Forest	≈ 0.96	≈ 0.25	High accuracy, low recall
XGBoost	≈ 0.96	≈ 0.35	Better recall than RF, still low

RF and XGB achieve very high accuracy but relatively low recall on bankruptcies, indicating that they prioritize the majority class. AdaBoost and Bagging provide performance comparable to LR in this configuration, without clear gains in recall or AUC.

For deep neural networks, we trained feed-forward networks on SMOTE-balanced data, with and without PCA, using class weights and early stopping. The best DNN configuration without PCA reaches:

- accuracy ≈ 0.95,
- recall on class 1 ≈ 0.63,
- AUC-ROC ≈ 0.94.

Although this DNN performs well in terms of AUC, its recall on bankruptcies remains below that of LR with random oversampling, and it is more complex and less interpretable.

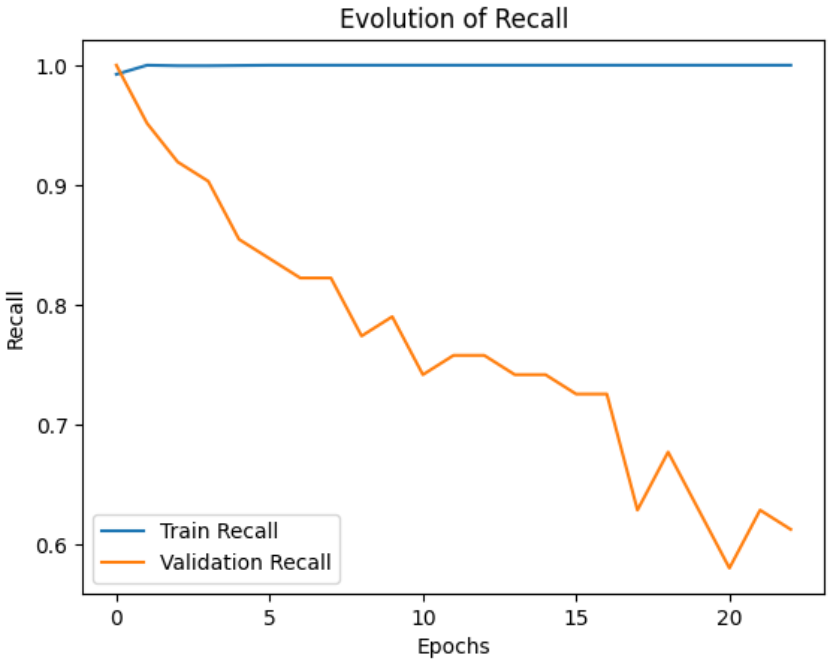

Figure 11: Evolution of training and validation recall for the tuned deep neural network. The divergence between the curves highlights overfitting: training recall remains near 1.0 while validation recall progressively decreases.

Figure 11 illustrates the evolution of recall during training for the tuned deep neural network. The training recall (blue curve) quickly reaches almost 1.0 and stays flat, while the validation

recall (orange curve) peaks after only a few epochs and then steadily decreases. This pattern is a clear indication of overfitting: the network continues to improve on the training set, but its ability to generalize to unseen data deteriorates as training progresses. Despite the use of early stopping and class weights, the model starts to memorize the oversampled SMOTE data instead of learning robust patterns related to bankruptcy risk.

6 Results and Discussion

6.1 Comparison of main approaches

Table 3 summarizes the performance of selected strategies on the test set.

Table 3: Summary of selected models on the test set (approximate values).

Model / Strategy	Resampling	PCA	Recall_1	AUC-ROC
LR (baseline)	None	Yes	≈ 0.19	≈ 0.93
LR (baseline)	None	No	≈ 0.28	≈ 0.93
LR	Undersampling	No	≈ 0.86	≈ 0.80
LR	Oversampling	No	≈ 0.86	≈ 0.87
LR	SMOTE	No	≈ 0.80	≈ 0.96 (CV)
AdaBoost (LR)	Oversampling	No	≈ 0.86	≈ 0.88
DNN (best configuration)	SMOTE	No	≈ 0.63	≈ 0.94
Final LR (selected)	Oversampling	No	$\approx \mathbf{0.86}$	$\approx \mathbf{0.93}$

Key insights:

- Resampling is essential: all methods with resampling drastically outperform the original LR baseline in terms of recall on the minority class.
- Undersampling yields high recall but discards a large amount of data, leading to instability and many false positives.
- SMOTE provides strong AUC-ROC and recall, but relies on synthetic data whose interpretability may be questioned in some financial contexts.
- Random oversampling without PCA provides a strong and simple compromise: it preserves all real observations, improves recall to about 0.86, and maintains a high AUC-ROC.
- Ensemble methods (RF, XGB) show excellent accuracy but lower recall on bankruptcies, which is not aligned with our primary objective.
- DNNs perform well in terms of AUC but fail to surpass the best LR-based methods in terms of recall and simplicity.

6.2 Final model: Logistic Regression + Random Oversampling (No PCA)

We select **Logistic Regression with random oversampling and no PCA** as our final model. On the test set, this configuration yields approximately:

- recall on bankruptcies (class 1) ≈ 0.86,
- precision on class 1 ≈ 0.20,
- accuracy ≈ 0.88,
- AUC-ROC ≈ 0.93.

The confusion matrix for this model (schematically) shows that:

- the vast majority of bankruptcies are correctly detected (few false negatives),
- there are several hundred false positives (non-bankrupt firms flagged as risky),
- most non-bankrupt firms are correctly classified.

In risk management applications, this trade-off is acceptable: false alarms can be further investigated by analysts, whereas missing a truly distressed firm may incur large financial and reputational costs.

6.3 Interpretability and practical use

Logistic Regression retains a linear decision boundary in the original feature space, making it relatively easy to interpret. The learned coefficients can be viewed as weights assigned to standardized financial ratios, indicating how each variable affects the log-odds of bankruptcy. This is an important advantage over more complex models such as gradient boosting or deep neural networks.

In practice, the predicted probabilities can be used to construct risk scores at the firm level. Thresholds can then be set according to the institution's risk appetite, allowing the model to serve as an early-warning system. Because we train the model on oversampled data but evaluate on the original distribution, calibration of predicted probabilities may be required before deployment; this can be addressed through techniques such as Platt scaling or isotonic regression [13].

6.4 Limitations

Despite its good performance, the project has several limitations:

- The dataset is specific to Taiwanese firms between 1999 and 2009; external validity to other countries, sectors, or time periods is not guaranteed.
- We focused on resampling rather than cost-sensitive learning or advanced imbalance-aware algorithms, which could further improve performance.
- The effect of temporal dynamics (e.g. using time series of ratios) and macroeconomic variables has not been explored.
- Hyperparameter search for some models (e.g. DNNs, XGBoost) was constrained by computational resources; more extensive searches might yield additional gains.

7 Conclusion and Future Work

We have presented a complete machine learning pipeline for corporate bankruptcy prediction on a highly imbalanced financial dataset. Our work comprises detailed data cleaning and feature selection, systematic comparison of PCA vs. original features, evaluation of several baseline and advanced models, and an extensive study of resampling strategies for handling class imbalance.

The main conclusion is that a relatively simple model—**Logistic Regression with random oversampling and no PCA**—offers a highly effective and interpretable solution. It achieves a recall of approximately 0.86 on bankruptcies, with a high AUC-ROC and reasonable overall accuracy, making it suitable as an early-warning tool in risk management.

Future work may consider:

- Exploring cost-sensitive learning (e.g. class weights, custom loss functions) and threshold optimization directly on recall vs. false positive trade-offs.

- Investigating temporal models (recurrent neural networks, transformers) applied to sequences of financial statements.
- Incorporating macroeconomic indicators or text data (e.g. news, management reports) to enrich the feature space.
- Evaluating calibration methods to improve the probabilistic interpretation of predicted scores.
- Extending the analysis to other datasets (e.g. US or European firms) to assess robustness across markets.

References

[1] Edward I. Altman. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance*, 23(4):589–609, 1968. URL: https://doi.org/10.1111/j.1540-6261.1968.tb00843.x.

[2] James A. Ohlson. Financial ratios and the probabilistic prediction of bankruptcy. *Journal of Accounting Research*, 18(1):109–131, 1980. URL: https://doi.org/10.2307/2490395.

[3] S. Sarojini Devi and Y. Radhika. A survey on machine learning and statistical techniques in bankruptcy prediction. *International Journal of Machine Learning and Computing*, 8(2):133–139, 2018. URL: https://www.ijml.org/vol8/676-L0125.pdf.

[4] Apostolos Dasilas and A. Rigani. Machine learning techniques in bankruptcy prediction: A systematic literature review. *Expert Systems with Applications*, 255:124761, 2024. URL: https://doi.org/10.1016/j.eswa.2024.124761.

[5] Shekar Shetty, Mohamed Musa, and Xavier Brédart. Bankruptcy prediction using machine learning techniques. *Journal of Risk and Financial Management*, 15(1):35, 2022. URL: https://doi.org/10.3390/jrfm15010035.

[6] Mustapha Hamdi, Maher O. Kebaili, and others. Artificial intelligence techniques for bankruptcy prediction: Empirical evidence from Tunisian firms. *Journal of Risk and Financial Management*, 17(4):132, 2024. URL: https://www.mdpi.com/1911-8074/17/4/132.

[7] Dimitrios Billios, Ioannis Eriotis, and others. The power of numerical indicators in predicting bankruptcy. *Journal of Risk and Financial Management*, 17(10):433, 2024. URL: https://www.mdpi.com/1911-8074/17/10/433.

[8] Aditya Narvekar and Debashis Guha. Bankruptcy prediction using machine learning and an application to the case of the COVID-19 recession. *Data Science in Finance and Economics*, 1(2):180–195, 2021. URL: https://doi.org/10.3934/DSFE.2021010.

[9] Z. Díaz, D. Bennaceur, and A. Pérez. Machine learning and statistical techniques: An application to the prediction of insolvency in Spanish non-life insurance companies. *International Journal of Digital Accounting Research*, 5(1):1–23, 2005. URL: https://www.uhu.es/ijdar/10.4192/1577-8517-v5_1.pdf.

[10] Z. Rustam et al. Enhancing bankruptcy prediction of banks through machine learning approaches. Preprint, arXiv:2510.06852, 2025. URL: https://arxiv.org/abs/2510.06852.

[11] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002. URL: https://doi.org/10.1613/jair.953.

[12] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. URL: https://www.springer.com/gp/book/9780387310732.

[13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd edition, Springer, 2009. URL: https://hastie.su.domains/ElemStatLearn/.

[14] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012. URL: https://mitpress.mit.edu/9780262018029/machine-learning/.

[15] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. URL: https://doi.org/10.1145/2939672.2939785.

[16] Scikit-learn developers. Supervised learning user guide. Online documentation, accessed 2025. URL: https://scikit-learn.org/stable/supervised_learning.html.

[17] Scikit-learn developers. Neural network models (supervised). Online documentation, accessed 2025. URL: https://scikit-learn.org/stable/modules/neural_networks_supervised.html.

[18] Scikit-learn developers. Getting started with scikit-learn. Online documentation, accessed 2025. URL: https://scikit-learn.org/stable/getting_started.html.

[19] Federico Soriano. Company Bankruptcy Prediction. Kaggle dataset, accessed November 2025. URL: https://www.kaggle.com/datasets/fedesoriano/company-bankruptcy-prediction.

[20] Matej Papík et al. Automated machine learning in bankruptcy prediction of companies. *Procedia Computer Science*, 232:222–229, 2024. URL: https://doi.org/10.1016/j.procs.2024.01.141.

[21] A. Dasilas, A. Rigani. Machine learning techniques in bankruptcy prediction: A systematic literature review. *Expert Systems with Applications*, 255:124761, 2024. (Cited here as an example of a comprehensive review). URL: https://doi.org/10.1016/j.eswa.2024.124761.