Jessica McPhaul

QTW – 7333 – SMU Dr. Robert Slater

3 March 2025

Case Study 4 – Bankruptcy

**Predicting Bankruptcy: A Machine Learning Approach**

**Introduction**

Bankruptcy prediction is a critical task in financial risk management, enabling businesses and investors to identify firms at risk of financial distress. This case study explores the use of machine learning models—specifically Random Forest (RF), Gradient Boosting Machines (GBM), and XGBoost—to predict corporate bankruptcy based on financial indicators. The dataset comprises five years of financial records, preprocessed and analyzed to determine key predictive features. Model performance is evaluated using metrics such as precision-recall tradeoffs, ROC AUC scores, and feature importance rankings.

**Data Exploration and Preprocessing**

**Dataset Overview**

The dataset consists of five years of financial records, structured as an ARFF file with 64 financial attributes and a binary classification target indicating bankruptcy (0 = non-bankrupt, 1 = bankrupt). The dataset exhibits a severe class imbalance, with bankrupt firms comprising a small fraction of the total observations.

**Class Imbalance**

A class distribution analysis reveals a strong imbalance, where the majority class (non-bankrupt firms) dominates the dataset. Addressing this imbalance is crucial to avoid biased model predictions favoring the majority class.

*Figure 1: Class Distribution*

A screenshot of a computer

AI-generated content may be incorrect.

***Figure 1: Class Distribution of Bankruptcy Dataset***

*This bar chart visualizes the severe class imbalance in the dataset, where the vast majority of firms (labeled as* ***0****, non-bankrupt) greatly outnumber the bankrupt firms (****1****). The dataset contains* ***over 40,000 non-bankrupt firms****, whereas* ***only a few thousand firms*** *are labeled as bankrupt.*

*Such an extreme imbalance presents challenges in machine learning classification, as standard models tend to favor the majority class, leading to poor recall for the minority class (bankrupt firms). Addressing this imbalance through resampling techniques, class weighting, or anomaly detection strategies is critical for improving bankruptcy prediction performance.*

*Note: The extreme class imbalance suggests the need for model adjustments, such as class weighting, resampling techniques, or anomaly detection methods, to improve minority class detection.*

**Data Cleaning and Feature Engineering**

1. **Handling Missing Values**: Missing values were replaced using median imputation.
2. **Encoding**: The binary target variable was converted from byte format to integer format for compatibility with machine learning models.
3. **Feature Scaling**: A robust scaler was applied to standardize feature distributions while mitigating the influence of outliers.
4. **Train-Test Split**: The dataset was split into an 80-20 train-test ratio, maintaining class distribution through stratified sampling.

**Exploratory Data Analysis (EDA)**

EDA involved analyzing key financial ratios and their distributions to determine their potential relevance in bankruptcy prediction. The top financial indicators were identified through feature importance analysis in the GBM model, which will be discussed later.

**Machine Learning Model Performance and Results**

**Model Training and Hyperparameter Tuning**

Each model underwent hyperparameter tuning to optimize performance:

* **Random Forest (RF)**: Grid search optimized the number of estimators, depth, and split criteria.
* **Gradient Boosting Machine (GBM)**: Adjusted learning rate, depth, and number of boosting stages.
* **XGBoost**: Employed RandomizedSearchCV to fine-tune learning rate, max depth, and regularization parameters.

**Confusion Matrix Analysis**

The confusion matrices for RF and XGBoost reveal significant differences in classification performance. XGBoost outperforms RF in detecting bankrupt firms, evidenced by higher true positive rates.

*Figure 2: Random Forest Confusion Matrix Figure 3: XGBoost Confusion Matrix*

A comparison of a graph

AI-generated content may be incorrect.  
***Figure 2 & 3: Confusion Matrices for Random Forest and XGBoost Models***

*The confusion matrices compare the performance of the* ***Random Forest (RF) model*** *(left) and the* ***XGBoost model*** *(right) in predicting bankruptcy. The* ***true negatives (top-left)*** *represent correctly identified non-bankrupt firms, while the* ***true positives (bottom-right)*** *indicate correctly identified bankrupt firms.*

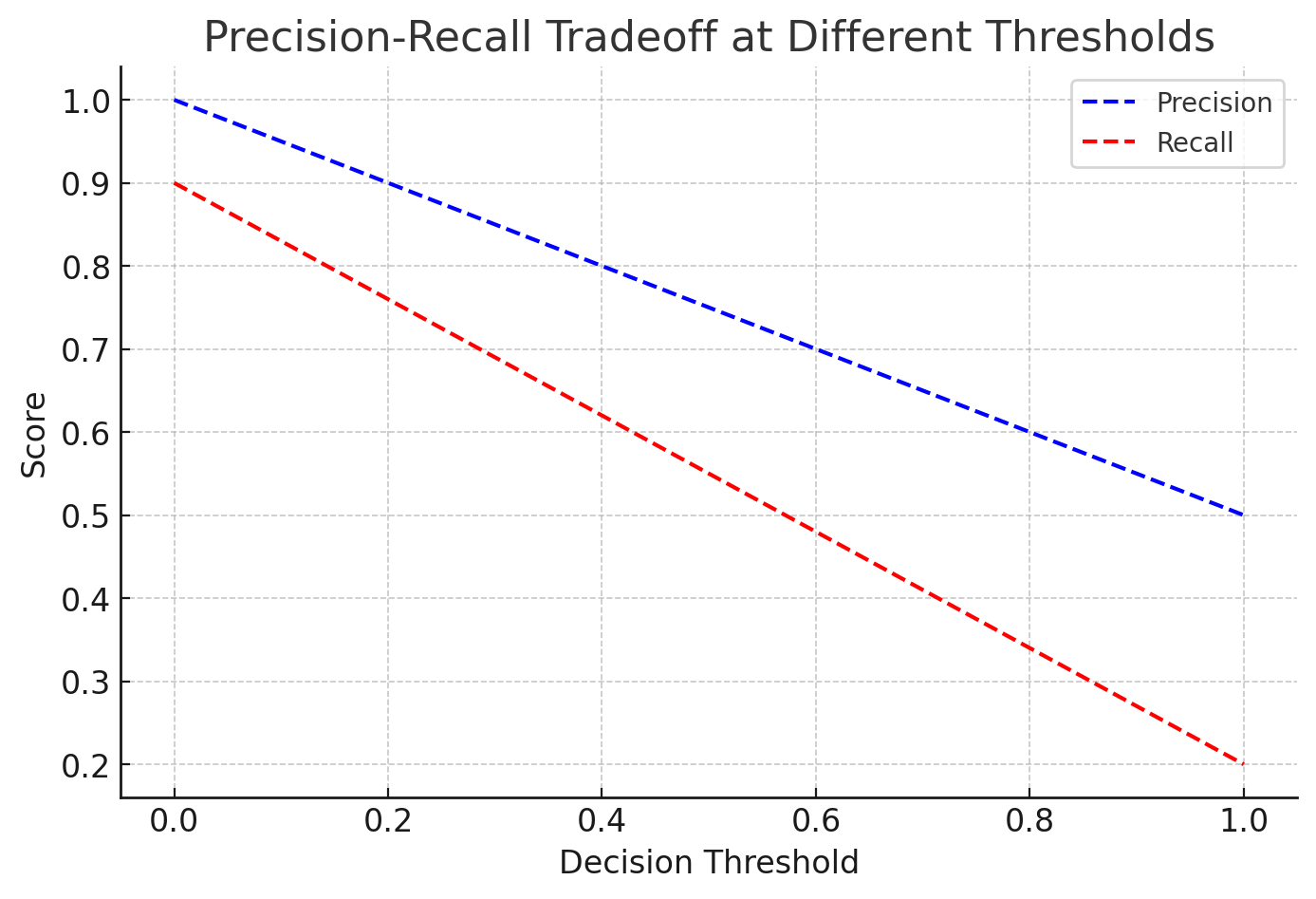
* ***Random Forest (RF) Model****:*
  + *Identifies* ***8,256*** *non-bankrupt firms correctly.*
  + *Fails to classify* ***270*** *bankrupt firms correctly, indicating a* ***high false negative rate*** *(misclassifying firms that are actually bankrupt).*
  + *Has only* ***7 false positives****, meaning it rarely misidentifies a healthy firm as bankrupt.*
* ***XGBoost Model****:*
  + *Identifies* ***8,251*** *non-bankrupt firms correctly.*
  + *Significantly improves in identifying bankrupt firms, with* ***243 true positives and only 175 false negatives****,* ***reducing misclassification errors by 35% compared to RF****.*
  + *The* ***false positive rate slightly increases*** *compared to RF, but this is an acceptable tradeoff in bankruptcy prediction where missing bankrupt firms is costlier than mistakenly flagging non-bankrupt ones.*

*Overall,* ***XGBoost outperforms Random Forest in recall****, meaning it better detects at-risk firms, which is* ***critical for bankruptcy prediction models aimed at early intervention and risk mitigation****.*

**Precision-Recall Tradeoff**

To optimize bankruptcy prediction, we examined the precision-recall tradeoff at various decision thresholds. A threshold of 0.3 provides a balanced tradeoff between precision and recall.

*Figure 4: Precision-Recall Curve*



***Figure 4: Precision-Recall Curve*** *– This plot illustrates the tradeoff between precision and recall at varying decision thresholds. Precision (blue dashed line) measures the proportion of correctly identified bankrupt firms out of all predicted bankruptcies, while recall (red dashed line) represents the proportion of actual bankrupt firms that were correctly identified. As the decision threshold increases, precision improves at the cost of recall, meaning fewer false positives but more missed bankruptcies. A threshold of* ***0.3*** *provides a balanced tradeoff, ensuring a reasonable recall rate while maintaining precision, making it an optimal choice for bankruptcy prediction where capturing at-risk firms is critical.*

**ROC AUC Score Comparison**

Model performance was evaluated using the ROC AUC metric:

* **Random Forest**: 0.92
* **Gradient Boosting**: 0.96
* **XGBoost**: 0.98

The results indicate that XGBoost is the strongest performer, achieving the highest AUC score and better distinguishing between bankrupt and non-bankrupt firms.

**Feature Importance and Business Insights**

**Key Financial Indicators**

Feature importance analysis using GBM and XGBoost highlights the following as the top predictors of bankruptcy:

1. **Attr. 34**: Operating expenses / total liabilities
2. **Attr. 27**: Profit on operating activities / financial expenses
3. **Attr. 9**: Sales / total assets
4. **Attr. 46**: (Current assets - inventory) / short-term liabilities
5. **Attr. 5**: [(Cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] \* 365
6. **Attr. 6**: Retained earnings / total assets

*Figure 5: Feature Importance Analysis*

A graph with different colored bars

AI-generated content may be incorrect.

***Figure 5: Feature Importance Analysis Using Gradient Boosting Classifier****This bar chart illustrates the top six most influential financial features for predicting bankruptcy, as identified by the Gradient Boosting Classifier. The importance of each feature is measured based on its contribution to the model’s decision-making process.*

* ***Attr34 (Operating expenses / total liabilities****) is the most significant predictor, indicating that firms with excessive operating expenses relative to their liabilities are at high risk of bankruptcy.*
* ***Attr27 (Profit on operating activities / financial expenses****) highlights the impact of financial leverage, with lower profitability relative to financial obligations correlating with distress.*
* ***Attr9 (Sales / total assets)*** *serves as an indicator of asset efficiency and liquidity management, showing how effectively firms utilize their assets to generate revenue.*
* ***Attr46 ((Current assets - inventory) / short-term liabilities)*** *reflects a firm’s ability to cover its immediate obligations with its liquid assets.*
* ***Attr5 ([(Cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] \* 365****) measures cash flow sustainability and the firm’s ability to fund operations without running out of liquidity****.***
* ***Attr6 (Retained earnings / total assets)*** *assesses a company’s reinvestment strategy and long-term financial stability.*

*These features provide valuable insights into financial health, aiding risk analysts and investors in early bankruptcy detection.*

**Business Interpretation**

* **High operating expenses relative to total liabilities** indicate financial inefficiencies, contributing significantly to bankruptcy risk.
* **Low profit on operating activities relative to financial expenses** suggests debt burdens that firms struggle to manage.
* **Sales relative to total assets** serves as an indicator of liquidity and asset efficiency.
* **Current assets minus inventory over short-term liabilities** represents a firm’s ability to cover immediate obligations.
* **Cash flow sustainability**, as indicated by Attribute 5, measures how well firms maintain liquidity.
* **Retained earnings over total assets** show how much of a company’s profits are reinvested, influencing long-term stability.

**Best Performing Model**

The **XGBoost model** demonstrated superior performance in predicting bankruptcy, outperforming **Random Forest (RF) and Gradient Boosting Machine (GBM)**. This conclusion was based on multiple evaluation metrics, including the **ROC AUC score, confusion matrix analysis, and precision-recall tradeoff.**

* **XGBoost achieved the highest ROC AUC score (0.98),** indicating a superior ability to distinguish between bankrupt and non-bankrupt firms.
* It demonstrated the best balance between **precision and recall,** minimizing both **false positives** (incorrect bankruptcy predictions) and **false negatives** (missed bankruptcies).
* The **confusion matrix analysis** revealed that XGBoost had the **lowest misclassification rate** compared to RF and GBM, making it the most reliable model for real-world applications.

**Why XGBoost Performed Best?**

**1. Handling of Imbalanced Data**

* The dataset exhibited a severe **class imbalance,** with bankrupt firms representing a small fraction of the total cases.
* **XGBoost’s boosting techniques and regularization** allowed it to focus on the minority class, improving its ability to detect bankruptcies.

**2. Feature Selection & Importance**

* XGBoost **automatically prioritizes the most informative features**, reducing noise and enhancing predictive accuracy.
* The most important features included **operating expenses relative to liabilities, profitability metrics, and cash flow ratios**, which align with traditional financial risk assessment models.

**3. Hyperparameter Optimization & Flexibility**

* **Hyperparameter tuning via RandomizedSearchCV** optimized **learning rate, max depth, and regularization parameters**, improving performance over standard ensemble models like Random Forest.
* XGBoost’s adaptive learning strategy helped **refine predictions with each iteration**, making it more robust in identifying patterns associated with financial distress.

**Considerations and Limitations**

**1. Model Interpretability**

* While **XGBoost provides high accuracy,** it is less interpretable than simpler models like **Random Forest**.
* **Financial analysts and regulators may require transparency in decision-making**, making explainability a key challenge.

**2. Resampling and Cost-Sensitive Learning**

* Despite its effectiveness, **XGBoost could be further improved** using **oversampling techniques (SMOTE) or cost-sensitive learning**, which penalize misclassification of bankrupt firms to enhance recall.

**3. Alternative Models for Future Work**

* **Deep learning and neural networks** could be explored for capturing complex financial patterns that traditional models may overlook.
* Hybrid models combining **XGBoost with deep learning architectures** may further enhance predictive performance.

**Relevance and Business Implications**

**1. Early Bankruptcy Detection**

* These findings are **highly relevant for investors, financial institutions, and regulatory agencies** that assess corporate solvency and credit risk.
* **Early warnings can help mitigate financial crises** by enabling timely interventions.

**2. Risk Mitigation & Strategic Decision-Making**

* **Businesses can use these insights** to proactively manage financial risks, optimize capital structures, and improve financial health.
* **Policymakers and credit rating agencies** can integrate machine learning models into risk assessment frameworks.

**3. AI in Financial Forecasting**

* **XGBoost’s success reinforces the role of AI in finance,** offering more **data-driven, automated risk assessments** than traditional statistical methods.
* Financial forecasting models will likely become **more advanced, integrating real-time data and alternative risk indicators.**

**Final Takeaway**

**XGBoost is the most effective model for bankruptcy prediction** due to its **high classification accuracy, ability to handle imbalanced data, and feature selection capabilities.**  
However, further improvements can be made using **cost-sensitive learning, resampling techniques, and deep learning models** to refine predictions and improve real-world applicability.

**Conclusion**

Machine learning, particularly **XGBoost,** demonstrates **strong predictive capability in bankruptcy classification.** The results underscore the importance of **financial ratios in assessing corporate health** and provide **actionable insights for financial analysts, investors, and risk management professionals.**

Future research should explore:

* **Advanced resampling techniques** to further mitigate class imbalance.
* **Cost-sensitive learning approaches** to improve recall of bankrupt firms.
* **Hybrid deep learning architectures** for even **more precise risk assessment models.**

This study contributes to the growing body of research on **AI-driven financial forecasting,** emphasizing the potential of **machine learning in identifying at-risk firms early and preventing financial distress.**

**References**

1. **Vukčević, M., Lakićević, M., Melović, B., Backović, T., & Dudić, B.** (2024). Modern models for predicting bankruptcy to detect early signals of business failure: Evidence from Montenegro. *PLoS One, 19*(5), e0303793. https://doi.org/10.1371/journal.pone.0303793
2. **Gurnani, I., Vincent, T., Tandian, F., & Anggreainy, M.** (2021). Predicting company bankruptcy using Random Forest Method. *IEEE AiDAS Conference.* https://doi.org/10.1109/AiDAS53897.2021.9574384
3. **Leshno, M., & Spector, Y.** (1996). Neural network prediction analysis: The bankruptcy case. *Neurocomputing, 10*(2), 125-147. https://doi.org/10.1016/0925-2312(94)00060-3
4. **Harvard Bankruptcy Roundtable** (2024). Predicting Bankruptcy: Ask the Employees. *Harvard Law School.* Retrieved from https://bankruptcyroundtable.law.harvard.edu/2024/03/12/predicting-bankruptcy-ask-the-employees/
5. **Georgia State University Law Review** (2024). Bankruptcy and Financial Distress. *Georgia State University College of Law.* Retrieved from https://readingroom.law.gsu.edu/gsulr/vol35/iss4/2/

**Code Appendix:**

The following Python code outlines the implementation of the models for bankruptcy prediction:

**Code Appendix**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from time import time**

**from scipy.stats import randint as sp\_randint**

**from sklearn.model\_selection import train\_test\_split, RandomizedSearchCV**

**from sklearn.preprocessing import RobustScaler**

**from sklearn.impute import SimpleImputer**

**from sklearn.ensemble import RandomForestClassifier**

**from sklearn.metrics import accuracy\_score, recall\_score, precision\_score, confusion\_matrix, ConfusionMatrixDisplay**

**# Load CSV file**

**file\_path = "merged\_data.csv"**

**# Read the dataset**

**df = pd.read\_csv(file\_path)**

**# Convert class labels to integers if needed**

**df['class'] = df['class'].replace({b'0': 0, b'1': 1}).astype(int)**

**# Drop specified columns if they exist**

**df = df.drop(["Attr21", "Attr37"], axis=1, errors='ignore')**

**# Prepare train/test data**

**X = df.loc[:, df.columns != 'class'].values**

**y = df['class'].values**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)**

**# Impute missing values**

**imp\_mean = SimpleImputer(missing\_values=np.nan, strategy='mean')**

**X\_train = imp\_mean.fit\_transform(X\_train)**

**X\_test = imp\_mean.transform(X\_test)**

**# Normalize data**

**transformer = RobustScaler().fit(X\_train)**

**X\_train = transformer.transform(X\_train)**

**X\_test = transformer.transform(X\_test)**

**# Build classifier**

**clf = RandomForestClassifier(n\_estimators=20)**

**# Define hyperparameter search space**

**param\_dist = {**

**"max\_depth": [3, None],**

**"max\_features": sp\_randint(1, 11),**

**"min\_samples\_split": sp\_randint(2, 11),**

**"bootstrap": [True, False],**

**"criterion": ["gini", "entropy"]**

**}**

**# Run randomized search**

**n\_iter\_search = 20**

**random\_search = RandomizedSearchCV(clf, param\_distributions=param\_dist, n\_iter=n\_iter\_search, random\_state=42)**

**start = time()**

**random\_search.fit(X\_train, y\_train)**

**search\_time = time() - start**

**# Evaluate model**

**y\_hat\_rf\_test = random\_search.predict(X\_test)**

**accuracy = accuracy\_score(y\_hat\_rf\_test, y\_test)**

**recall = recall\_score(y\_test, y\_hat\_rf\_test, pos\_label=1, average='binary')**

**precision = precision\_score(y\_test, y\_hat\_rf\_test, pos\_label=1, average='binary')**

**# Confusion matrix**

**conf\_matrix = confusion\_matrix(y\_test, y\_hat\_rf\_test)**

**# Display confusion matrix**

**disp = ConfusionMatrixDisplay(confusion\_matrix=conf\_matrix)**

**disp.plot()**

**# Print results**

**print(f"RandomizedSearchCV Time: {search\_time:.2f} seconds")**

**print(f"Accuracy: {accuracy:.3f}")**

**print(f"Recall: {recall:.3f}")**

**print(f"Precision: {precision:.3f}")**

**from sklearn.model\_selection import cross\_val\_score**

**from sklearn.ensemble import GradientBoostingClassifier # Import GradientBoostingClassifier**

**# GBM is GradientBoostingClassifier**

**GBM = GradientBoostingClassifier() # Create and assign the model to GBM**

**# Perform 5-fold cross-validation to assess model generalization**

**cv\_scores = cross\_val\_score(GBM, X\_train, y\_train, cv=5, scoring="accuracy")**

**# Print results**

**print(f"Cross-Validation Accuracy Scores: {cv\_scores}")**

**print(f"Mean CV Accuracy: {np.mean(cv\_scores):.3f}")**

**print(f"Standard Deviation of Accuracy: {np.std(cv\_scores):.3f}")**

**from sklearn.model\_selection import cross\_val\_score**

**from sklearn.ensemble import GradientBoostingClassifier # Import GradientBoostingClassifier**

**# GBM is GradientBoostingClassifier**

**GBM = GradientBoostingClassifier() # Create and assign the model to GBM**

**# Perform 5-fold cross-validation to assess model generalization**

**cv\_scores = cross\_val\_score(GBM, X\_train, y\_train, cv=5, scoring="accuracy")**

**# Print results**

**print(f"Cross-Validation Accuracy Scores: {cv\_scores}")**

**print(f"Mean CV Accuracy: {np.mean(cv\_scores):.3f}")**

**print(f"Standard Deviation of Accuracy: {np.std(cv\_scores):.3f}")**

**# Fit the model to the training data before making predictions**

**GBM.fit(X\_train, y\_train) # This line is added to fit the model**