Case Study 4: Bankruptcy Prediction Using Random Forest and XGBoost

Kristin Henderson

June 21, 2025

1 Introduction

This study aims to identify companies at risk of bankruptcy using two tree-based classification models: Random Forest and eXtreme Gradient Boosting (XGBoost). Accurate bankruptcy prediction enables investors to make proactive decisions that minimize financial risk and potential losses.

2 Data

2.1 Dataset

The Polish Companies Bankruptcy dataset used in this analysis comes from the University of California Irvine Machine Learning Repository. It includes 43,405 financial records from Polish companies across a five-year forecasting period (2000-2013). Each record, grouped by year, contains 64 numeric financial features and a binary target variable indicating whether the company went bankrupt during the period. The features are described in Table 1.

The target classes are highly imbalanced: only 4.8% of records indicate bankruptcy, while 95.2% indicate non-bankruptcy as shown in Figure 1.

This dataset was previously analyzed by Zięba et al. [1], who compared 16 classification models. Unlike the current study, which treats the dataset as a single prediction task, their approach split it into five separate classification problems based on the forecast year (i.e., 1st through 5th year of forecasting period).

2.2 Missing Values and Imputation

Each of the financial features contained some missing values, though 50 of the 64 features had fewer than 0.5% missing. For these, missing values were imputed using the median within each year. Visual inspection of feature distributions (with extreme values removed for clarity), showed many features to be right-skewed. Given this skewness, the median was preferred for imputation over the mean. The distributions of X37 representing (Current assets - inventories) / long-term liabilities and X21 representing Sales (n) / sales (n-1), the two features with the highest proportions of missing values, are shown in Figure 2 as representative examples of the typical distributions of all features.

For a subset of features, it was possible to calculate missing values by reconstructing component variables. These calculations are shown in Table 2, with the resulting imputed features listed in Table 3. In some cases, calculations were not possible due to division by zero or because components were themselves missing. When this occurred, again missing values were imputed with the median by year.

Table 1: The set of features considered in the classification process.

ID	Description	ID	Description
X1	Net profit / total assets	X33	Operating expenses / short-term liabilities
X2	Total liabilities / total assets	X34	Operating expenses / total liabilities
X3	Working capital / total assets	X35	Profit on sales / total assets
X4	Current assets / short-term liabilities	X36	Total sales / total assets
X5	[(Cash + short-term securities + receivables - short-term liabilities) / (Operating expenses - depreciation)] x 365	X37	(Current assets - inventories) / long-term liabilities
X6	Retained earnings / total assets	X38	Constant capital / total assets
X7	EBIT / total assets	X39	Profit on sales / sales
X8	Book value of equity / total liabilities	X40	(Current assets - inventory - receivables) / short-term liabilities
X9	Sales / total assets	X41	Total liabilities / ((Profit on operating activities + depreciation) x (12/365))
X10	Equity / total assets	X42	Profit on operating activities / sales
X11	(Gross profit + extraordinary items + financial expenses) / total assets	X43	Rotation receivables + inventory turnover in days
X12	Gross profit / short-term liabilities	X44	(Receivables x 365) / sales
X13	(Gross profit + depreciation) / sales	X45	Net profit / inventory
X14	(Gross profit + interest) / total assets	X46	(Current assets - inventory) / short-term liabilities
X15	(Total liabilities x 365) / (Gross profit + depreciation)	X47	(Inventory x 365) / cost of products sold
X16	(Gross profit + depreciation) / total liabilities	X48	EBITDA (Profit on operating activities - depreciation) / total assets
X17	Total assets / total liabilities	X49	EBITDA (Profit on operating activities depreciation) / sales
X18	Gross profit / total assets	X50	Current assets / total liabilities
X19	Gross profit / sales	X51	Short-term liabilities / total assets
X20	(Inventory x 365) / sales	X52	(Short-term liabilities x 365) / cost of products sold
X21	Sales (n) / sales (n-1)	X53	Equity / fixed assets
X22	Profit on operating activities / total assets	X54	Constant capital / fixed assets
X23	Net profit / sales	X55	Working capital
X24	Gross profit (in 3 years) / total assets	X56	(Sales - cost of products sold) / sales
X25	(Equity - share capital) / total assets	X57	(Current assets - inventory - short-term liabilities) / (Sales - gross profit - depreciation)
X26	(Net profit + depreciation) / total liabilities	X58	Total costs / total sales
X27	Profit on operating activities / financial expenses	X59	Long-term liabilities / equity
X28	Working capital / fixed assets	X60	Sales / inventory
X29	Logarithm of total assets	X61	Sales / receivables
X30	(Total liabilities - cash) / sales	X62	(Short-term liabilities x 365) / sales
X31	(Gross profit + interest) / sales	X63	Sales / short-term liabilities
X32	(Current liabilities x 365) / cost of products sold	X64	Sales / fixed assets
Year	nth year (1-5) of Forecasting Period	Class	Target:Bankruptcy or Not (1 or 0)

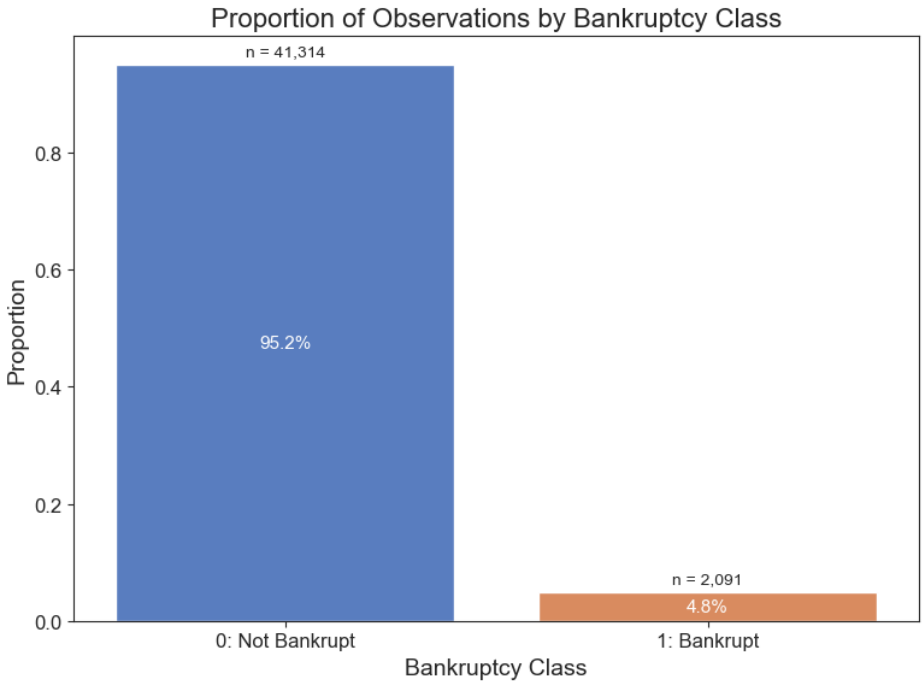

Figure 1: Barplot showing the proportion of observations in each bankruptcy class, with counts and percentages labeled on the bars.

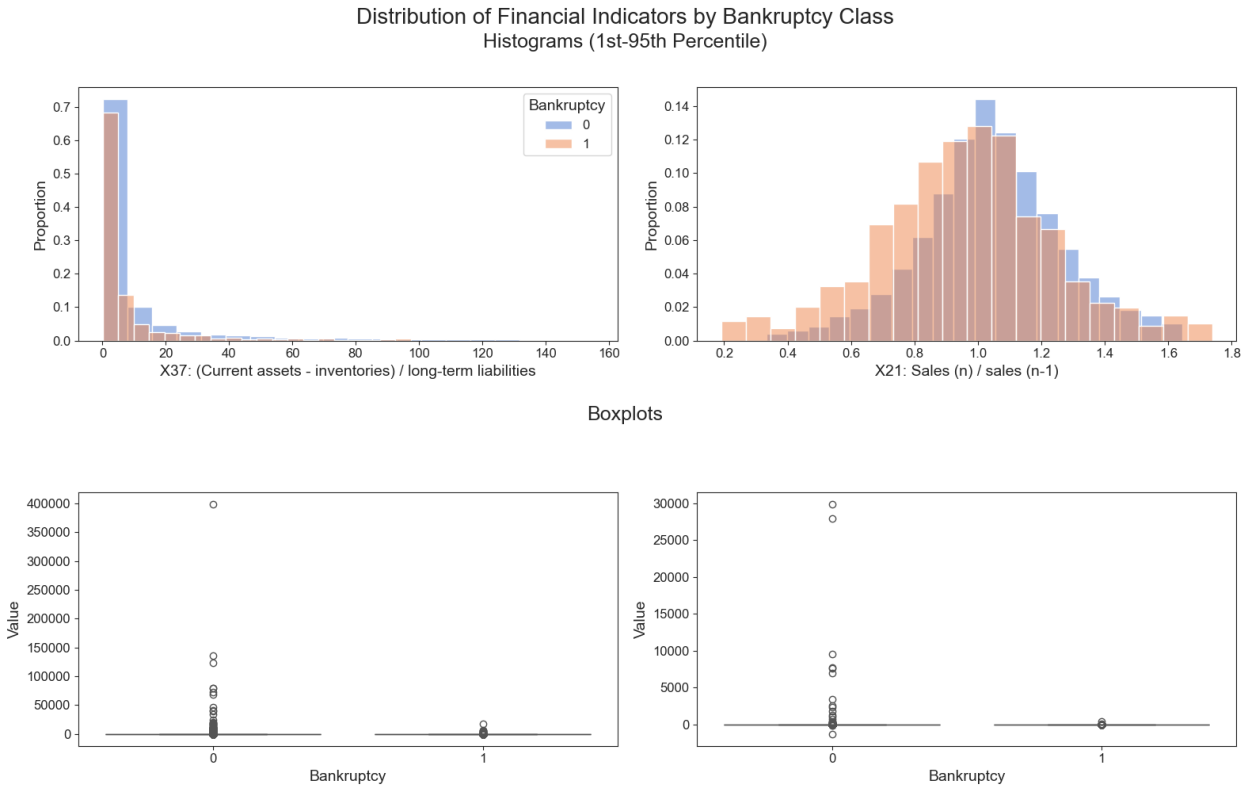

Figure 2: Histograms and boxplots of features X37 (top) and X21 (bottom) by bankruptcy class. Histograms are clipped to the 1st-95th percentile to remove extreme values for clarity. Boxplots show the full distribution.

Table 2: Feature components that can be calculated using X55 (Working capital)

Feature Component	**Formula for Calculation**
Total assets	X_{55}/X_3
Fixed assets	X_{55}/X_{28}
Equity	$X_{10} \times$ total assets
Constant capital	$X_{38} \times$ total assets
Sales	$X_9 \times$ total assets
Net profit	$X_1 \times$ total assets
Inventory	$(X_{20} \times \text{sales})/365$
Cost of products sold	sales $- (X_{56} \times \text{sales})$
Short-term liabilities	$X_{51} \times$ total assets
Gross profit	$X_{18} \times$ total assets
Depreciation	$(X_{13} \times \text{sales}) -$ gross profit
Profit on operating activities	$X_{22} \times$ total assets
Total liabilities	$X_2 \times$ total assets

Table 3: Features that can be imputed after calculating missing values

Feature (ID)	**Derived From**
X41	Total liabilities / ((Profit on operating activities + depreciation) $\times$ (12/365))
X45	Net profit / inventory
X47	(Inventory $\times$ 365)/cost of products sold
X52	(Short-term liabilities $\times$ 365)/cost of products sold
X53	Equity / fixed assets
X54	Constant capital / fixed assets
X60	Sales / inventory
X64	Sales / fixed assets

Two features contained the largest proportions of missing values, X37 (43.7% missing) and X21 (13.5% missing). These features did not show correlation with other variables and no missingness pattern was detected.

Some features showed consistent missingness patterns, often due to shared formula components, but otherwise the missing values appeared to be random. Those were X24 (2.1% missing), X27 (6.4% missing), X28 (1.9% missing), and X32 (0.1% missing).

Because these features could not be calculated, they were imputed with the median by year.

Two features, X7 (EBIT / total assets) and X14 ((Gross profit + interest) / total assets), were nearly perfectly correlated with each other. X14 was dropped from the analysis to avoid redundancy. Modeling was performed both with and without year as a feature. The year variable was also excluded from the final model because slightly higher area under the curve (AUC) was achieved without it.

3 Modeling

3.1 Random Forest

Hyperparameter tuning for the Random Forest classifier was performed using `RandomizedSearchCV` with 30 parameter combinations and 5-fold stratified cross-validation to preserve class balance. The `class_weight='balanced'` option was used to address class imbalance, and the data was shuffled to randomize the order of rows across forecasting years. To balance performance and computation time, 100 trees were used during tuning. As in Zięba et al. [1], the best parameter combination was selected by maximizing the AUC. The tuned parameters and their values are shown in Table 4.

The best parameter combination was then used to fit a model with 1,000 trees in a 5-fold cross-validation. Out-of-fold predictions and performance metrics were generated using `cross_val_predict`.

Table 4: Random Forest Hyperparameter Search Space

Hyperparameter	**Values Considered**
criterion	gini, entropy
max_depth	2, 4, 6, 8, 10, 12, 15, 20, 25 (to match XGBoost tuning)
max_features	sqrt, log2, 0.5, None
min_samples_leaf	1, 2, 4
min_samples_split	2, 4, 8

3.2 XGBoost

The XGBoost model was created using the `xgboost` library. Data was converted to a `DMatrix` object for model building. For hyperparameter tuning, the native `xgboost.cv` method was employed with 50 random parameter combinations to accommodate a larger search space than was used for the Random Forest. As before, 5-fold stratified cross-validation was used to preserve class distribution in the folds. Models were trained with up to 1,000 boosted rounds, with early stopping after 10 rounds. A `binary:logistic` objective function was used. The best parameter combination was selected based on AUC.

Initially, the learning rate (`eta`) was set to the default value of 0.3 and excluded from tuning. However, this typically resulted in fewer than 200 boosting rounds. To achieve a target between 200 and 500 boosting rounds, `eta` was later included in the tuning process. The parameters tuned and their values are shown in Table 5.

`xgboost.train` was used within a manually written 5-fold cross-validation loop to generate out-of-fold predictions and performance metrics.

Table 5: XGBoost Hyperparameter Search Space

Hyperparameter	**Values Considered**
max_depth	2, 4, 6, 8, 10, 12, 15, 20, 25
subsample	Random values between 0.5 and 1.0
colsample_bytree	Random values between 0.5 and 1.0
colsample_bylevel	Random values between 0.5 and 1.0
colsample_bynode	Random values between 0.5 and 1.0
reg_lambda	0.01, 0.1, 1, 10, 100
eta (learning rate)	0.1, 0.2, 0.3

3.3 Tuned Parameters

The best parameter combinations for each model are listed in Table 6. The Random Forest model performed best using entropy as the criterion for measuring information gain. To help prevent overfitting, the optimal configuration included a maximum depth of 25, 50% of features considered at each split, a minimum of 2 samples per leaf, and a minimum of 4 samples per split.

The XGBoost model performed best with random subsampling at high proportions of rows and features: 84.3% of rows, 84.8% of features per tree, 84.7% per level, and 95.9% per node. This randomness helps reduce overfitting and improve generalization. Additional overfitting control came from a maximum tree depth of 10 and an L2 regularization term (λ) of 0.01. A learning rate of 0.1 (lower than the default 0.3) led to more gradual convergence, resulting in 103 to 199 boosting rounds per fold.

Table 6: Best Hyperparameter Combinations for Tree-Based Models

Random Forest	**XGBoost**
criterion = entropy	max_depth = 10
max_depth = 25	subsample = 0.8430
max_features = 0.5	colsample_bytree = 0.8484
min_samples_leaf = 2	colsample_bylevel = 0.8471
min_samples_split = 4	colsample_bynode = 0.9595
	reg_lambda = 0.01
	eta (learning rate) = 0.1

Note: XGBoost parameter values have been rounded to four decimal place for clarity.

4 Results

4.1 Confusion Matrices

To generate the confusion matrices shown in Figure 3 (on a subsequent page), a default classification threshold of 0.5 was used for both models. There was virtually no difference in how the models classified the non-bankruptcy class: XGBoost correctly identified one additional observation compared to Random Forest—41,238 versus 41,237, respectively.

However, performance diverged on the bankruptcy class. XGBoost had a significantly lower false negative rate and a higher true positive count than Random Forest. Specifically, Random Forest misclassified 1,244 bankruptcy cases as non-bankruptcy and correctly identified 847, while XGBoost misclassified 975 and correctly identified 1,116. This suggests that XGBoost was more effective at identifying bankruptcy cases while maintaining nearly identical performance on non-bankruptcy cases at a threshold of 0.5.

4.2 ROC Curves

The Receiver Operating Characteristic (ROC) curves for the two models are shown in Figure 4. The mean out-of-fold AUC was 0.9594 for the Random Forest model and 0.9719 for the XGBoost model. The AUC is a threshold independent metric for comparison of models.

4.3 Classification Report

A classification report for the two models is shown in Table 7, and a barplot of the metrics is shown in Figure 5. Precision and Recall are reported for the bankruptcy class. Mean out-of-fold accuracy, precision and recall were all higher for XGBoost than for Random Forest: 0.9758, 0.9362, and 0.5337 versus 0.9696, 0.9167, and 0.4051, respectively.

Table 7: Model Performance Comparison (5-fold Stratified Cross-Validation)

Metric (mean ± std)	**Random Forest**	**XGBoost**
Accuracy	0.9696 ± 0.0014	0.9758 ± 0.0013
Precision	0.9167 ± 0.0268	0.9362 ± 0.0171
Recall	0.4051 ± 0.0232	0.5337 ± 0.0328
AUC	0.9594 ± 0.0014	0.9719 ± 0.0027

5 Conclusion

The non-overlapping confidence intervals (mean $\pm$ 1.96 $\times$ std) suggest that XGBoost significantly outperformed Random Forest on accuracy, recall, and AUC. However, precision was similar between models, with

overlapping intervals. This likely reflects their similar handling of the much larger non-bankruptcy class, which resulted in comparable false positives. Although XGBoost identified more bankruptcy cases, the overall number of positive cases remained small—limiting the impact on precision. Recall was substantially higher for XGBoost, 53.4% versus 40.5%, suggesting it was much better at identifying companies that went bankrupt, the primary goal of the stakeholders in this analysis.

XGBoost is the recommended model for identifying companies at risk of bankruptcy. Not only were all performance metrics higher, but it was also considerably more efficient. In this implementation, 5-fold cross-validation including training and prediction took only 0.082 minutes for XGBoost. In contrast, Random Forest required 4.941 minutes to run `cross_val_predict` for 1,000 trees. This large efficiency gap may be partly explained by XGBoost requiring fewer iterations (103 to 199 boosting rounds), compared to 1,000 trees for Random Forest, as well as shallower optimal tree depths (10 versus 25).

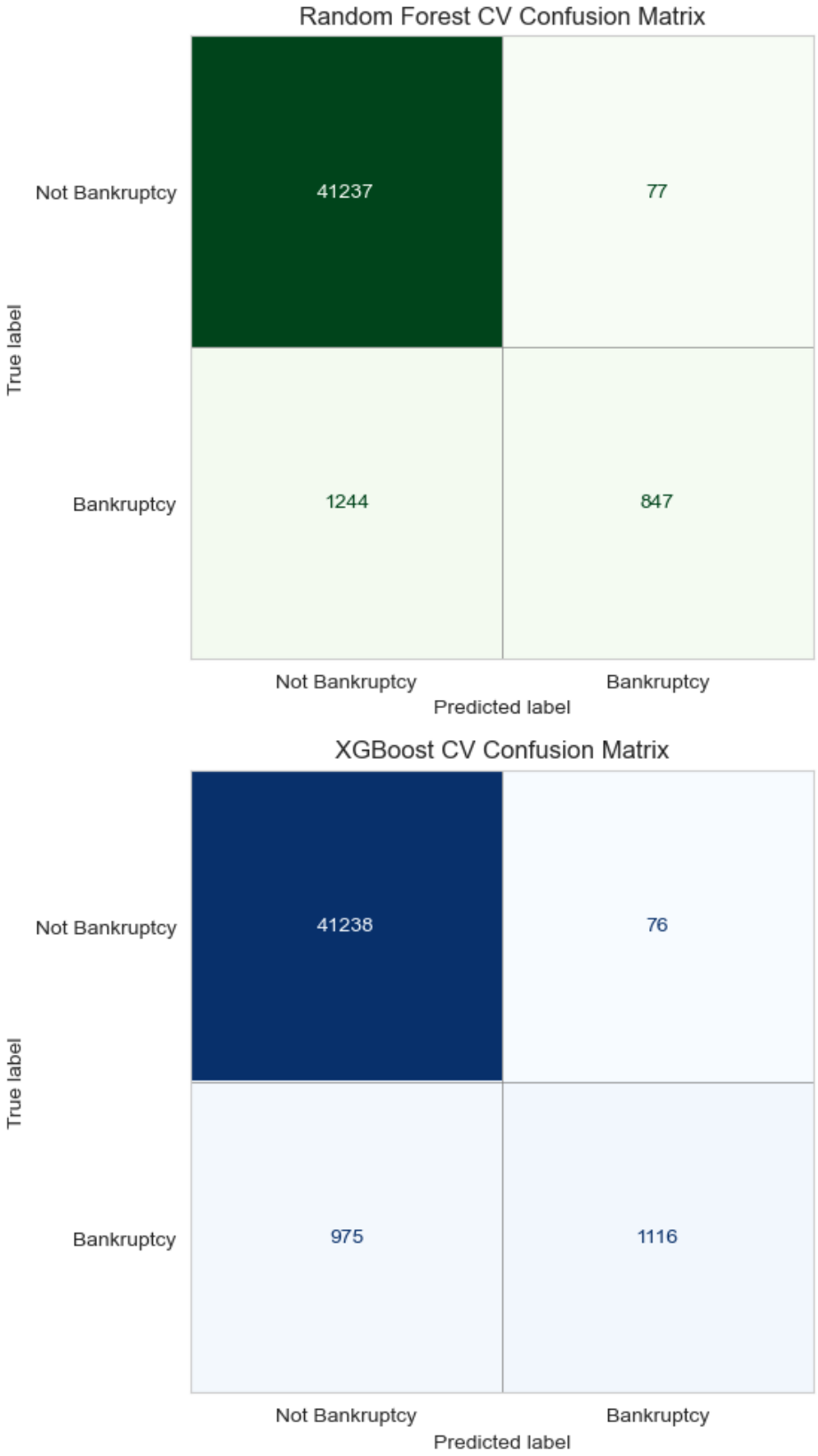

Figure 3: Confusion matrices showing classification results for the Random Forest (top, green) and XGBoost (bottom, blue) models.

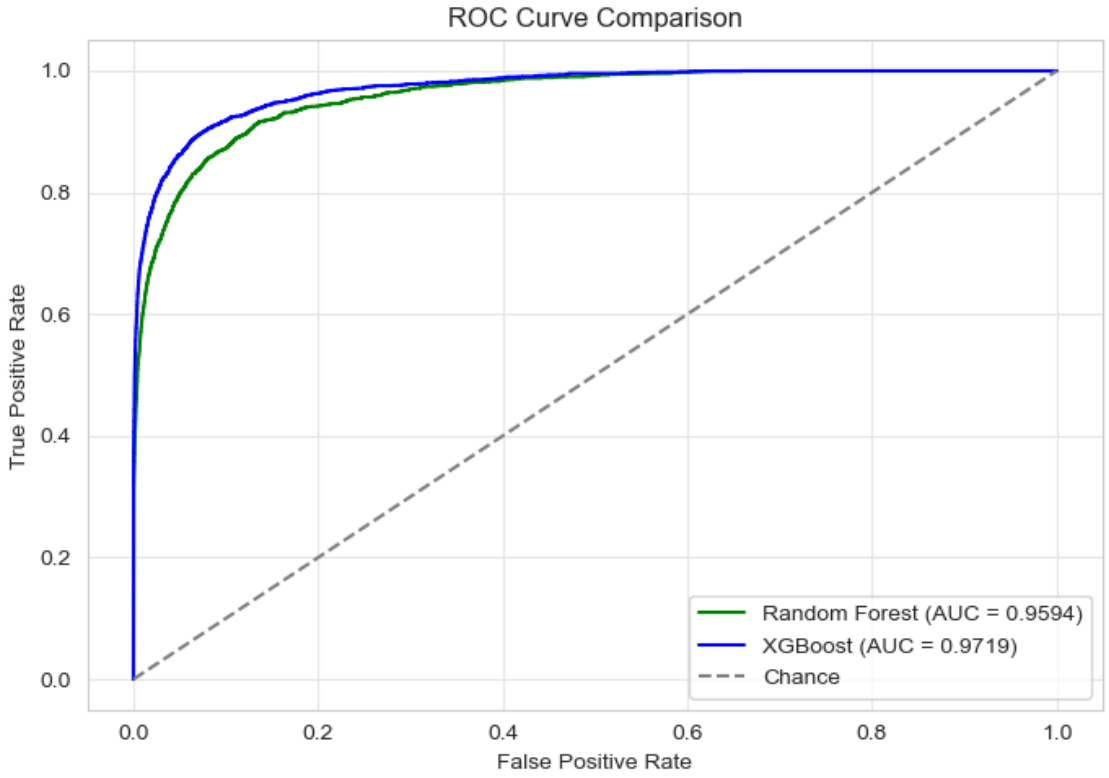

Figure 4: ROC curves for the Random Forest (green) and XGBoost (blue) models.

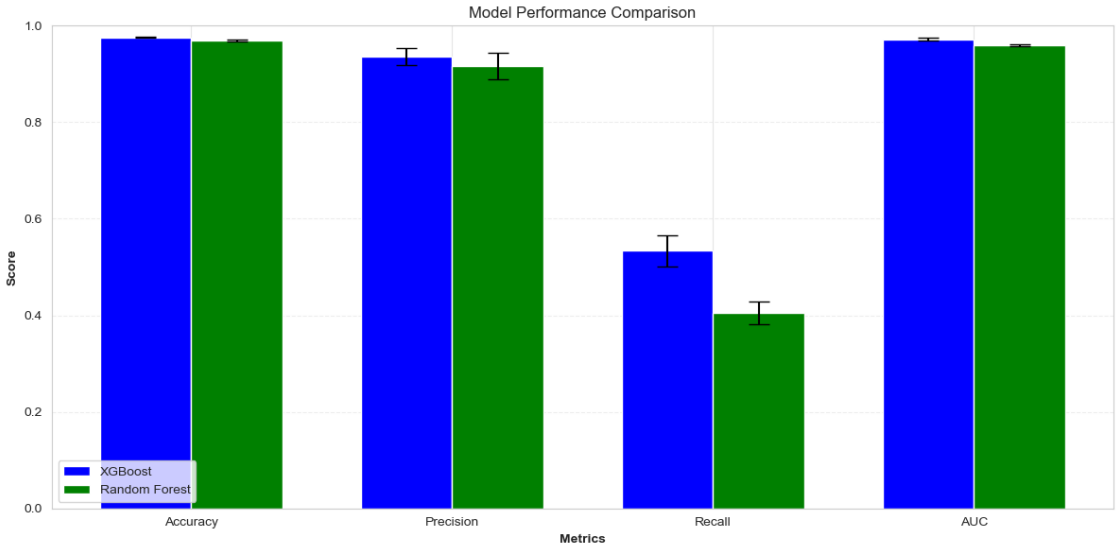

Figure 5: Mean performance metrics for the Random Forest (green, right bars) and XGBoost (blue, left bars) models. Error bars indicate standard deviation across cross-validation folds.

References

[1] Zięba, M., Tomczak, S. K., & Tomczak, J. M. (2016). *Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction.* Expert Systems with Applications, 58, 93-101. https://doi.org/10.1016/j.eswa.2016.04.001.