Loan Default Prediction Using XGBoost, Logistic Regression, and Neural Networks

Siddhant Maji
Independence High School
Frisco, Texas
siddhant.maji@gmail.com

Abstract

This paper evaluates the results of three machine learning models: XGBoost, logistic regression, and feedforward neural networks for predicting loan default risk based on a structured dataset of more than 200,000 loan applications. Financially significant features like affordability ratio and total interest were engineered, and skewed variables were transformed with Box-Cox. Techniques such as label encoding and class balancing methods were used. All models were evaluated on ROC AUC, precision, recall, and F1-Score. The best overall performance was achieved with XGBoost with an ROC AUC of 0.76, where some of the key predictors were age, affordability ratio, and interest rate. The results indicate the usefulness of tree-based models in financial risk forecasting and show how it can be further enhanced with the addition of interpretability and inclusion of temporal data.

1 Introduction

Loan risk assessment is fundamental to lending institutions, directly influencing their ability to protect capital and price loans appropriately. Predicting defaulters before disbursement of loans can facilitate data-based credit scoring [1], portfolio risk assessment, and loan product design. Modern machine learning methods provide a flexible alternative to traditional methods like proprietary scoring models or discriminant analysis. Gradient-boosted decision trees and neural networks are algorithms which can learn nonlinearity and complex feature interactions. Such practices have formed part of financial modeling, driving applications in credit card fraud detection, algorithmic underwriting, and more. This paper takes a comparative perspective in loan default prediction [2] and examines the interpretable tree-based models in contrast to deeper neural networks [], in addition to logistic regression [3], a widely used statistical model for default prediction. A real-life dataset of more than 200,000 consumer loan applications is used, feature engineering is applied to create financially meaningful features, and issues like data skewness and class imbalance are addressed. The performance of the models is evaluated through several metrics like ROC AUC, precision, recall, and F1 score.

2 Dataset and Preprocessing

2.1 Data Description

The dataset consists of borrower loan default records and was provided by Coursera:

– **Training set:** 204,277 labeled samples with a binary `Default` indicator (0 = non-default, 1 = defaulted)
– **Test set:** 51,070 labeled samples for final evaluation

Feature Name	Data Type	Description
Age	Integer	The age of the borrower.
Income	Integer	The annual income of the borrower.
LoanAmount	Integer	The amount of money being borrowed.
CreditScore	Integer	The credit score of the borrower, indicating their creditworthiness.
MonthsEmployed	Integer	The number of months the borrower has been employed.
InterestRate	Float	The interest rate for the loan.
LoanTerm	Integer	The term length of the loan in months.
DTIRatio	Float	Debt-to-Income ratio, indicating debt compared to income.
NumCreditLines	Integer	The number of credit lines the borrower has open.
Education	String	Highest level of education attained (PhD, Master's, Bachelor's, High School).
EmploymentType	String	Type of employment (Full-time, Part-time, Self-employed, Unemployed).
MaritalStatus	String	Marital status (Single, Married, Divorced).
LoanPurpose	String	Purpose of the loan (Home, Auto, Education, Business, Other).
HasMortgage	String	Whether the borrower has a mortgage (Yes or No).
HasDependents	String	Whether the borrower has dependents (Yes or No).
HasCoSigner	String	Whether the loan has a co-signer (Yes or No).
LoanID	String	A unique identifier for each loan.
Default	Integer	Binary target indicating if the loan defaulted (1) or not (0).

Table 1: List of features used in the dataset along with their data types and descriptions.

The dataset contains no missing values, allowing for direct modeling without imputation.

2.2 Financial Feature Engineering

To enhance predictive power, derived features grounded in financial principles were constructed as follows:

1. **Affordability Ratio (AffRatio):** Calculated as the ratio of loan amount to borrower income, capturing the relative loan burden.
$$\texttt{AffRatio} = \frac{\texttt{LoanAmount}}{\texttt{Income}}$$
2. **Total Interest (TotalInterest):** Estimated as the product of interest rate and loan term, representing overall borrowing cost.
$$\texttt{TotalInterest} = \texttt{InterestRate} \times \texttt{LoanTerm}$$
3. **Debt (Debt):** Approximated by scaling income with the debt-to-income ratio, reflecting total debt obligations.
$$\texttt{Debt} = \texttt{DTIRatio} \times \texttt{Income}$$
4. **Average Borrowed per Credit Line (AvgBorrowed):** Indicating average exposure across credit lines.
$$\texttt{AvgBorrowed} = \frac{\texttt{LoanAmount}}{\texttt{NumCreditLines}}$$

These features encode borrower financial capacity and debt exposure, which are critical indicators in default prediction.

Skewness and Box–Cox Transformation

A few of these features, namely AffRatio, TotalDebt, and AvgBorrowed, exhibited strong right skewness with long tails, which can negatively impact model performance.

To address this, the **Box-Cox transformation** [4] to affected variables:

$$y_i^{\lambda} = \begin{cases} \frac{y_i^{\lambda}-1}{\lambda}, & \lambda \neq 0 \\ \ln(y_i), & \lambda = 0 \end{cases}$$

Box–Cox transforms data to approximate a Gaussian distribution by finding an optimal λ that minimizes log-likelihood error under normality.

Post-transformation, skewness dropped significantly.

Feature	**Skewness (Before)**	**Skewness (After)**
AffRatio	2.334	0.001
AvgBorrowed	1.424	-0.025
TotalInterest	0.866	-0.061
Debt	0.847	-0.059
Education	0.007	0.007
CreditScore	0.005	0.005
InterestRate	0.005	0.005
EmploymentType	0.001	0.001
Age	0.001	0.001
MaritalStatus	0.000	0.000
HasMortgage	-0.000	-0.000
NumCreditLines	-0.000	-0.000
LoanPurpose	-0.000	-0.000
Income	-0.000	-0.000
HasCoSigner	-0.000	-0.000
HasDependents	-0.001	-0.001
DTIRatio	-0.001	-0.001
LoanAmount	-0.002	-0.002
MonthsEmployed	-0.002	-0.002
LoanTerm	-0.002	-0.002

Table 2: Skewness of numerical features before and after Box–Cox transformation. Highly skewed variables showed substantial reductions.

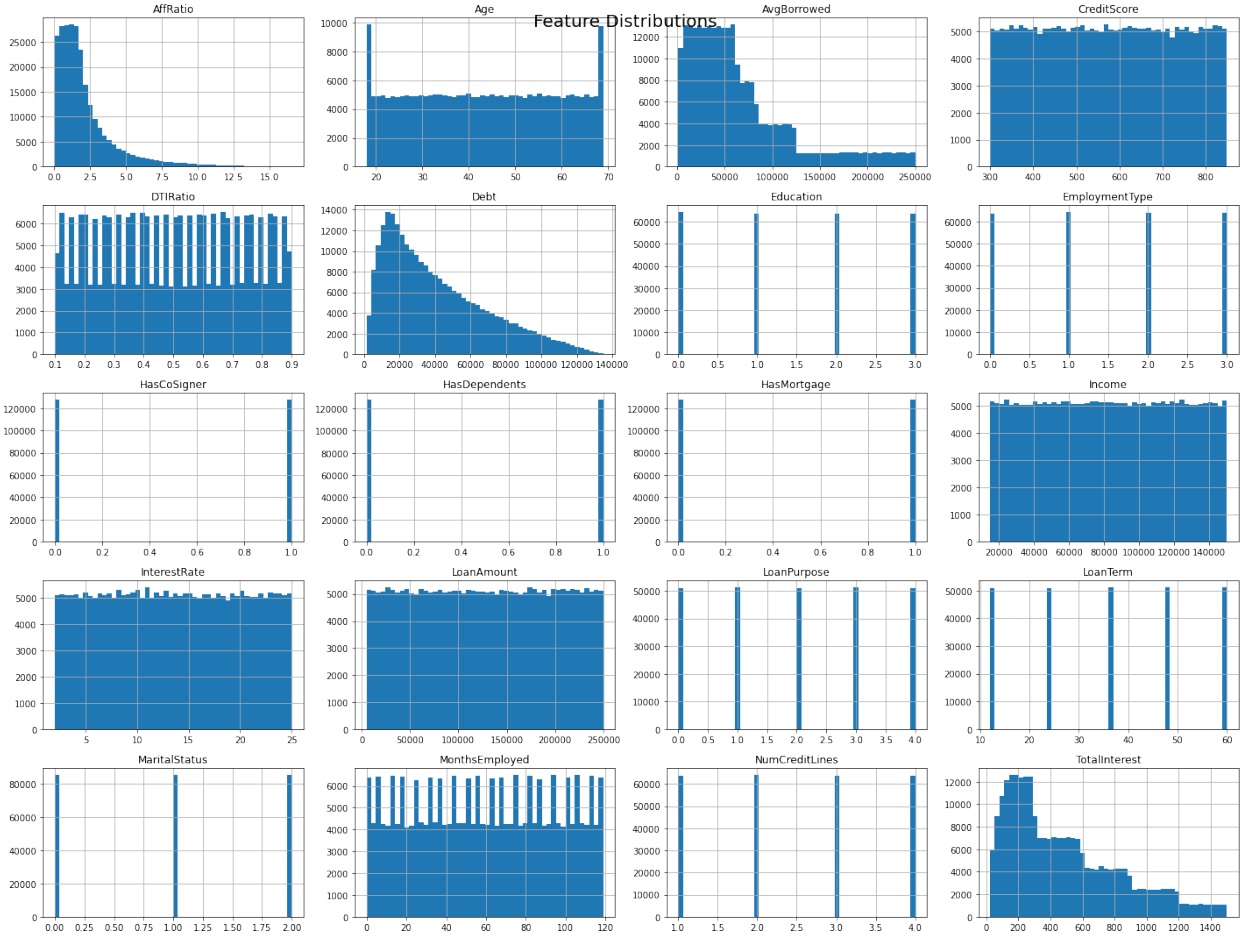

Figure 1: Feature distributions before Box–Cox transformation.

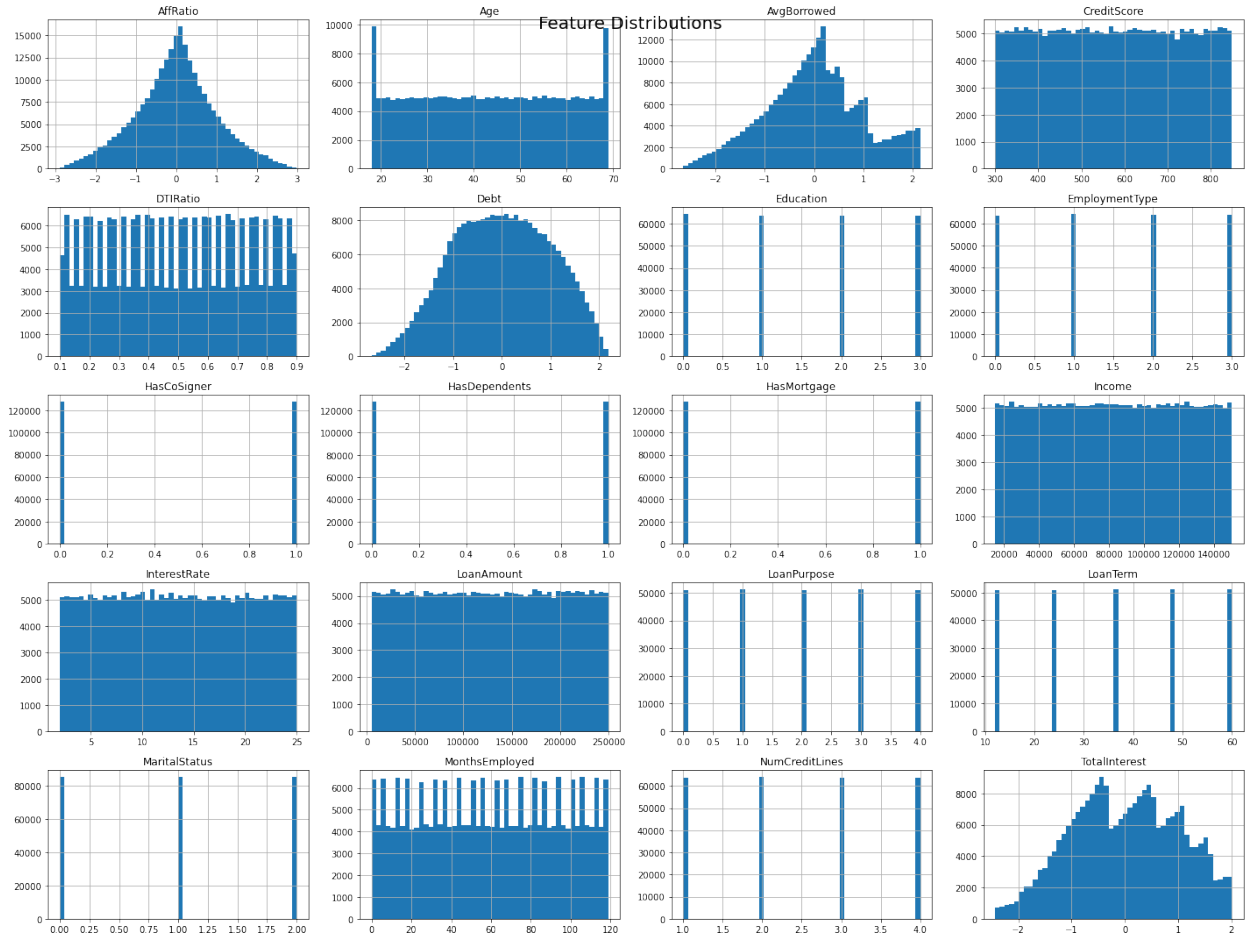

Figure 2: Feature distributions after Box–Cox transformation.

2.3 Encoding Categorical Variables and Addressing Class Imbalance

To feed data into ML models, various preprocessing strategies were used to make the dataset numeric and well-balanced.

Categorical Encoding

All categorical variables were encoded using **label encoding** via scikit-learn's `LabelEncoder`. While one-hot encoding could have preserved more flexibility, tree-based models like XGBoost and structured Neural Networks often work just as well with label-encoded category representations[5].

Handling Class Imbalance

Defaulted loans were significantly outnumbered by non-defaulted loans, which is a common issue in credit risk datasets. To prevent models from defaulting to majority-class predictions, class-specific strategies were used:

– **XGBoost:** Used the built-in `scale_pos_weight` parameter, set as:

$$\frac{\texttt{\# negative samples}}{\texttt{\# positive samples}}$$

This adjusts the gradient updates during boosting to penalize false negatives more heavily.

– **Logistic Regression:** Passed `class_weight="balanced"` during training as a parameter.
– **Neural Network:** Passed `class_weight="balanced"` into scikit-learn's `compute_class_weight` function, which was then passed into the Neural Network's class_weight parameter.

Both approaches helped prevent overfitting to the non-default class, which would have otherwise inflated accuracy but degraded AUC and generalization.

3 Methodology

3.1 Logistic Regression

Logistic Regression was implemented using scikit-learn with L2 regularization and the `liblinear` solver, suitable for small datasets and binary classification tasks. L2 regularization, also known as Ridge regularization, was chosen to prevent overfitting by penalizing large coefficients. The cost function minimized is:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \lambda\sum_{j=1}^{n}\theta_j^2$$

where $\lambda = \frac{1}{C}$, with $C = 1.0$ controlling the inverse of the regularization strength. A balanced class distribution was enforced using `class_weight="balanced"` to combat dataset imbalance, ensuring the model didn't bias toward the majority class [6].

3.2 XGBoost Classifier

The XGBoost classifier [5] was selected due to its high performance on structured/tabular data and its regularization mechanisms. Key parameters included:

– `n_estimators=100`, `learning_rate=0.1` to balance bias-variance tradeoff
– `max_depth=4` to control tree complexity and mitigate overfitting
– `tree_method="hist"` for faster histogram-based training
– `eval_metric="logloss"` to align with binary classification objective
– `scale_pos_weight` $= \frac{\#\text{negative samples}}{\#\text{positive samples}}$, a technique shown to improve performance on imbalanced datasets [6, 7]

These settings reflect a model tuned for generalization and interpretability, while still robust to minority class suppression.

3.3 Feedforward Neural Network (FFNN)

A Feedforward Neural Network (FFNN) was constructed using TensorFlow/Keras, optimized for binary classification with standardized inputs to accelerate convergence. The network architecture comprised:

– Input layer equal to the number of features
– Three hidden dense layers with 64, 32, and 16 neurons respectively
– Dropout rate of 30% after each hidden layer to prevent co-adaptation and overfitting
– Rectified Linear Unit (ReLU) activation defined as $f(x) = \max(0, x)$, chosen for its non-saturating gradient and computational efficiency [8]
– Output layer with a sigmoid activation $\sigma(x) = \frac{1}{1+e^{-x}}$ for probability interpretation

The network was compiled with binary cross-entropy loss and the Adam optimizer [9] (learning rate = 0.001), known for combining the benefits of RMSProp and momentum. Training was conducted over 20 epochs with batch size 32, using class weighting to address imbalance.

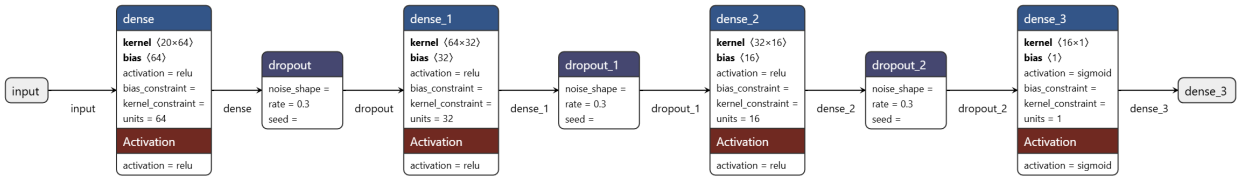

Figure 3: Feedforward Neural Network architecture with three hidden layers, ReLU activation, and dropout for regularization.

4 Evaluation and Results

4.1 Performance Metrics

Given the imbalanced nature of the dataset (88% non-default, 12% default), evaluation focused on metrics that go beyond simple accuracy to provide a more nuanced view of model performance.

- **ROC AUC (Receiver Operating Characteristic – Area Under Curve)**: The primary evaluation metric, ROC AUC measures the model's ability to distinguish between the positive and negative classes across all thresholds. A higher AUC indicates better performance and is really useful for classification tasks and imbalanced data.
- **Accuracy**: Although commonly reported, accuracy can be misleading in imbalanced settings, as predicting all samples as the majority class would still yield high accuracy. It was included for completeness but not used as a primary metric.
- **Precision**: Precision evaluates the proportion of positive identifications that were actually correct (i.e., true positives over all predicted positives). It is crucial when false positives are costly (e.g., wrongly flagging someone as high-risk).
- **Recall (Sensitivity)**: Recall measures the proportion of actual positives that were correctly identified. It is particularly important in this case where failing to identify a defaulter (false negative) can be more harmful than a false positive.
- **F1 Score**: The harmonic mean of precision and recall, the F1 score balances both concerns. It was especially useful when comparing models that had differing precision and recall metrics.

A comprehensive evaluation of each model's performance was performed by combining these metrics, especially in the context of real-world implications of classification errors.

4.2 Confusion Matrices

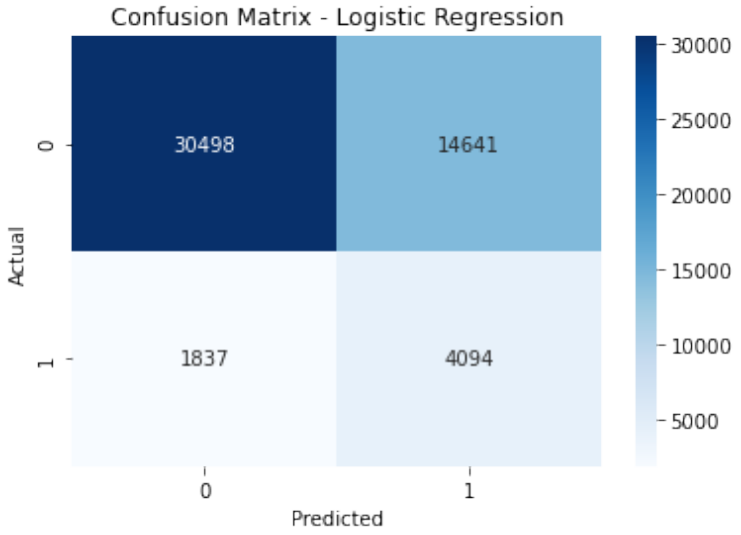

Figure 4: Confusion matrix for Logistic Regression.

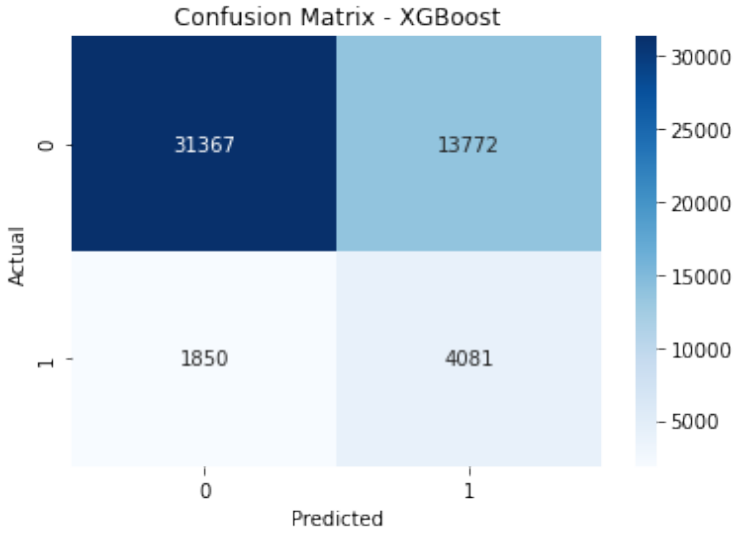

Figure 5: Confusion matrix for XGBoost.

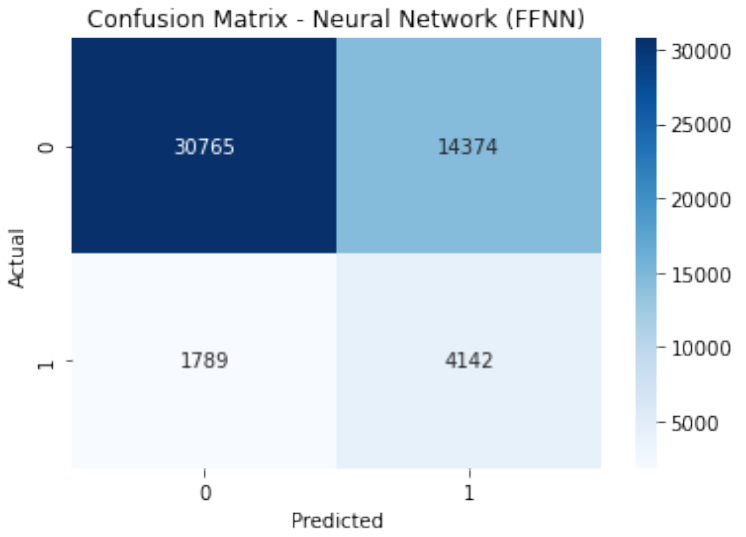

Figure 6: Confusion matrix for Feedforward Neural Network (FFNN).

4.3 Feature Correlations

Post feature engineering, Pearson correlations with the target were computed. Figure 7 visualizes key relationships.

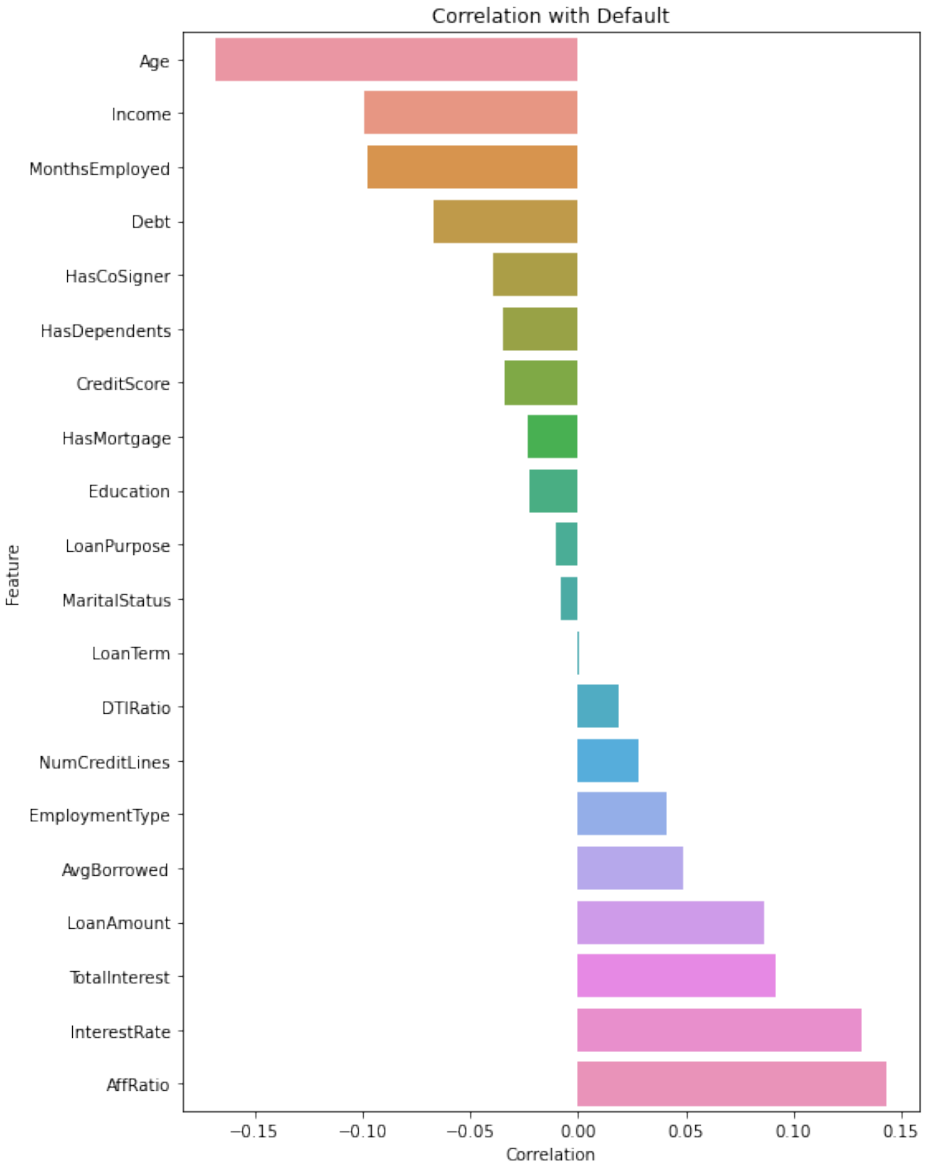

Figure 7: Pearson correlation between features and loan default status. AffRatio, InterestRate, and Age are among the strongest predictors.

Box–Cox transformations reduced feature skewness (e.g., `AffRatio` from 2.33 to 0.001), aiding model convergence though correlation rankings remained stable.

4.4 Model Performance Summary

Model	ROC AUC	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.745659	0.677345	0.218521	0.690271	0.331955
XGBoost	0.758221	0.694106	0.228589	0.688080	0.343172
FFNN	0.758164	0.683513	0.223698	0.698365	0.338855

Table 3: Validation performance comparison across models.

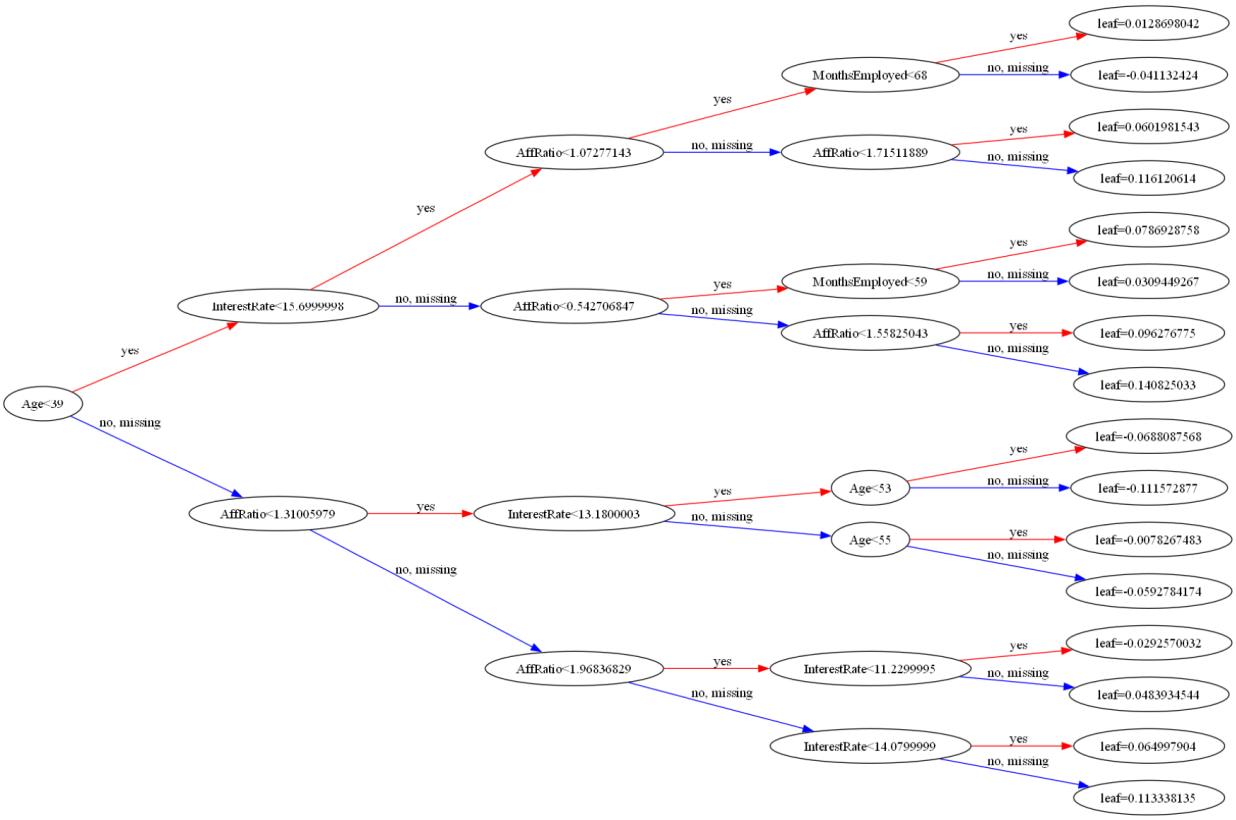

Figure 8: Visual representation of XGBoost model after fitting.

4.5 Feature Importance and Interpretability

XGBoost Feature Importances

The most influential predictors, based on how many times each feature was split on in all the trees, were:

Feature	Importance	Rank
Age	0.249379	1
AffRatio	0.147728	2
InterestRate	0.120047	3
MonthsEmployed	0.067752	4
HasDependents	0.059392	5
HasCoSigner	0.059134	6
EmploymentType	0.047123	7
HasMortgage	0.036152	8
Education	0.033412	9
NumCreditLines	0.031509	10
CreditScore	0.021113	11
MaritalStatus	0.018974	12
LoanPurpose	0.018024	13
Income	0.016566	14
DTIRatio	0.016308	15
LoanAmount	0.013850	16
Debt	0.011415	17
LoanTerm	0.011077	18
TotalInterest	0.010758	19
AvgBorrowed	0.010287	20

Table 4: XGBoost feature importances sorted by magnitude.

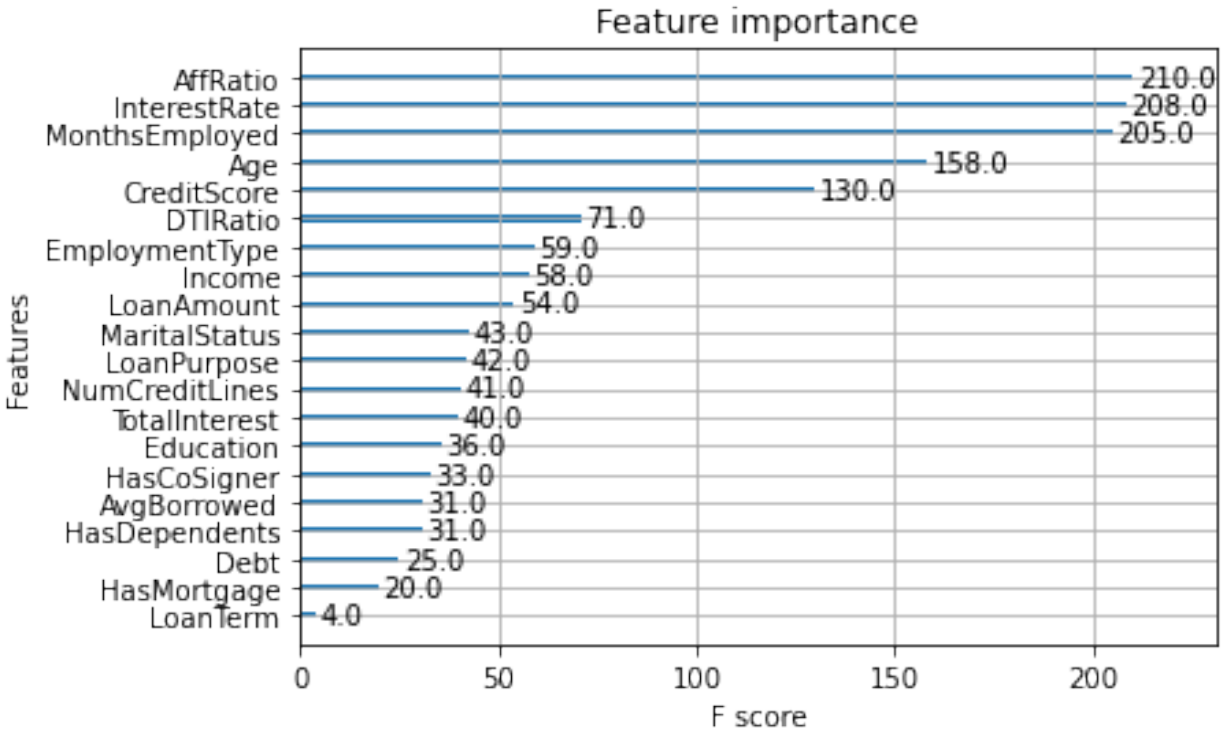

Figure 9: Logistic Regression feature weights.

Logistic Regression Feature Weights

Although logistic regression underperformed, coefficient magnitudes provide interpretability (see Table 5).

Feature	Importance	Rank
EmploymentType	0.149102	1
TotalInterest	0.117167	2
NumCreditLines	0.104088	3
AffRatio	0.103316	4
InterestRate	0.087162	5
HasCoSigner	0.075851	6
HasDependents	0.073878	7
AvgBorrowed	0.053450	8
DTIRatio	0.053149	9
Education	0.045324	10
HasMortgage	0.040633	11
Age	0.034580	12
Debt	0.030601	13

Table 5: Top 13 logistic regression feature importances sorted by magnitude.

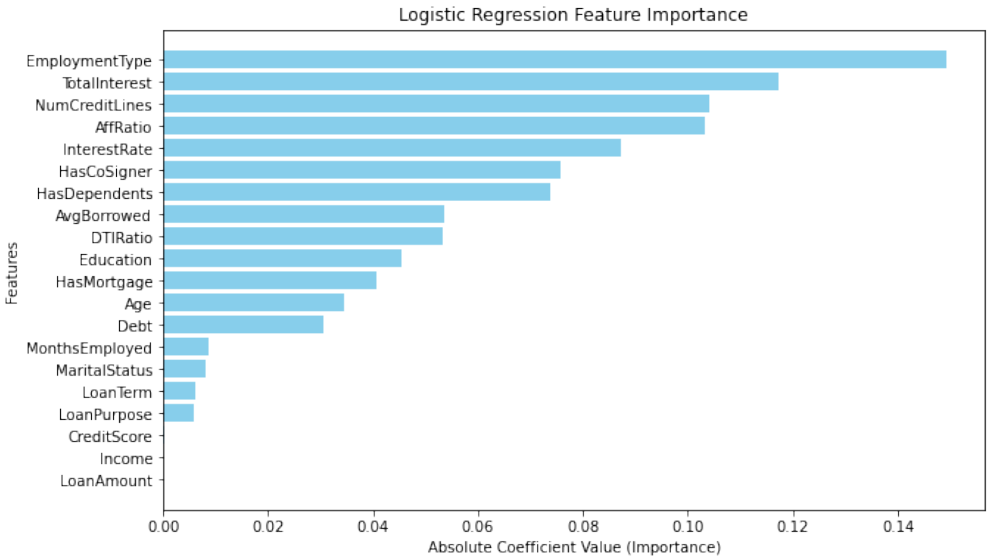

Figure 10: Logistic Regression feature weights.

The low weights on Income, LoanAmount, and CreditScore illustrate the limitation of linear models in capturing complex nonlinear interactions present in the data.

5 Discussion

Strengths: Key advantages of the modeling pipeline include:

– **Effective preprocessing pipeline:** Handling skewed distributions with Box–Cox transformations improved feature distributions and model learning dynamics.
– **Custom feature engineering:** AffRatio and TotalInterest were particularly valuable, surfacing as top predictors in both linear and nonlinear models.

– **Class imbalance mitigation:** By leveraging `scale_pos_weight` and `class_weight="balanced"`, minority class recall was greatly improved.

Limitations: However, several limitations affect generalizability:

– **No temporal dynamics:** All features are static at the time of loan issuance; borrower behavior over time is excluded.
– **Potential label leakage:** Without time-aware splits, certain engineered features (e.g., TotalInterest) might indirectly reflect future events.

Future Work: Several directions could enhance performance and interpretability:

– **SHAP-based interpretability:** Offers per-instance explanations and can clarify complex feature interactions. [10]
– **Model ensembling:** Combining XGBoost with CatBoost or LightGBM may improve generalization.
– **Behavioral data:** Incorporating time-series features such as payment consistency and monthly balances.
– **Model calibration:** Using Platt scaling to better align predicted probabilities with real-world default rates.

6 Conclusion

The work shows that, after careful preprocessing, i.e., feature engineering, skewness correction, and class imbalance addressing, structured loan data can provide reliable default predictions. The result of 0.76 ROC AUC is already a good baseline; further gains may be based on interpretability improvements as well as the use of richer and more dynamic data sources.

References

[1] Sami Mestiri. Credit scoring using machine learning and deep learning-based models. *Data Science in Finance and Economics*, 4(2):236–248, 2024.

[2] Xinyu Zhang, Tianhui Zhang, Lingmin Hou, Xianchen Liu, Zhen Guo, Yuanhao Tian, and Yang Liu. Data-driven loan default prediction: A machine learning approach for enhancing business process management. *Systems*, 13(7), 2025.

[3] Rambod Rahmani, Marco Parola, and Mario G. C. A. Cimino. A machine learning workflow to address credit default prediction, 2024.

[4] Jason Osborne. Improving your data transformations: Applying box-cox transformations as a best practice. *Pract Assess Res Eval*, 15:1–9, 01 2010.

[5] 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*, KDD '16, page 785–794. ACM, August 2016.

[6] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[7] Yi Chen, Youzhong Dong, and Wen Liu. Prediction of credit default based on the xgboost model. *Applied and Computational Engineering*, 96:85–92, 11 2024.

[8] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, 2010.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[10] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.