International Journal of Data Science and Analysis
2026; 12(3): 37-51
http://www.sciencepublishinggroup.com/j/ijdsa
doi: 10.11648/j.ijdsa.20261203.11

Determining Loan Eligibility in the Banking Sector Using a Hybrid Model of Support Vector Machine and Extreme Gradient Boosting

James Githaiga Muhoro*, Harun Mwangi Gitonga, Josephine Njeri Ngure

School of Pure and Applied Sciences, Kirinyaga University, Kerugoya, Kenya

Email address:

jamesmuhoro95@gmail.com (James Githaiga Muhoro)

To cite this article:

James Githaiga Muhoro, Harun Mwangi Gitonga, Josephine Njeri Nguree. (2026). Determining Loan Eligibility in the Banking Sector Using a Hybrid Model of Support Vector Machine and Extreme Gradient Boosting. *International Journal of Data Science and Analysis, 12*(3), 37-51. https://doi.org/10.11648/j.ijdsa.20261203.11

Received: 20 May 2026; **Accepted:** 30 May 2026; **Published:** 3 July 2026

Abstract: The loan prediction models are ever-changing due to changes in technology, whereby financial institutions are adopting various technologies to automate the loan process. The surge of loan applicants with diverse attributes has accelerated the need for machine learning models which can incorporate different applicant attributes and improve accuracy in determining loan eligibility. While the use of individual machine learning models was more robust and accurate, these models have some limitations that may hinder the achievement of optimal results when establishing loan eligibility. Thus, the need to combine two or more individual models and leverage their strengths to improve accuracy and robustness. This study aimed to develop an eXtreme Gradient Boosting (XGBoost)-Support Vector Machine (SVM) hybrid model to determine loan eligibility in the banking sector. This work utilized secondary financial dataset from Google Kaggle. The dataset was preprocessed, transformed and used to train and test the models. Evaluation metrics, namely accuracy, precision, F1-score, recall and Receiver Operating Characteristics-Area Under Curve (ROC-AUC), were used to evaluate the performance and reliability of the XGBoost, SVM and the XGBoost-SVM hybrid model. The XGBoost-SVM hybrid model posted strong performance metrics results with accuracy of 0.78, precision of 0.27, a balanced recall of 0.49, F1-Score of 0.34 and AUC-ROC curve of 0.74. It was therefore evident that the hybrid model was able to leverage the standalone model's strength for better performance in determining loan eligibility.

Keywords: Loan Eligibility, Machine Learning, XGBoost, Support Vector Machine, Hybrid Model

1. Introduction

Credit facility plays a fundamental role in individuals, firms and economies of the countries by bridging the gap between available financial needs and cash reserves. It facilitates ownership of homes, expansion of business startups to large corporations, entrepreneurship, consumption, education and smooth flow of operational liquidity. As such, banks play a pivotal intermediary role of allocating investment capital from savers to loan applicants; hence, the efficiency, profitability and stability of the entire allocation process is a core function of the banks [1]. The banks are entitled to carry out credit assessment to determine the loan eligibility and protect the lender from loan default, and safeguard the borrower from over-indebtedness.

Over the period, loan eligibility was a subjective process that relied on the experience, relationship and intuition between the lender and the borrower, supported by the quantitative historical analysis. The loan approval and the amount were highly reliant on the previous relationship, pledged and tangible assets and the borrower's repayment discipline. The development of the credit scoring engines, such as auto score and mortgage score, using statistical data, revolutionized the loan approval process since these models used discriminant statistical analysis characterizing borrowers on several factors such as age, default history, marital status, employment history, income and age of the borrower [2]. These factors enabled the generation of a numerical credit score, thus enhancing scalability, efficiency, standardization and reduction

of human bias when determining loan eligibility.

Recently, the banking industry has experienced a paradigm shift, which highlights the disadvantages of traditional credit scoring models. The rise of fin-tech companies, the stiffening of regulatory scrutiny, the evolution of digital loans, and the explosion of consumer demographic information have led to the emergence of a competitive, dynamic, and complex financial environment. Consequently, this has led to credit mispricing, making it difficult to approve the loan [3]. As such, banking sectors need more robust, accurate and stable credit pricing models to streamline the loan application process to enhance institution stability and reduce over-indebtedness of the borrower.

The traditional approach faced significant challenges, such as a linear relationship, whereby they struggled to capture non-linear relationships between the variables of different loan applicants. Models were also unable to analyze an imbalanced data set, leading to inaccuracies and low recall when determining the loan eligibility of the borrower. These models were insensitive to economic fluctuations since they were not agile enough to adapt to various market changes and sudden economic impacts. They were also susceptible to human bias since the loan approval is done manually, which could lead to discrimination during the loan approval process.

The development and application of machine learning models set the stage for solving most of the challenges linked to traditional models. Machine learning techniques entail classical statistical techniques in the development of algorithms that are improved sequentially through training and experience. Machine Learning models can improve the automation of predictive accuracy and also harness different data sources, offering the outcome on loan eligibility [4]. However, the use of a single machine learning model poses a set of limitations when working with big banking data, which are related to real-world applicability, technical constraints, and data dependency.

The limitations of the single machine learning models and data constraints can lead to unreliable predictions due to low-quality and unrepresentative data. Additionally, generalization and technical limitations can lead to over-fitting or under-fitting of the model, thus making the model unable to capture complex relationships such as an applicant with different sources of income and with many dependents [5]. They can also posit narrow expertise, thus performing exceptionally well on a specific data component while failing to transfer the expertise to other data components. These limitations can compromise the loan approval rate in banks, which can end up affecting both the borrower and the lender.

As a result of the limitations of using single machine learning models, the development of a hybrid machine learning model is paramount during loan prediction due to its robustness, interpretability, and stability when working with big banking data. The hybrid machine learning model integrates two or more distinct models to leverage their individual strengths, thus mitigating individual challenges. These hybrid models showcase a powerful way for loan prediction by providing the accuracy required for risk management, ethical lending, and regulatory requirements. Carrying out research in this area was imperative, focusing on the development of hybrid models for eXtreme Gradient Boosting (XGBoost) and Support Vector Machine (SVM) models and using performance metrics that are robust, accurate, and ultimate in loan prediction in banking sectors.

2. Literature Review

2.1. Introduction

The banking sector holds various products such as reserves, treasuries, and loans. Their primary role is to be intermediaries between depositors and borrowers. loans are the main source of income for banks [6]. The amount of profit realized by banks depends on the default rate or the ability of the customers to repay. This section discusses the conventional and modern methods used in loan prediction and their advantages and disadvantages.

2.2. Traditional Models in Loan Prediction

Traditional loan prediction methods employed manual audits from historical data such as credit history and repayment period, employment status, and amount of income and collateral [3]. Employment status represents an essential factor defining the applicant's regular income source, while the income amount reflects the financial strength to repay a specified loan amount in a given timeline. Credit history provides valuable insights into the evaluation of the applicant's creditworthiness. Collateral, on the other hand, act as security to mitigate risk in case the borrower defaults on the loan. Although these parameters help evaluate the borrower's eligibility for a loan, it faces several challenges.

Manual loan approval is labour-intensive, involves a lot of documentation and paperwork, and therefore time-consuming [7]. During peak seasons, financial institutions may incur additional costs in hiring more personnel or lead to customer dissatisfaction due to delays in processing [3]. Decision-making based on manual evaluation and subjective human judgment may result in inequalities in loan approvals due to potential stereotypes, bias, or favouritism. Moreover, human error is inevitable. According to [8], there are two main types of errors in loan assessment. The first type involves categorizing a desirable loan candidate as bad, a False Negative. This denies the financial institution a low-risk and profitable opportunity and leads to poor customer relations. The second type of error occurs when a high-risk borrower is mistakenly recognized as creditworthy, a False Positive. Consequently, the loan is approved, leading to high non-performing loans, financial losses, and inadequate risk mitigation. These challenges reveal the need for automated and data-driven systems to predict loan eligibility.

2.3. Machine Learning in Loan Prediction

Machine learning is a function of artificial intelligence whose systems classify and predict outcomes automatically using algorithms and statistical models. It relies on trained datasets rather than explicit instructions. With technological advancements, several researchers have developed machine learning and deep learning models, which have been attributed to several advantages over conventional methods. Machine learning models are flexible in adapting to new and big datasets, learning from them and analyzing trends and patterns reflecting the customer's creditworthiness. It has the ability to handle complex and diverse datasets, leading to fast loan processing and more accurate and dependable credit approval outcomes. This reduces the burden to the credit officers, saves on cost and improves customer relationship and satisfaction. On the other hand, the accuracy and efficiency of machine learning depend on the initial data. Incorrect data or programming leads to unreliable and inaccurate output.

Several studies employed different machine learning techniques and demonstrated their effectiveness to some extent. A study on loan default prediction [9] achieved accuracy and recall of 64%, precision and F1 score of 63% and 62% respectively using decision trees. Logistic regression and Naïve Bayes attained 63% in precision, recall and accuracy and an F1 score of 62%. Loan default was predicted sing XGBoost,Light Gradient Boosting Machine (LightGBM), Random Forest and Decision Tree [10]. Highest accuracy, precision and F1 score of 97.64%, 97.47% and 95.03%, respectively, was achieved by LightGBM. XGBoost exhibited the highest recall and Receiver Operating Characteristics-Area Under the Curve (ROC AUC) of 93.07% and 99.17%, respectively. Decision tree and random forest had an accuracy of 94.08% and 93.94%, respectively and 86.72% and 89.33% in precision, respectively. Overall, LightGBM emerged as the most efficient model in performance, followed by XGBoost.

2.3.1. XGBoost

XGBoost is a scalable, portable and efficient boosting-based tree algorithm. It focuses on utilizing decision trees as base learners. Gradient boosting involves combining many weak learners to reinforce a single weak learner. The sequential combination of multiple weak learners minimizes prediction errors, enhancing the performance of the XGBoost model. Large data sets are quickly trained by a built-in parallel processing. It mitigates overfitting by splitting leaves in the same layer through regularization methods such as penalizing leaf scores by L1 (Lasso) and L2 (Ridge) [11]. This feature improves the model's generalization ability. It implies the capability to choose significant characteristics over trivial ones by default; hence, the model can efficiently handle sparse and high-dimensional data [12]. Ensures high computational efficiency through cache optimization, which promotes memory data storage, reducing input/output operations. Technological innovations utilized by the model, such as frequency, gain and cover metrics, foster practicability and interpretability of the model [13]. The model's ability to optimize newly added trees and minimize loss functions improves its overall accuracy.

A study [14] in 2023 determined the performance of logistic regression, decision tree, lightGBM and XGBoost in loan default prediction. Similar to the [10] study, LightGBM recorded an accuracy of 81.04%, 72.13% AUC, and 57.51% precision, which were higher than in other models. XGBoost ranked second in performance with 80.98% accuracy, 72.06% AUC and 55.83% precision. The decision tree performed poorly in accuracy (63.17%) and precision (30.15%), as well as logistic regression, whose accuracy and precision were 65.55% and 31.10%, respectively.

In a different research, the performance of Gradient Boosting, XGBoost, Random Forest, and LightGBM in loan default prediction was investigated [11]. Gradient Boosting was regarded as the most effective model following its highest evaluation metrics in 80.84% accuracy, 80.21% recall and 80.84% F1 -score. Likewise, this model outperformed other models in accuracy (89.5%) and F1 score (90.5%) in a similar study by [7]. However, Gradient Boosting is sequential in nature, thus time-consuming during training as compared to LightGBM and Random Forest. Moreover, the model is predisposed to overfitting, hence the need to employ cross-validation to improve generalization [11]. XGBoost outperformed other models in AUC score (97.14%). LightGBM and Random Forest showed lower precision and recall.

2.3.2. Support Vector Machine

The Support Vector Machine (SVM) was created in 1963, and it is ranked among the best machine learning models[15]. This model focuses on dividing the characteristic space using hyperplanes. The hyperplanes act as the most suitable boundaries flanked by classes; thus, several subspaces are created as labels [16]. Support vectors, extreme vector points in the SVM algorithm, help to create suitable hyperplanes [15]. Hyperplane optimization contributes to low generalization errors, hence minimal overfitting chances. [17] postulates the effectiveness of SVM when working with relatively small and high-dimensional spaces. SVM is advantageous in terms of memory as it requires less memory storage [15]. However, in cases where there is a high amount of noise in the dataset, the model performs inefficiently. Moreover, SVM fails to compute the prediction confidence level, increasing the computation cost when an alternative method, such as k-fold cross-validation, is utilized [17].

Research on predictions of loan defaults in Kenya utilized SVM and logistic regression models [18] . Performance comparison between the two models, SVM with linear kernel, showed better performance than logistic regression. The accuracy and precision of SVM (linear kernel) were 88.29% and 87.85%, respectively, while logistic regression achieved 77.27% accuracy and 84.40% precision. In a separate study, SVM with sigmoid kernel achieved worst performance (83%) while polynomial kernel obtained the best performance (97%) in accuracy [19]. However, the precision and recall were very low due to the unbalanced datasets. A study on using SVM

with radial basis function kernel to optimize loan approvals in the banking sector was conducted in 2024 [20]. Comparison of SVM with decision trees and logistic regression revealed outstanding performance of SVM with 87% accuracy, 84% precision, 89% recall, 86.5% F1 score and 0.92 ROC-AUC score.

2.4. Hybrid Machine Learning Approach

Hybrid models combine two or more machine learning models, aiming to enhance interpretability and performance. It is designed to provide optimal solutions to complex and diverse data modalities faced by single models. Several researchers have employed hybrid models in varied domains. For instance, a hybrid model containing LightGBM, Categorical Boosting (CatBoost) and XGBoost was utilized to create a credit risk prediction model [21]. The study utilized L1, Random Forest, Extra Trees, Analysis of Variance (ANOVA), Weight of Evidence (WOE) and Feed Forward feature selection to create hybrid feature selection. Synthetic Minority Oversampling (SMOTE) was used to balance the imbalanced datasets. The hybrid model outperformed standalone models for medium and large data sets, achieving an AUC of 87% and 96%, respectively. The F1 score and accuracy were also higher than for the individual models. However, small datasets led to lower metric values than single classifiers. Hence, it is difficult to generalize hybrid stacked models.

In 2025, a hybrid approach was used to predict credit risk in financial institutions [22]. The study combined LightGBM and CatBoost to create the hybrid model. This model outperformed single models with 88.04% accuracy, 91.77% precision, 94.54% AUC score, and 87.51% F1-score. The hybrid model combining the two models brings out a synergistic relationship, improving computational efficiency, accuracy and high generalization ability of varied tasks and datasets [23].

Apart from financial institutions, hybrid machine learning has been recommended in other industries. For instance, combined reinforcement and supervised learning algorithms was employed to increase precision and accuracy in predicting a machine?s remaining useful life [24]. The hybrid integrated Q-learning and multi-layer perception algorithms achieved a 15% increase in accuracy over single models. Hybrid models promote early predictions of machine downtime, leading to low maintenance costs, enhanced reliability, and time saving [24]. Conclusively, hybrid models show greater potential than single models in varied applications. The synergistic relation helps to solve one model's weakness while complementing the other model(s) with its strength, hence high performance.

3. Materials and Methods

3.1. Introduction

This section entails reviewing the research designs and their justifications, data sources and their relevance to the problem, data collection and analysis methods, and their justification in relevance to the problem statement.

3.2. Data Source and Data Attributes

This study used a secondary data set consisting of loan application records of the year 2023 taken from a banking institution that is publicly available on the Google Kaggle repository. The loan dataset comprised the following attributes: age, loan amount, income, credit score, months employed, interest rate, loan period, education, marital status, employment type, has dependents, has a cosigner and loan purpose. These attributes are significant in determining the applicant's eligibility for a loan.

3.3. Data Preprocessing

3.3.1. Data Cleaning

The dataset was explored by checking the missing values and duplicates.

3.3.2. Feature Transformation and Splitting

Categorical encoding was carried out, which entailed one-hot encoding for nominal variables such as loan purpose and label encoding for ordinal variables such as account status in XGBoost. Additionally, numerical minimum and maximum scaling was carried out for the SVM model to ensure that all the variables had a mean of 0 and a variance of 1, respectively. Scaling enhanced consistency in hybridization. Data was split into training and testing data, that is, 80% and 20%, respectively.

3.3.3. Imbalance Testing

The data was tested for class imbalance and mitigated using SMOTE, which generated the synthetic samples for the default class.

3.4. Model Development and Hybridization

3.4.1. XGBoost Model

This model was used due to its ability to handle the missing data type, highly effective in detecting outliers and providing feature importance by quantifying how much each feature contributed to the prediction model through the calculation of metrics such as frequency, gain, and cover.

Let define loan eligibility dataset as;

$$M = (X_i,\ Y)_i^N = 1 \tag{1}$$

Where: $X_i \in R^d$ represents the feature vector for the $i^(th)$ loan applicant such as Income, loan amount, employment history, and credit score. The feature matrix will be represented as $X_i \in R^{m\ x\ n}$

$y_i \in (0, 1)$ represent the loan eligibility outcome which is binary outcome.

N represent the number of samples

The model objective function will be:

$$LXGB = i = \sum \sum_{i=1}^{n} [l(y_i,\ y^{i(t)}) + \phi(ft)] \tag{2}$$

Where: $l(y_i,\ y^{i(t)})$ is the loss function, $y^{i(t)} = \sum_{k=1}^{t} fk(xi)$ represents prediction in t and fk represents decision tree. $\phi(ft)$ represents the regularization. Logistic loss:

$$l(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}) \tag{3}$$

The regularization term was directly included in the XGBoost to avoid overfitting. This is given by;

$$\phi(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{4}$$

Where T is the leaves number in the tree, w_j is the prediction value/ weight of the j^{th} leaf.

The generation of leaf output was:

$$zi = [leaf_{i1}, leaf_{i2}, \ldots, leaf_{iK}] \tag{5}$$

Where $leaf_{ik}$ is the leaf index for tree k and K is the number of trees

The feature transformation was:

$$f_{XGB}(x_i) = \ [y^{iXGB}, leaf_{(xi)1}, leaf_{(xi)2}, \ldots, leaf_{(xi)K}] \tag{6}$$

Where f^{iXGB} represents the predicted probability

3.4.2. Support Vector Machines (SVM)

This model was used for its ability to handle complex data; it was also effective in giving clear margin separation as well as being sensitive to kernel choice and noise.

$$X' = [\phi_{XGB}(X),\ X_{original}] \tag{7}$$

Where: $X' \in\ R^{n\times(m+K+1)}$

The SVM primal problem was for minimal solution

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{n} n\xi i \tag{8}$$

$$y_i(W^T \phi_{SVM}(x') + b) \geq 1 - \xi i \tag{9}$$

Where: ϕ_{SVM} represents the SVM kernel mapping.
C represent the regularization parameter
ξi represent slack variables
The maximum solution is given by:

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x'_i x'_j) \tag{10}$$

Where $0 \leq \alpha_i \leq\ c$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{11}$$

The decision function for SVM is given by:

$$f_{svm}x = sign(\sum_{i\in SVM} \alpha_i y_i K(x'_i x' + b) \tag{12}$$

The study employed linear kernel whose equations are per equations (13) and (14).

$$K(x, y) = x^T y \tag{13}$$

$$K(x, y) = \exp\left(-\gamma||x - y||^2\right) \tag{14}$$

Where γ controls the spread of the kernel.

3.4.3. Hybrid Model Development

The foundation of the hybrid system involved training the individual models. In the XGBoost model, loan data was split into training and validation sets. Using the XGBoost classifier, the model was initialized and trained using the binary objective function. Similarly in SVM model, data was divided into training and testing tests. Using linear kernel, the model was initialized and trained using training data. The performance of both models was then optimized separately using random search with cross validation.

The construction of the hybrid model involved integrating XGBoost and SVM through a stacking ensemble learning strategy. During stacking, SVM and XGBoost predictions were used as input variables for a meta-classifier, and the model optimally combined the base models' predictions. The SVM and XGBoost models were trained independently, and then the hyper-parameters were optimized using cross-validation.

The hybrid XGBoost-SVM were transformed x_i from the original input to a new feature:

$$zi = \Psi(xi) = [f1(xi), f2(xi), \ldots, fT(xi)] \tag{15}$$

The customized model was as follows:

$$P(loan\ approval/x) = \sigma(\beta 1 fXGB(x) + \beta 2 fSVM(x')) \tag{16}$$

The final loan eligibility decision was given by;

$$\hat{Y} = \{^{1,\ f_{hybrid}(x)\ \geq\ 0\ (Approved)}_{0,\ otherwise\ (Rejected)} \tag{17}$$

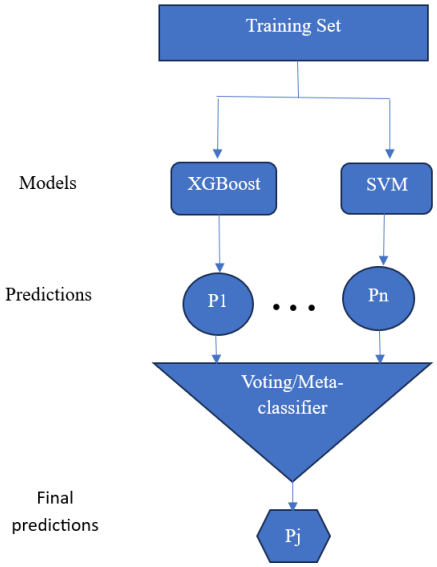

Figure 1. *XGBoost+SVM Hybrid model.*

3.5. Model Evaluation Metrics

The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) was used to measure the ability of the model to discriminate between classes across all thresholds.

The F1-score which is the harmonic mean of the recall and precision, was significant when the probability of false positives (rejecting a good client) and false negatives (missing a default) was high.

$$F1 - score = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \tag{18}$$

Accuracy was used to establish the overall correct prediction rate.

$$Accuracy = \frac{\underset{Positives}{True} + \underset{Negatives}{True}}{\underset{Positives}{True} + \underset{Negatives}{True} + \underset{Positives}{False} + \underset{Negatives}{False}} \tag{19}$$

Precision was imperative in measuring predicted defaults against the actual defaults in loan prediction.

$$Precision = \frac{True\ Positives}{(True\ Postives + False\ Positives)} \tag{20}$$

Recall assisted in measuring proportion of actual default that are accurately identified

$$Recall = \frac{(True\ Positives)}{(True\ Positives + False\ Negatives)} \tag{21}$$

Specificity measured the proportion of actual non-defaults that are correctly identified.

$$Specificity = \frac{True\ Negatives}{True\ Negatives + False\ Negatives} \tag{22}$$

A confusion matrix was used to visualize and give a performance summary in a tabular representation as shown in Table 1 [14].

Table 1. *Confusion Matrix.*

Confusion Matrix	Positive (Actual)	Negative (Actual)
Positive (Predicted)	True Positive (TP)	False Positive (FP)
Negative (Predicted)	False Negative (FN)	True Negative (TN)

For model comparison the statistical test, for instance, the t-test was carried out on cross-validation to check the existence of significant performance difference.

4. Results and Discussion

4.1. Data Preprocessing and Splitting

Raw dataset was converted into a dataset that machine learning models could comprehend. The loan default dataset was loaded in Jupyter notebook in Google Colab and various python libraries such as pandas, matplotlib, numpy, sklearn and seaborn were imported (Figure 2).

```
[2]
✓ 0s
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_validate, RandomizedS
from imblearn.over_sampling import SMOTE
from xgboost import XGBClassifier
from sklearn.metrics import (classification_report, confusion_matrix,
                             roc_auc_score, roc_curve,
                             precision_recall_curve, average_precision_score,
                             ConfusionMatrixDisplay)
from sklearn.svm import SVC, LinearSVC
from sklearn.calibration import CalibratedClassifierCV
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_predict
```

Figure 2. *Python Libraries.*

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 255347 entries, 0 to 255346
Data columns (total 18 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   LoanID          255347 non-null  object
 1   Age             255347 non-null  int64
 2   Income          255347 non-null  int64
 3   LoanAmount      255347 non-null  int64
 4   CreditScore     255347 non-null  int64
 5   MonthsEmployed  255347 non-null  int64
 6   NumCreditLines  255347 non-null  int64
 7   InterestRate    255347 non-null  float64
 8   LoanTerm        255347 non-null  int64
 9   DTIRatio        255347 non-null  float64
 10  Education       255347 non-null  object
 11  EmploymentType  255347 non-null  object
 12  MaritalStatus   255347 non-null  object
 13  HasMortgage     255347 non-null  object
 14  HasDependents   255347 non-null  object
 15  LoanPurpose     255347 non-null  object
 16  HasCoSigner     255347 non-null  object
 17  Default         255347 non-null  int64
dtypes: float64(2), int64(8), object(8)
memory usage: 35.1+ MB
```

Figure 3. *The dataset information.*

```
[ ]: df.isnull().any()
```

[9]:

	0
LoanID	False
Age	False
Income	False
LoanAmount	False
CreditScore	False
MonthsEmployed	False
NumCreditLines	False
InterestRate	False
LoanTerm	False
DTIRatio	False
Education	False
EmploymentType	False
MaritalStatus	False
HasMortgage	False
HasDependents	False
LoanPurpose	False
HasCoSigner	False
Default	False

dtype: bool

Figure 4. *Check for missing values.*

The dataset comprised 255,347 rows and 18 columns (Figure 3). It had 2 float features, 8 integer features and 8 object features. It had no missing values and duplicates (Figure 4).

Figure 5 shows a clear description of how the features in the dataset were aligned. All variables had equal counts supporting the evidence of the absence of missing values. It also indicates the mean, standard deviation, minimum, maximum and quartiles of every feature.

```
df.describe()
```

	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRatio
count	255347.000000	255347.000000	255347.000000	255347.000000	255347.000000	255347.000000	255347.000000	255347.000000	255347.000000
mean	43.498306	82499.304597	127578.865512	574.264346	59.541976	2.501036	13.492773	36.025894	0.500212
std	14.990258	38963.013729	70840.706142	158.903867	34.643376	1.117018	6.636443	16.969330	0.230917
min	18.000000	15000.000000	5000.000000	300.000000	0.000000	1.000000	2.000000	12.000000	0.100000
25%	31.000000	48825.500000	66156.000000	437.000000	30.000000	2.000000	7.770000	24.000000	0.300000
50%	43.000000	82466.000000	127556.000000	574.000000	60.000000	2.000000	13.460000	36.000000	0.500000
75%	56.000000	116219.000000	188985.000000	712.000000	90.000000	3.000000	19.250000	48.000000	0.700000
max	69.000000	149999.000000	249999.000000	849.000000	119.000000	4.000000	25.000000	60.000000	0.900000

Figure 5. *Descriptive Statistics Summary.*

There was a class imbalance between the number of defaults and non-defaults as indicated in the histogram below (Figure 6). SMOTE technique was employed during class modeling to mitigate this imbalance.

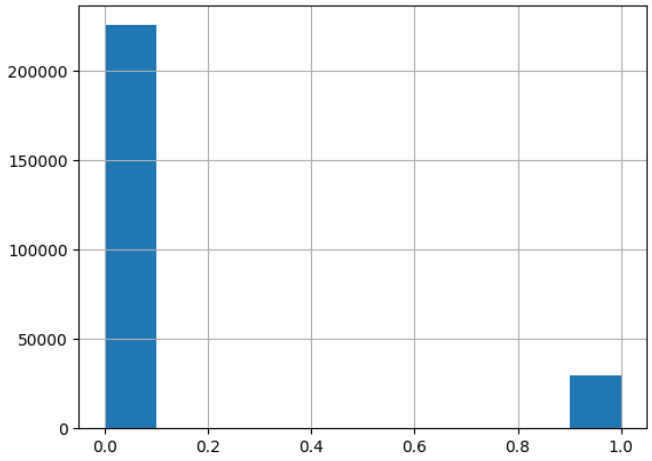

Figure 6. *Imbalance between number of defaults and non-defaults.*

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=SEED, stratify=y
)
print(f"Train: {X_train.shape} | Default rate: {y_train.mean():.3f}")
print(f"Test:  {X_test.shape} | Default rate: {y_test.mean():.3f}")
```

```
Train: (204277, 24) | Default rate: 0.116
Test:  (51070, 24) | Default rate: 0.116
```

Figure 7. *Training and Testing Dataset.*

The Figure 7 above indicates the target label and feature matrix with x and y values showing default or not. Also, 20% of the data was used for testing, while the use of stratify=y ensures both test and training sets maintain the same proportion of non-default and default from the original dataset. The training set had 204277 features with 24 rows, with a default rate of 0.116, while the test set contained 51070 features with 24 rows and a default rate of 0.116. The test and training set had the same default rate to confirm that the stratification worked correctly. The stratification was significant in imbalance classification, and without it, the test set could have a lower or higher default rate, leading to a wrong result during model evaluation. The same percentages in default rate show that the test set indicates a representative sample, hence performance metrics should work well on new data.

```
scaler = StandardScaler()
X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test[num_cols]  = scaler.transform(X_test[num_cols])
```

```
smote = SMOTE(random_state=SEED)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)
```

```
X
```

	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRatio	HasMortgage	...	Education_PhD
0	56	85994	50587	520	80	4	15.23	36	0.44	1	...	False
1	69	50432	124440	458	15	1	4.81	60	0.68	0	...	False
2	46	84208	129188	451	26	3	21.17	24	0.31	1	...	False
3	32	31713	44799	743	0	3	7.07	24	0.23	0	...	False
4	60	20437	9139	633	8	4	6.51	48	0.73	0	...	False
...	...	...	...	...	...	...	...	...	...	...	...	...
255342	19	37979	210682	541	109	4	14.11	12	0.85	0	...	False
255343	32	51953	189899	511	14	2	11.55	24	0.21	0	...	False
255344	56	84820	208294	597	70	3	5.29	60	0.50	1	...	False
255345	42	85109	60575	809	40	1	20.90	48	0.44	1	...	False
255346	62	22418	18481	636	113	2	6.73	12	0.48	1	...	False

255347 rows × 24 columns

Figure 8. *Feature Scalin.*

Feature scaling using a standard scaler was initialized to train the numerical columns, as shown in Fig.8. Also, the mean and standard deviation of each column were computed and standardized to a mean of 0 and a variance of 1. Additionally, scaler transforms the test set to ensure the test data is well scaled by the training set parameters to prevent leakage of data.

The class imbalance was handled using SMOTE (Synthetic Minority Over-sampling Technique) by generating synthetic samples in the minority class, leading to a balanced dataset. Applying a scaler before SMOTE was crucial since SMOTE operates in feature space, and it performed well on well-scaled features.

4.2. Modeling

This step involved developing models and forecasting the outcomes of the models. Support Vector Machine (SVM) and XGBoost models were developed as shown in Figure 9. The calibrated classifier was used in SVM to increase the probability calibration on the decision function values that are not well calibrated. Class weight was crucial in adjusting weights that were inversely proportional to class frequencies, thus handling class imbalance. The XGBoost model had 200 estimators, which were the number of boosting rounds (trees), with a 0.05 learning rate that shrinks the contribution of every tree, helping in generalization. Additionally, the scaling weight was vital in handling imbalance during loss calculation by weighing the positive class. Therefore, both models account for imbalance in the dataset.

```
[35]
✓ 0s
    models = {
        'SVM (Linear)': CalibratedClassifierCV(
            LinearSVC(class_weight='balanced', random_state=SEED, max_iter=2000)
        ),
        'XGBoost': XGBClassifier(
            n_estimators=200, learning_rate=0.05, max_depth=5,
            scale_pos_weight=(y_train == 0).sum() / (y_train == 1).sum(),
            random_state=SEED, eval_metric='logloss', verbosity=0
        )
    }
```

Figure 9. Modeling.

The SVM and XGBoost machine learning models were trained, and the trained models were stored in the dictionary, as shown in Figure 10. It started by creating an empty dictionary to hold the trained models. Then, looping through the dictionary and training the model using the trained data that was resampled to handle class imbalance.

```
trained = {}
for name, model in models.items():
    model.fit(X_train_res, y_train_res)
    trained[name] = model
```

Figure 10. Training XGBoost and SVM Models.

Models' performance was compared according to Figure 11. It entailed evaluating the trained XGBoost and SVM models that were stored in the dictionary. Then, prediction probabilities were made by computing the classification report through precision, accuracy, recall, F1 and ROC-AUC and stored for the positive class labeled 1.

From the output table, XGBoost had a high recall of 0.90 but low precision of 0.15, thus predicting nearly everything as positive, hence producing many false positives, while catching most actual positives.

SVM (Linear) showed high accuracy of 0.73 and low precision of 0.22, hence many false positives. It also had a lower recall of 0.52, so the model was more conservative in predicting positives. Therefore, although XGBoost had low accuracy, it had a ROC-AUC of 0.74, which was relatively higher than SVM, whose ROC-AUC was 0.70, indicating better ability to separate various classes.

```
test_results = {}
for name, model in trained.items():
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1]
    cr = classification_report(y_test, y_pred, output_dict=True)
    test_results[name] = {
        'Accuracy':  round((y_pred == y_test).mean(), 4),
        'Precision': round(cr['1']['precision'], 4),
        'Recall':    round(cr['1']['recall'], 4),
        'F1':        round(cr['1']['f1-score'], 4),
        'ROC-AUC':   round(roc_auc_score(y_test, y_prob), 4),
        'Avg Prec':  round(average_precision_score(y_test, y_prob), 4),
    }

test_df = pd.DataFrame(test_results).T.sort_values('ROC-AUC', ascending=False)
print("Test Set Results ")
display(test_df)
```

Test Set Results

	Accuracy	Precision	Recall	F1	ROC-AUC	Avg Prec
XGBoost	0.4049	0.1524	0.9042	0.2609	0.7404	0.2979
SVM (Linear)	0.7290	0.2192	0.5207	0.3085	0.6964	0.2521

Figure 11. Performance Metrics.

4.3. XGBoost and SVM Model Evaluation

4.3.1. Confusion Matrices-Test Set

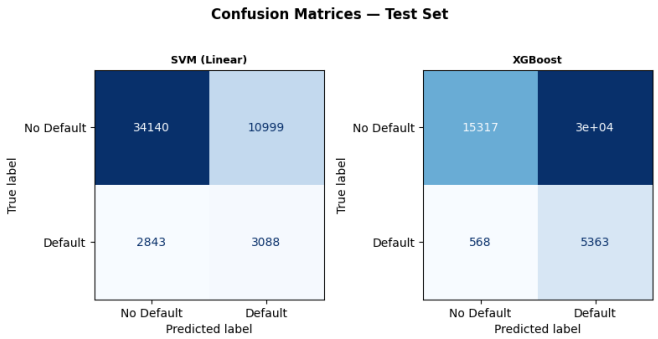

Figure 12. Confusion Matrices.

The above confusion matrices (Figure 12) provide a side-by-side comparison of XGBoost and SVM (Linear), visualizing how well they can predict the probability of a borrower defaulting or not defaulting on the loan.

The SVM(Linear) model shows 34140 as actual non-default, 2,843 as actual default from the true label, while the model predicted 10999 as non-default and 3088 as default.

Therefore, this model has a low probability of identifying loan defaulters and can predict no-default even in cases which might default, resulting in false negatives, which is dangerous during the loan approval process.

The XGBoost shows 15317 as non-default and 568 as actual default from the true label, while the model predicted 30000 cases as non-default and 5363 as default cases. Therefore, the model is more aggressive in predicting default cases in comparison to SVM and catching more true positives, hence XGBoost turns out to be a better model in reducing false negatives, where an actual default is incorrectly categorized as no default.

4.3.2. ROC-AUC Curves

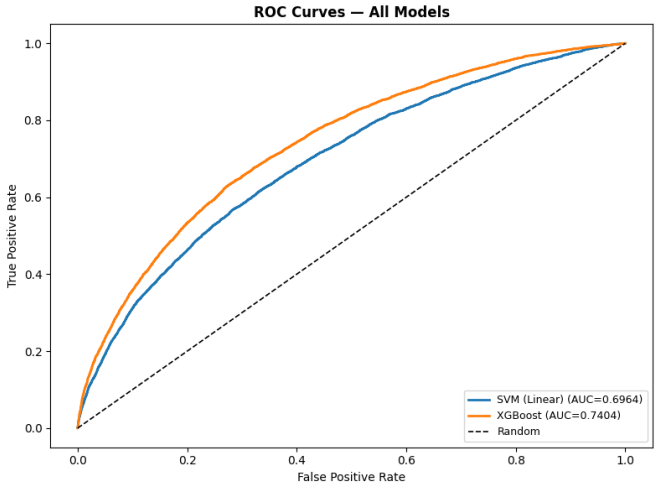

Figure 13. *Receiver Operating Characteristics (ROC) Curves.*

The Figure 13 above shows the ROC curves for SVM Linear and XGBoost models in comparison to a random classifier.

The x-axis represents the false positive rate used in measuring the proportion of actual negative values incorrectly predicted as positives. The y-axis represents the true positive rate, which measures the proportion of actual positive values correctly identified. The diagonal dashed line was a random classifier, and for any useful model, its ROC curve should be above the random classifier.

The SVM Linear model was represented with a blue line, with an AUC of 0.6964, with its curve starting above the random classifier, indicating that it is better than random. However, the AUC of 0.7 was moderate, suggesting the existence of overlap between negative and positive classes.

The XGBoost model was represented by a yellow line that sits above the SVM curve with an AUC of 0.7404, indicating better separability between different classes. The model captured the nonlinear relationship better in comparison to the SVM model.

4.3.3. Precision- Recall Curves

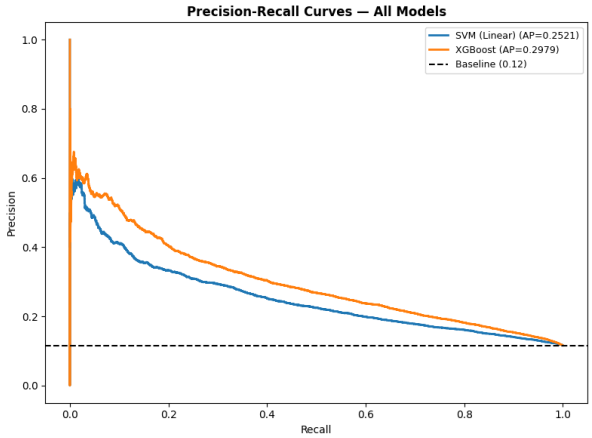

Figure 14. *Precision Recall Curves for XGBoost and SVM Models.*

The Precision-Recall Curves above (Figure 14) compare the performance of Linear SVM and XGBoost on a binary classification dataset. The XGBoost had an Average Precision of 0.2979, indicating that it performs better in comparison to SVM, and its curve is relatively higher at every recall level, achieving higher precision. The SVM model had an average precision of 0.2521, better than the random average precision. The curve starts with a low recall, with decent precision dropping quickly as recall increases.

The x-axis represents the recall, which is the true positive rate, measuring all actual positive cases, while the y-axis represents the precision, which is a positive predictive value measuring all positive cases in the model. The precision-recall curve is more informative for an imbalanced dataset, focusing on the positive class, which is the non-default class during the loan prediction.

4.3.4. Classification Model Evaluation

This report was generated from the trained model with the best performance based on the prior metrics. The Figure 15 below is the XGBoost classification report, which showed better performance in comparison to SVM.

Classification Report: XGBoost

	precision	recall	f1-score	support
No Default	0.96	0.34	0.50	45139
Default	0.15	0.90	0.26	5931
accuracy			0.40	51070
macro avg	0.56	0.62	0.38	51070
weighted avg	0.87	0.40	0.47	51070

Figure 15. *Evaluation of XGBoost.*

The precision of 0.96 indicates that the model predicts no default of 96%, therefore, causes an alarm for default when determining loan eligibility. From the recall of 0.34, the model finds only 0.34 of actual non-defaulters. An F1-score of 0.50 was low as a result of low recall.

The default precision of 0.15 indicates that the model had 0.15 correctness when predicting default rate, while the recall of 0.90 indicates that the model was very sensitive to actual defaulters. The low F1 score of 0.26 was a result of poor precision.

4.4. Hyperparameter Tuning Process

```
param_grid = {
    'SVM (Linear)': {
        'base_estimator__C': [0.01, 0.1, 1, 10, 100]
    },
    'XGBoost': {
        'n_estimators':     [100, 200, 300],
        'max_depth':        [3, 5, 7],
        'learning_rate':    [0.01, 0.05, 0.1],
        'subsample':        [0.7, 0.85, 1.0],
        'min_child_weight': [1, 3, 5],
    }
}

print(f"Tuning: {best_name}")
search = RandomizedSearchCV(
    best_model, param_grid[best_name],
    n_iter=30,
    scoring='roc_auc',
    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=SEED),
    random_state=SEED, n_jobs=-1, verbose=1, refit=True
)
search.fit(X_train_res, y_train_res)
tuned = search.best_estimator_

print(f"\nBest params: {search.best_params_}")
print(f"Best CV AUC: {search.best_score_:.4f}")
```

```
Tuning: XGBoost
Fitting 5 folds for each of 30 candidates, totalling 150 fits

Best params: {'subsample': 0.85, 'n_estimators': 300, 'min_child_weight': 3, 'max_depth': 7, 'learning_rate': 0.1}
Best CV AUC: 0.9615
```

Figure 16. *Hyperparameter Tuning Process.*

The Figure 16 above shows the hyperparameter tuning process using randomized search cross-validation. For the linear SVM model tuned only for the regularization parameters, while XGBoost model tuning was done for the number of boosting rounds, maximum tree depth to control overfitting, step size shrinkage and the fraction of sample per tree used. The tuning setup was done using a randomized search.

The maximum depth of 7, which was moderate, allowed for the learning of the models' complex patterns moderately, while the learning of 0.1 indicates faster learning, though is not conservative. The subsample of 0.85 indicates the percentage of data each tree used.

```
Tuned XGBoost - Test AUC: 0.7330
              precision    recall  f1-score   support

  No Default       0.94      0.60      0.74     45139
     Default       0.19      0.73      0.31      5931

    accuracy                           0.62     51070
   macro avg       0.57      0.67      0.52     51070
weighted avg       0.86      0.62      0.69     51070
```

Figure 17. *XGBoost Model Tuning.*

The XGBoost model was tuned (Figure 17) to predict the probability of loan default, with class 0 representing no default and class 1 representing default. The no-default class had a precision of 0.94, indicating the percentage of the correctly predicted values in the model, while the recall of 0.60 indicates the percentage of the model's catches of the actual non-default classes.

The default class had a precision of 0.19, which was very low, showing the percentage of defaults predicted is indeed actual defaults. The recall of 0.73 showed the percentage of actual defaults the model identifies. The AUC of 0.73 indicates that the model was able to rank non-default lower than default. Therefore, since the objective was to minimize the default, the recall of 0.73 shows that the model works reasonably better.

4.5. Model Performance Comparison

This involved comparing the performances of the three models: XGBoost, tuned XGBoost and Linear SVM. Their accuracy, precision, recall, F1, AUC-ROC and average precision are given in Figure 18. The XGBoost model improved the recall and F1, hence the tuning minimized the false positives. Also, all models showcase low precision, thus struggling to avoid false negatives during loan prediction. From the ROC-AUC curves, all the models possess moderate ability to separate classes.

FINAL MODEL COMPARISON

Model	Accuracy	Precision	Recall	F1	ROC-AUC	Avg Prec
XGBoost	0.4049	0.1524	0.9042	0.2609	0.7404	0.2979
XGBoost (Tuned)	0.6176	0.1940	0.7269	0.3063	0.7330	0.3010
SVM (Linear)	0.7290	0.2192	0.5207	0.3085	0.6964	0.2521

Figure 18. *Final Model Comparison.*

4.6. Hybrid Models

```
SVM out-of-fold predictions shape: (361110,)
XGBoost out-of-fold predictions shape: (361110,)
```

Figure 19. *Number of Arrays in XGBoost and SVM Models.*

The number of arrays in the models is as per Figure 19 Both had 361110 arrays, which was essential for carrying out fold prediction for both models. Every array contained positive class predicted probabilities between 0 and 1. The out of fold probabilities were crucial in blending and training the final classifier to develop the hybrid model.

```
# Build meta-feature matrix for training

meta_train = np.column_stack([svm_oof, xgb_oof])
print(f"\nMeta-training matrix shape: {meta_train.shape}")
print("Columns: [SVM_prob, XGBoost_prob]")
```

```
Meta-training matrix shape: (361110, 2)
Columns: [SVM_prob, XGBoost_prob]
```

Figure 20. *Meta Feature Construction.*

The construction of the meta-feature matrix was important for training a stacking ensemble machine learning model. It involved taking out-of-fold predictions from the trained SVM and XGBoost base models and stacking them side by side as input variables for higher-level prediction. From the output, 361110 training samples were used per row as per the original training data and two columns, one column per SVM and XGBoost base models. Therefore, the meta feature aimed to determine the more reliable feature of either of the base models and use it for the final prediction.

4.7. Train the Meta-classifier

```
# Train meta-classifier
meta_clf = LogisticRegression(random_state=SEED, max_iter=1000)
meta_clf.fit(meta_train, y_train_res)

print(f"\nMeta-classifier weights:")
print(f" SVM coefficient: {meta_clf.coef_[0][0]:.4f}")
print(f" XGBoost coefficient: {meta_clf.coef_[0][1]:.4f}")
print("Higher coefficient = that model contributes more to the final prediction.")
```

```
Meta-classifier weights:
 SVM coefficient: -1.0392
 XGBoost coefficient: 14.6468
Higher coefficient = that model contributes more to the final prediction.
```

Figure 21. *Training Meta-Classifier.*

The meta-classifier was used to learn the best way to combine the predictions of the base models to make the final decision (Figure 21). The meta-model executes the weighted combination of the outputs of the base models. The coefficients of the meta classifier for the SVM and XGBoost were -1,0392 and 14.6468, respectively. These were the learned weights before the sigmoid function application. The XGBoost model had a very large positive weight, thus making it increase the positive class, increasing the final probability, hence dominating the final decision. The negative weight of the SVM model decreased the final positive outcome, thus acting as a negative regulator. Therefore, the SVM was used to improve the workability of the XGBoost, which acted as the primary driver.

4.8. Evaluation of Hybrid Model

```
Hybrid Model (SVM + XGBoost  to LR Meta-classifier)
              precision    recall  f1-score   support

   No Default       0.93      0.83      0.87     45139
      Default       0.27      0.49      0.35      5931

     accuracy                           0.79     51070
    macro avg       0.60      0.66      0.61     51070
 weighted avg       0.85      0.79      0.81     51070

Accuracy     0.7873
Precision    0.2708
Recall       0.4913
F1           0.3492
ROC-AUC      0.7407
Avg Prec     0.2986
dtype: float64
```

Figure 22. *Hybrid Model Evaluation.*

Figure 22 gives an evaluation of a hybrid machine learning model in predicting the loan default. It had SVM and XGBoost models, which were the base models. Logistic regression was used as the meta-classifier, where the predictions from the models were fed into the meta-classifier to learn and combine them in the best way possible.

The no-default class had a high precision of 0.93, showing few positives among the no-default predicted, while the recall was 0.83, indicating the percentage of actual non-defaulters the model was able to catch. The F1-score of 0.87 showed a good balance for most of the class.

For the default class, the precision was 0.27, showing the percentage of correct predictions the model was able to make, and a recall of 0.49, revealing the percentage of actual defaulters the model was able to find. The F1-score of 0.35 indicated relatively low performance for most of the class, mostly due to low precision.

4.9. Comparing Hybrid Versus Individual Models

Model Comparison: Individual vs Hybrid

Model	Accuracy	Precision	Recall	F1	ROC-AUC	Avg Prec
Hybrid (SVM + XGBoost)	0.7873	0.2708	0.4913	0.3492	0.7407	0.2986
XGBoost	0.4049	0.1524	0.9042	0.2609	0.7404	0.2979
XGBoost (Tuned)	0.6176	0.1940	0.7269	0.3063	0.7330	0.3010
SVM (Linear)	0.7290	0.2192	0.5207	0.3085	0.6964	0.2521

Figure 23. *Models Comparison Table.*

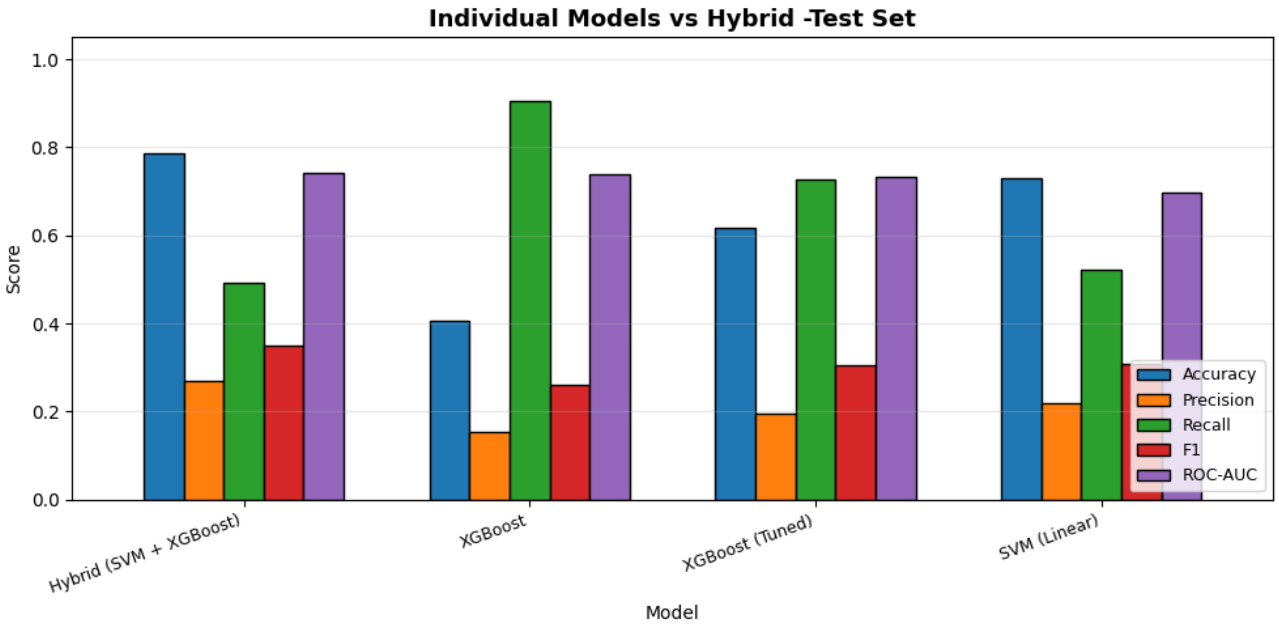

Figure 24. *Models Comparison Graph.*

The above Figure 23 and 24 gave a clear reflection of the performance of the three models. The hybrid model was overall the best with the highest accuracy of 0.78, the highest precision of 0.27, the highest ROC-AUC of 0.74, the highest average precision of 0.29, and a balanced recall of 0.49; hence, the model was able to combine the strengths of SVM and XGBoost. The precision of 0.27 was significant because it showed fewer false alarms, thus minimizing rejecting good customers in the lending business. Also, the balanced recall of 0.49 was crucial since the hybrid model was able to capture nearly half of the actual defaulters, thus minimizing the losses among the defaults and reducing the provision for bad loans. The hybrid model was able to balance F1-score and ROC-AUC curve, 0.35 and 0.74, respectively, thus maximizing the profitability while reducing credit risk for loan default. Therefore, the hybrid (XGBoost-SVM) was the most preferred model in predicting loan default since it achieved the best balances across all performance metrics, which directly impacts customer satisfaction, lenders' profitability, and thus reduces financial losses.

5. Conclusions

The accuracy in assessing loan eligibility is a strategic function and critical operation in the banking sector in determining the financial aptness. While traditional underwriting systems were interpretable, they could not capture non-linear relationships, and complex data from an increased number of loan applicants, leading to wrong credit decisions, potential customer attrition and increased non-performing credit facilities. This research was set out to address these limitations through the development and evaluation of a hybrid machine learning model that leveraged the strengths of Support Vector Machine (SVM) and Extreme Gradient Boosting (XGBoost) models.

During the data preprocessing phase, which comprised handling missing values, handling class imbalance and feature encoding, the research involved training and validating three distinct machine learning models. The comprehensive assessment of performance metrics was carried out to evaluate accuracy, precision, recall, F1-score, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC). In all these performance metrics, the hybrid model unequivocally demonstrated greater results with an accuracy of 0.78, precision of 0.27, the highest ROC-AUC of 0.74, the highest average precision of 0.29, and a balanced recall of 0.49. The superior predictive accuracy indicated that the model was accurately able to identify the proportion of both ineligible and eligible loan applicants, thus lowering misclassifications of defaults during loan eligibility determination.

Notably, balanced recall and precision of 0.49 indicated the ability of the hybrid model to successfully harmonize precision and recall metrics, leading to the highest FI-score of 0.34, giving robust performance and reducing the default rate during the loan approval process. Additionally, hybrid models achieved the greatest AUC-ROC of 0.74, enhancing their capability to distinguish between eligible and ineligible loan applicants, making it a reliable decision tool during the loan approval process. The greater performance of the SVM-XGBoost hybrid model was attributed to its ability to leverage the strengths of the standalone SVM and XGBoost models, where the model compensated for the individual models' weaknesses, leading to more resilient loan predictive results. Therefore, application of the XGBoost-SVM hybrid model would significantly reduce the loan default rate, enhance financial inclusion, as well as accelerating turnaround time for loan approval, thus improving customer satisfaction and reducing financial losses in the banking sector.

Abbreviations

ANOVA	Analysis of Variance
AUC	Area Under Curve
CatBoost	Categorical Boosting
LightGBM	Light Gradient Boosting Machine
ROC	Receiver Operating Curve
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
WOE	Weight of Evidence
XGBoost	eXtreme Gradient Boosting

Author Contributions

James Githaiga Muhoro: Conceptualization, Data curation, Formal Analysis, Investigation, Methodology, Resources, Visualization, Writing - original draft

Harun Mwangi Gitonga: Project administration, Supervision, Validation, Writing - review & editing

Josephine Njeri Ngure: Project administration, Supervision, Validation, Writing - review & editing

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Nazim Uddin et al. "An ensemble machine learning based bank loan approval predictions system with a smart application". In: International Journal of Cognitive Computing in Engineering 4 (2023), pp. 327-339. https://doi.org/10.1016/j.ijcce.2023.09.001

[2] CL Perera and SC Premaratne. "An ensemble machine learning approach for forecasting credit risk of loan applications". In: WSEAS Transactions on Systems 23 (2024), pp. 31-46. https://doi.org/10.37394/23202.2024.23.4

[3] Sinap, Vahid (2024). Vahid Sinap. "A comparative study of loan approval prediction using machine learning methods". In: Gazi Universitesi Fen Bilimleri Dergisi

Part C: Tasarim ve Teknoloji 12.2 (2024), pp. 644-663. https://doi.org/10.29109/gujsc.1455978

[4] Haque FM Ahosanul and Mahedi Hassan. "Bank Loan Prediction Using Machine Learning Techniques". In: arXiv e-prints (2024), arXiv-2410. https://doi.org/10.48550/arXiv.2410.08886

[5] Cheuk Lam Lai. "A Novel Machine Learning-based Ensemble Model for Loan Prediction". In: 2025 3rd International Conference on Image, Algorithms, and Artificial Intelligence (ICIAAI 2025). Atlantis Press. 2025, pp. 263-273. https://doi.org/10.2991/978-94-6463-823-3_25

[6] Debabrata Dansana et al. "Analyzing the impact of loan features on bank loan prediction using R andom F orest algorithm". In: Engineering Reports 6.2 (2024), e12707. https://doi.org/10.1002/eng2.12707

[7] Davinder Paul Singh et al. "Predictive Modeling for Bank Loan Approval: From Data to Decisions". In: Procedia Computer Science 259 (2025), pp. 1426-1431. https://doi.org/10.1016/j.procs.2025.04.097

[8] Eslam Hussein Sayed et al. "Machine learning and deep learning for loan prediction in banking: Exploring ensemble methods and data balancing". In: IEEE Access 12 (2024), pp. 193997-194019. https://doi.org/10.1109/ACCESS.2024.3509774

[9] Golda T Kisutsa. "Loan Default Prediction Using Machine Learning: a Case of Mobile Based Lending". Doctoral dissertation. University of Nairobi, 2021. https://erepository.uonbi.ac.ke/handle/11295/155863

[10] Zheng Zhi Kang et al. "Loan Default Prediction Using Machine Learning Algorithms". In: Journal of Informatics and Web Engineering 4.3 (2025), pp. 232-244. https://doi.org/10.33093/jiwe.2025.4.3.14

[11] Xinyu Zhang et al. "Data-driven loan default prediction: A machine learning approach for enhancing business process management". In: Systems 13.7 (2025), p. 581. https://doi.org/10.3390/systems13070581

[12] Keke Yu et al. "Loan approval prediction improved by XGBoost model based on fourvector optimization algorithm". In: Applied and Computational Engineering (2024). https://doi.org/10.20944/preprints202410.0783.v1

[13] Ruoyu Qi. "Loan default prediction and feature importance analysis based on the XGBoost model". In: European Journal of Business, Economics & Management 1.2 (2025), pp. 141-149. https://doi.org/10.71222/p9qmaa87

[14] Xu Zhu et al. "Explainable prediction of loan default based on machine learning models". In: Data Science and Management 6.3 (2023), pp. https://doi.org/10.1016/j.dsm.2023.04.003

[15] Malti Bansal, Apoorva Goyal, and Apoorva Choudhary. "A comparative analysis of K-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning". In: Decision analytics journal 3 (2022), p. 100071. https://doi.org/10.1016/j.dajour.2022.100071

[16] Vikas Kumar et al. "AI-based hybrid models for predicting loan risk in the banking sector". In: Big Data Mining and Analytics 6.4 (2023), pp. 478-490. https://doi.org/10.26599/BDMA.2022.9020037

[17] Dhritiman Saha and Annamalai Manickavasagan. "Machine learning techniques for analysis of hyperspectral images to determine quality of food products: A review". In: Current Research in Food Science 4 (2021), pp. 28-44. https://doi.org/10.1016/j.crfs.2021.01.002

[18] Obare Dm and Muraya Mm. "Comparison of accuracy of support vector machine model and logistic regression model in predicting individual loan defaults". In: American Journal of Applied Mathematics and Statistics 6.6 (2018), pp. 266-271.

[19] Mehmet Furkan Akca and Onur Sevli. "Predicting acceptance of the bank loan offers by using support vector machines". In: International Advanced Researches and Engineering Journal 6.2 (2022), pp. 142-147. https://doi.org/10.35860/iarej.1058724

[20] Novita Angraini, Kelly Rosalina, and Andini Kosasih. "Optimizing Loan Approval Processes with Support Vector Machines (SVM)". In: ITEJ (Information Technology Engineering Journals) 9.2 (2024), pp. 53-61. https://doi.org/10.24235/itej.v9i2.138

[21] Purvi Prabhakar Shetty. "A Hybrid Feature Selection and Hybrid Prediction Model for Credit Risk Prediction". Doctoral dissertation. Dublin, National College of Ireland, 2023. https://norma.ncirl.ie/id/eprint/6303

[22] Shivam Krishna and Arun Solanki. "Hybrid Machine Learning Models for Credit Risk Prediction: An Explainable AI Approach". In: Proceedings of Data Analytics and Management. Cham: Springer Nature Switzerland, 2025, pp. 483-492. https://doi.org/10.1007/978-3-032-02831-0_39

[23] Reena Singh, Vedika Bengani, and Khushi Saini. Hybrid learning systems: Integrating traditional machine learning with deep learning techniques. 2024. https://doi.org/10.13140/RG.2.2.10461.22244/1

[24] Jorge Paredes et al. "A hybrid machine learning algorithm approach to predictive maintenance tasks: a comparison with machine learning algorithms". In: Results in Engineering (2025), p. 105137. https://doi.org/10.1016/j.rineng.2025.105137