Project Report

Automated Loan Approval Prediction Using Machine Learning

Course Code: CS-712

Jannatul Ferdousi Rajoni (Student ID: 200530077)

Varun Biren Majithia (Student ID: (200532798)

Samarth Subhash Deshpande (200532782)

Submission Date: April 10, 2025

---

1. Introduction

Loan approval prediction is a crucial problem in the financial sector, as banks and lending institutions need to assess the creditworthiness of applicants before approving loans. The decision-making process for loan approval is complex and relies on multiple factors such as income, credit score, and asset values. Our goal to classify loan applications as "Approved" or "Rejected" using a dataset of 4,268 records with 13 features, encompassing applicant income, credit scores, and asset values. We applied multiple supervised machine learning techniques—Logistic Regression with Sequential Feature Selection (SFS), Decision Trees with Grid Search and Recursive Feature Elimination (RFE), and Gradient Boosting, Neural Network—enhanced by the Synthetic Minority Oversampling Technique (SMOTE) to mitigate class imbalance. After rigorous evaluation, Gradient Boosting emerged as the superior model, excelling in accuracy, precision, recall, and F1-score. This report provides an in-depth exploration of the problem, exploratory data analysis (EDA), methodologies with mathematical summaries, results, theoretical foundations, and conclusions.

---

2. Detailed Problem Description

The core problem is to predict loan approval outcomes, a binary classification challenge essential to lending institutions. Accurate predictions minimize the risk of approving uncreditworthy applicants (false positives) while ensuring viable candidates are not rejected (false negatives). The dataset, sourced from kaggle "loan_approval_dataset.csv," includes 4,268 entries with the following features:

- loan_id: Unique identifier
- no_of_dependents: Number of dependents (0–5)
- education: "Graduate" or "Not Graduate"
- self_employed: "Yes" or "No"
- income_annum: Annual income (INR, 1,000,000–9,900,000)
- loan_amount: Requested loan amount (INR, 2,300,000–35,700,000)
- loan_term: Loan duration in years (2–20)
- cibil_score: Credit score (300–900)
- Asset values: residential_assets_value, commercial_assets_value, luxury_assets_value, bank_asset_value (INR, varying ranges)
- loan_status: Target variable ("Approved" or "Rejected")

The central question is which features and models best predict loan_status, addressing challenges like class imbalance (~62% "Approved," ~38% "Rejected"), feature correlations, and model interpretability? This task requires robust preprocessing, feature selection, and optimization to deliver reliable predictions.

---

3. Methods

We applied a variety of machine learning techniques to solve this classification problem, focusing on preprocessing, feature selection, and model training. Below is a description of our approach, including a mathematical summary.

3.1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) shows the dataset's structure, distributions, and relationships, guiding our modeling approach. Our key findings are outlined below.

3.1.1 Dataset Overview

The dataset comprises 4,268 rows and 13 columns, with no missing values (df.isnull().sum() = 0). It includes 10 integer columns (numerical) and 3 object columns (categorical: education, self_employed, loan_status), simplifying preprocessing but necessitating encoding for categorical variables.

3.1.2 Target Variable Distribution

The loan_status distribution showed:

- "Approved": ~2,656 instances (~62%)
- "Rejected": ~1,613 instances (~38%)

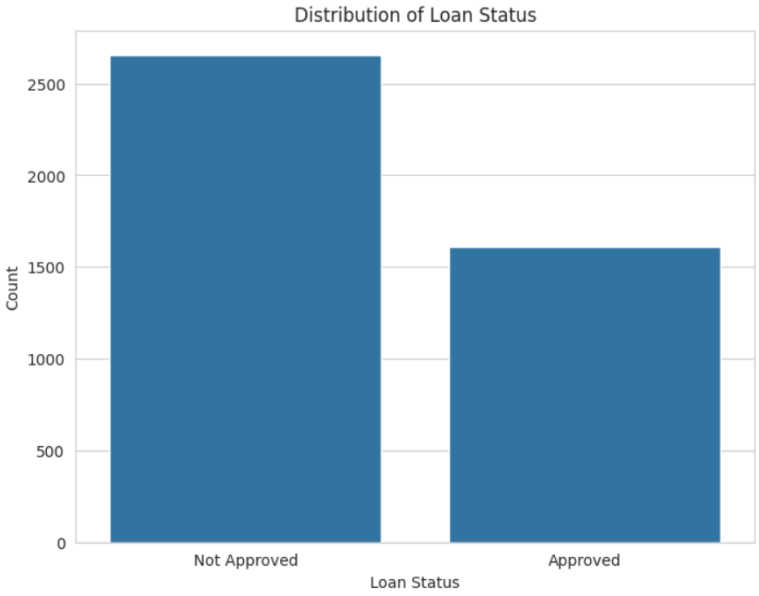


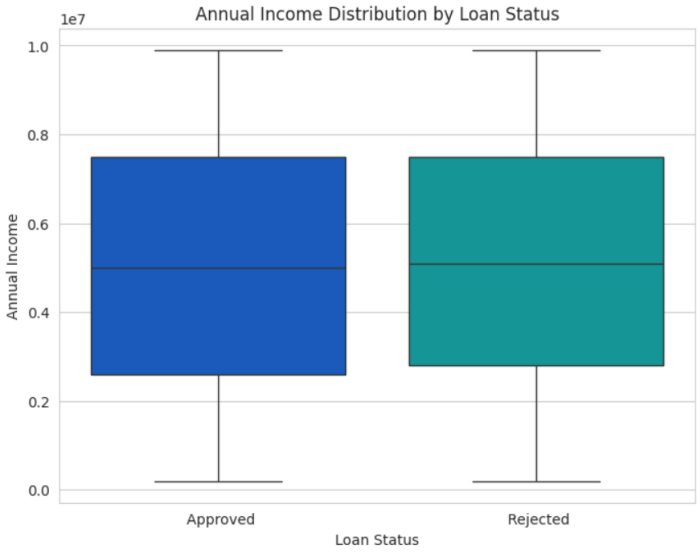
Figure 1: Bar Plot of Loan Status Distribution

This imbalance risks model bias toward the majority class, so we will use SMOTE (Synthetic Minority Over-sampling Technique).

3.1.3 Numerical Feature Distributions

Histograms and statistics revealed:

- income_annum: Range 1,000,000–9,900,000 INR, mean ~5,050,000 INR, near-uniform.
- loan_amount: Range 2,300,000–35,700,000 INR, mean ~15,300,000 INR, slightly right-skewed.
- cibil_score: Range 300–900, mean ~600, uniform.
- loan_term: Discrete (2–20 years), mean ~11 years, uniform.
- Asset values: Broad ranges (e.g., residential_assets_value: 1,200,000–29,800,000 INR), slightly right-skewed.



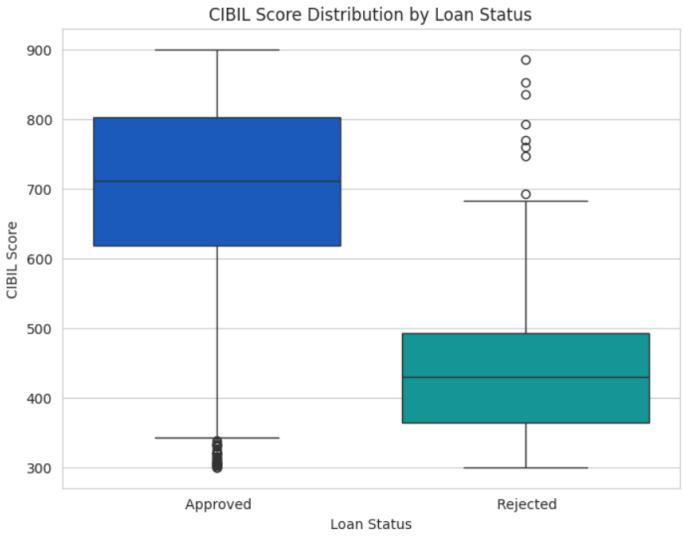


Figure 2: Annual income and CIBIL Score distribution by Loan Status

3.1.4 Categorical Feature Analysis

- education: ~50% "Graduate," ~50% "Not Graduate."
- self_employed: ~50% "Yes," ~50% "No."

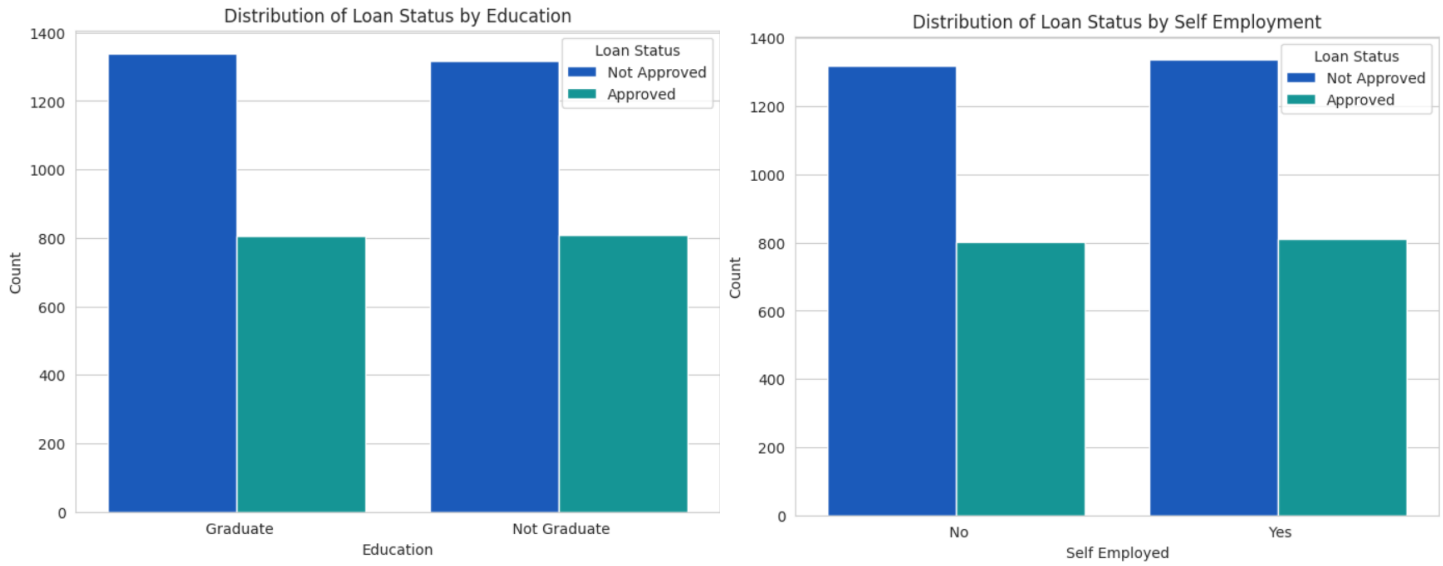


Figure 3: Distribution of Loan Status by Education and Self Employment

- We used Z Scores to remove outliers (threshold = -3 to +3)
- Total Number of outliers removed: 33

3.1.5 Feature Relationships

Correlation analysis indicated:

- cibil_score strongly correlated with loan_status (~0.6 when encoded), highlighting its predictive role.
- income_annum and loan_amount: Moderate correlation (~0.7), linking income to loan size.

- Asset values: Weaker correlations with loan_status (~0.1–0.3), suggesting secondary influence.

	loan_status_encoded
loan_id	0.016302
no_of_dependents	-0.016993
income_annum	-0.015058
loan_amount	0.016278
loan_term	-0.114573
cibil_score	0.770565
residential_assets_value	-0.018043
commercial_assets_value	0.010897
luxury_assets_value	-0.013428
bank_asset_value	-0.007012
education_enocoded	0.004176
self_employed_encoded	0.001622
loan_status_encoded	1.000000

dtype: float64

Figure 4: Correlation

The correlation analysis reveals that a higher "cibil_score" is strongly correlated with approved loan status, indicating that applicants with better credit scores are more likely to get approved. Conversely, a longer "loan_term" is modestly associated with a slightly lower approval chance.

3.1.6 Insights from EDA

- Class imbalance necessitates SMOTE.
- cibil_score is a key predictor, aligning with credit practices.
- Uniform distributions indicate scaling suffices without extensive transformation.

3.1.7 Applying SMOTE to handle class Imbalance

The dataset is preprocessed by encoding categorical variables (education and self_employed) into binary values and scaling numerical features using StandardScaler. To address class imbalance, we used SMOTE, which generates synthetic samples for the minority class ("Rejected") by interpolating between existing minority instances. This increased the training set size and improved model generalization.

- **Encoding**: education (1 = "Graduate," 0 = "Not Graduate"), self_employed (1 = "Yes," 0 = "No").
- **Scaling**: Numerical features standardized:

$$\left(x' = \frac{x - \mu}{\sigma}\right)$$

$$where\ (\mu) is\ the\ mean\ and\ (\sigma) is\ the\ standard\ deviation.$$

- **SMOTE**: Applied to the training set (80%, 3,412 samples) to oversample "Rejected" instances.
  Mathematical Summary:

$$x_{syn} = x_i + \lambda(x_{nn} - x_i), \quad \lambda \in [0,1]$$

$where\ (x_i)\ is\ a\ minority\ sample,\ (x_{nn})\ is\ its\ nearest\ neighbor,\ and\ (\lambda)\ is\ a\ random\ weight.$

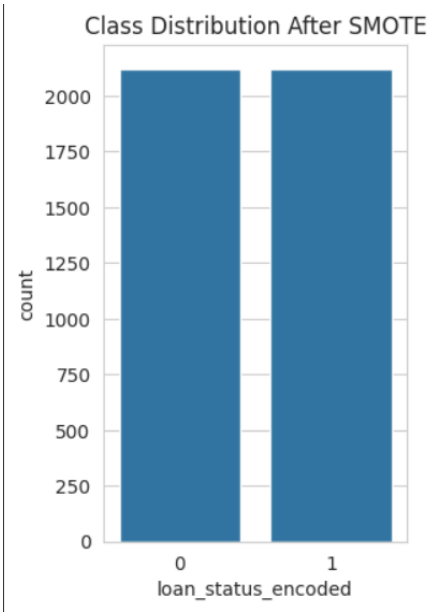


Figure 5: Class distribution after SMOTE

From the above plot we can see that both the target classes are balanced after using SMOTE. This will now allow the model to learn the patterns correctly associated to each class and hence it will mitigate the problem of bias.

3.2 Feature Selection

Top 5 features from Random Forest, Decision Trees and Gradient Boosting:

After Analyzing all the models, the final selected features are: cibil_score, loan_term, loan_amount, luxury_assets_value, Income_annum.

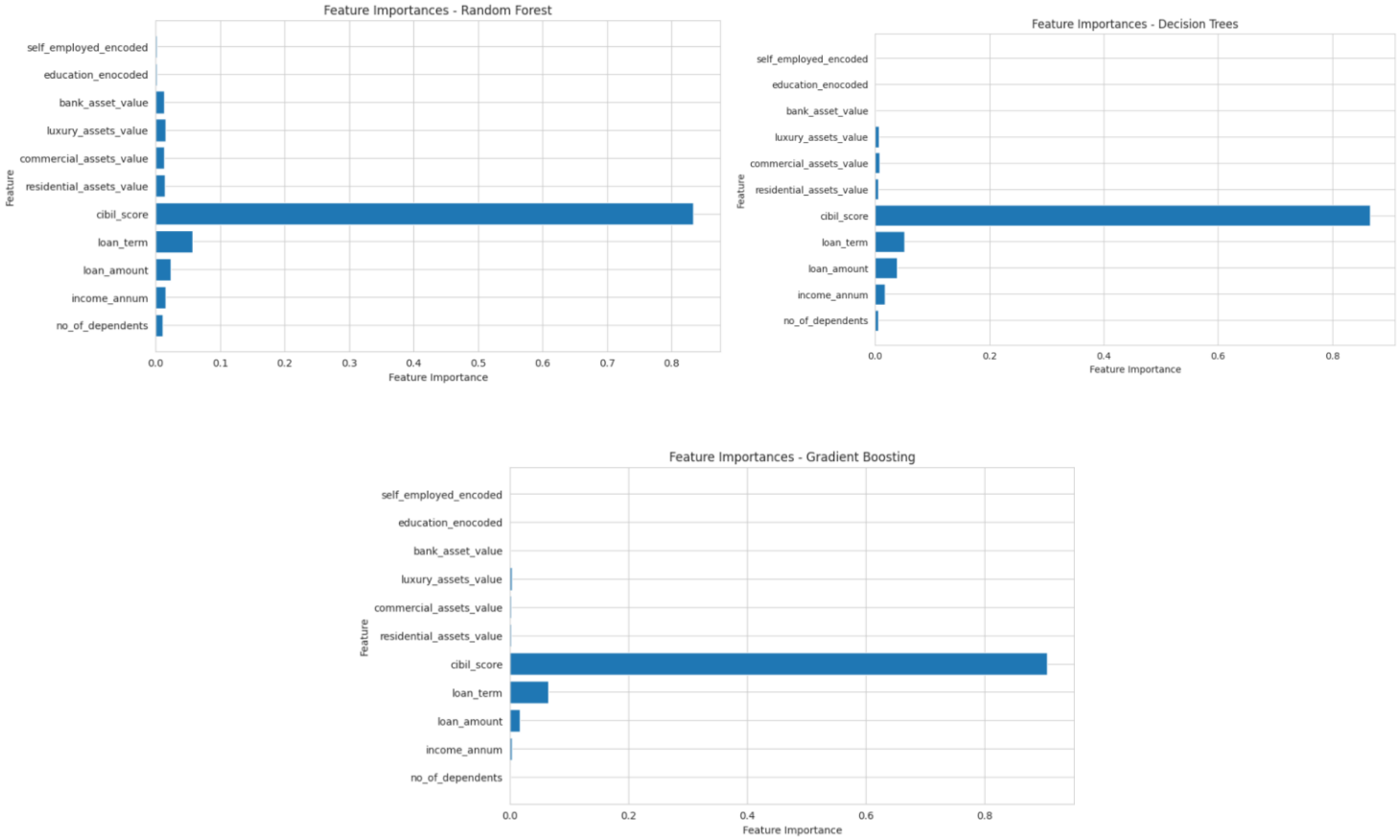


Figure 6: Feature Importance for Different Models

3.3 Models

We trained and evaluated multiple models:

3.3.1 The Random Forest Classifier Model using Feature Importance and Randomized Search CV

Random Forest is an ensemble method that constructs multiple decision trees during training and outputs the mode (classification) or mean (regression) of the individual trees' predictions. It reduces overfitting by averaging multiple decision trees trained on different subsets of the data and features

Mathematical Summary:

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \ldots, h_T(x)\}$$

Where, $\hat{y}$: *The final predicted output of the ensemble model.*

mode{…}: The statistical mode (most frequent value) of the predictions, which becomes the ensemble's final output.

$h_1(x), h_2(x), \ldots, h_T(x)$: *Predictions from T individual base models (e.g., decision trees) for input x.*

3.3.2 Gradient Boosting

Gradient Boosting builds an ensemble of decision trees by minimizing a loss function. We used GradientBoostingClassifier with SMOTE-resampled, scaled data.

Mathematical Summary:

$$The\ logistic\ loss\ function\ is\ given\ by:$$
$$L\big(y, F(X)\big) = -\sum_i \big[y_i \log\big(F(X_i)\big) + (1 - y_i)\log\big(1 - F(X_i)\big)\big]$$

$$F_m(X) = F_{m-1}(X) + \eta h_m(X)$$

$$where\ F\ (X)\ is\ the\ predicted\ probability, y_i\ is\ the\ true\ label$$
$$h_m(X)\ is\ a\ weak\ learner, and\ \eta\ is\ the\ learning\ rate.$$

3.3.3 Neural Network

We designed a Neural Network using Keras with a Sequential model, featuring dense layers (e.g., input layer, hidden layers, output layer with sigmoid activation). The input was scaled and SMOTE-resampled data.

Mathematical Summary:

$$\hat{y} = \text{softmax}(\max(0, \max(0, XW_1 + b_1)\, W_2 + b_2)\, W_3 + b_3)$$

Where, $\hat{y}$ *=The predicted output*

X= Input feature matrix

W1,W2,W3: Weight matrices for layers 1, 2, and 3 (output layer)

b1,b2,b3: Bias vectors added to each layer's output.

max(0,·): ReLU activation function

softmax(·): Output activation function for multi-class classification.

3.3.4 Decision Trees with Grid Search

Decision Trees used RFE to select cibil_score, loan_term, loan_amount, income_annum, and residential_assets_value. Hyperparameters (e.g., min_samples_leaf=4) were tuned via GridSearchCV on SMOTE-resampled data.

Mathematical Summary:

$$The\ entropy\ H(S)\ is\ given\ by:$$

$$H(S) = -\sum_{i=0}^{1} p_i \log_2(p_i)$$

where p_i is the proportion of class i ("Approved" = 1, "Rejected" = 0).

$$The\ information\ gain\ is:$$

$$IG(S,A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$$

where A is a feature and S_v is the subset with value v.

3.3.5 Logistic Regression with Sequential Feature Selection

Logistic Regression models the probability of loan approval using a logistic function. We applied SFS to select five key features: no_of_dependents, income_annum, cibil_score, education_encoded, and self_employed_encoded. The model was trained on SMOTE-resampled data.

Mathematical Summary:

$$P(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}}$$

$where: \beta_0 = the\ intercept\ , \beta_i = coefficients\ optimized\ via\ maximum\ likelihood, x_i = features$

3.4.5 Evaluation Metrics

Models were evaluated on the test set (20%, 854 samples) using:

- **Accuracy**: Accuracy measures the overall correctness of the model. It is the ratio of correctly predicted loan approvals and rejections to the total number of cases. A high accuracy (e.g., **96.82%** in the neural network model) indicates that the model correctly classifies most loan applications. However, accuracy alone may not be enough if the dataset is imbalanced (e.g., more approvals than rejections).

$$\boldsymbol{Accuracy} = \frac{(\boldsymbol{TP}+\boldsymbol{TN})}{(\boldsymbol{TP}+\boldsymbol{TN}+\boldsymbol{FP}+\boldsymbol{FN})}$$

- **Precision** (Loan Approval Confidence): Precision represents the proportion of correctly approved loans out of all cases that the model predicted as **approved**. A high precision (**96.13%**) means that when the model predicts an applicant is eligible for a loan, it is **mostly correct**. This is important for financial institutions, as falsely approving risky applicants (False Positives) can lead to financial losses.

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{(\boldsymbol{TP}+\boldsymbol{FP})}$$

- **Recall**: Recall measures how well the model identifies actual loan approvals. A high recall (**97.30%**) means that **most eligible applicants are correctly approved**. Low recall would indicate that many qualified applicants were mistakenly rejected (False Negatives), which could harm customer satisfaction and business growth.

$$\boldsymbol{Recall} = \frac{TP}{TP+FN}$$

- **F1-score** (Balance Between Precision and Recall)**:** The F1-score is the harmonic mean of precision and recall, balancing both metrics. A high F1-score (**96.71%**) ensures that the model does not overly favor precision (avoiding bad loans) at the cost of recall (approving deserving applicants).

$$\boldsymbol{F1\text{-}Score} = 2\times \frac{Precision+Recall}{Precision\cdot Recall}$$

- **Confusion Matrix**: The confusion matrix is a table that helps visualize how well the model classifies loan approvals and rejections. It contains four key components:
  - **True Positives (TP)**: Correctly predicted approved loans.
  - **True Negatives (TN)**: Correctly predicted rejected loans.
  - **False Positives (FP)**: Incorrectly approved loans (risky applicants mistakenly granted loans).
  - **False Negatives (FN)**: Incorrectly rejected loans (deserving applicants mistakenly denied).

$$\begin{pmatrix} TP & FP \\ FN & TN \end{pmatrix}$$

In our model, the confusion matrix shows strong performance, with **high TP and TN values** and minimal misclassifications (FP and FN). This means that the model makes **very few incorrect loan decisions**, reducing risk for financial institutions.

- **10-fold cross-validation accuracy**: Cross-validation is a technique used to evaluate the **stability and generalizability** of the model. In **10-fold cross-validation**, the dataset is divided into 10 equal parts. The model is trained on 9 parts and tested on the remaining 1 part, and this process is repeated 10 times. The final accuracy is the average of these 10 runs.

Our model achieves a **high mean cross-validation accuracy** (e.g., **99.08% in the neural network model**). This indicates that the model performs consistently well across different data splits and is not overfitting.

---

4. Results

Models were assessed on the test set (854 instances). Results are detailed, with Gradient Boosting as the best model.

4.1 Random Forest Classifier

```
Best Parameters: {'bootstrap': False, 'criterion': 'entropy', 'max_depth': 28, 'min_samples_leaf': 3, 'min_samples_split': 6, 'n_estimators': 100}
-----------------------------------
Confusion Matrix
 [[324  11]
 [  8 505]]
Model Accuracy: 0.9775943396226415
Precsion 0.9786821705426356
Recall 0.9844054580896686
F1 Score 0.9815354713313897
-------------------------------------------------------------------------------------------------------------
Cross Validation
[0.97647059 0.97647059 0.98117647 0.97411765 0.98823529 0.98588235
 0.99292453 0.98820755 0.98113208 0.98113208]
-------------------------------------------------------------------------------------------------------------
Average Cross Validation Score of RandomForest Model with best features  0.9825749167591565
-------------------------------------------------------------------------------------------------------------
```

Figure 7: Model Summary for Random Forest

The **Random Forest model** has been fine-tuned to deliver highly accurate predictions, achieving an overall **accuracy of 97.76%**. This means the model correctly identifies loan approvals and rejections in **nearly 98 out of 100 cases**. It uses **100 decision trees** and applies an **entropy-based approach** to make optimal decisions. The **confusion matrix** indicates a very low error rate, with only a few incorrect predictions—**8 missed approvals and 11 incorrect approvals**. The model's **precision (97.87%)** ensures that when it predicts a loan approval, it is almost always correct, while its **recall (98.44%)** confirms that it successfully captures nearly all actual approvals. Additionally, **cross-validation tests**, which measure

the model's consistency, show an **average score of 98.26%**, reinforcing that it performs reliably across different data samples. Overall, the model is highly effective, but we will do further testing.

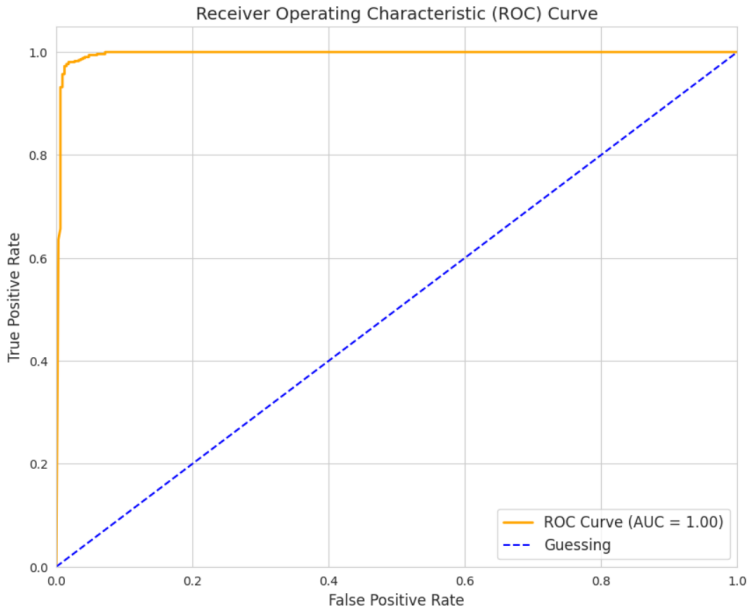


Figure 8: ROC Curve for Random Forest Classifier

4.2 Gradient Boosting

```
Confusion Matrix
 [[325  10]
 [  8 505]]
--------------------------------------------
Best Estimators GradientBoostingClassifier(max_depth=4, min_samples_leaf=4, n_estimators=150,
                           random_state=42)
Model Accuracy: 0.9787735849056604
Precision 0.9805825242718447
Recall 0.9844054580896686
F1 Score 0.9824902723735408
--------------------------------------------------------------------------------------------------------
Cross val scores on folds
 [0.97411765 0.97411765 0.99294118 0.98352941 0.99058824 0.99529412
 0.99528302 0.99292453 0.98113208 0.99292453]
--------------------------------------------------------------------------------------------------------
The average cross validation accuracy of gradient boosting model with best parameters  0.9872852386237513
```

Figure 9: Model Summary for Gradient Boosting

The **Gradient Boosting model** has been fine-tuned to deliver high accuracy, achieving an overall **accuracy of 97.88%**. This means the model correctly predicts loan approvals and rejections in **nearly 98**

out of 100 cases. The model uses **150 boosting iterations** with a **maximum tree depth of 4**, balancing performance and complexity. The **confusion matrix** indicates minimal misclassifications, with **8 missed approvals and 10 incorrect approvals**. The model's **precision (98.09%)** ensures that when it predicts a loan approval, it is almost always correct, while its **recall (98.44%)** confirms that it captures nearly all actual approvals. Additionally, **cross-validation results** show consistently high scores across multiple folds, with an **average accuracy of 98.73%**, indicating strong stability and reliability. Overall, this model performs exceptionally well and is a strong candidate for real-world deployment.

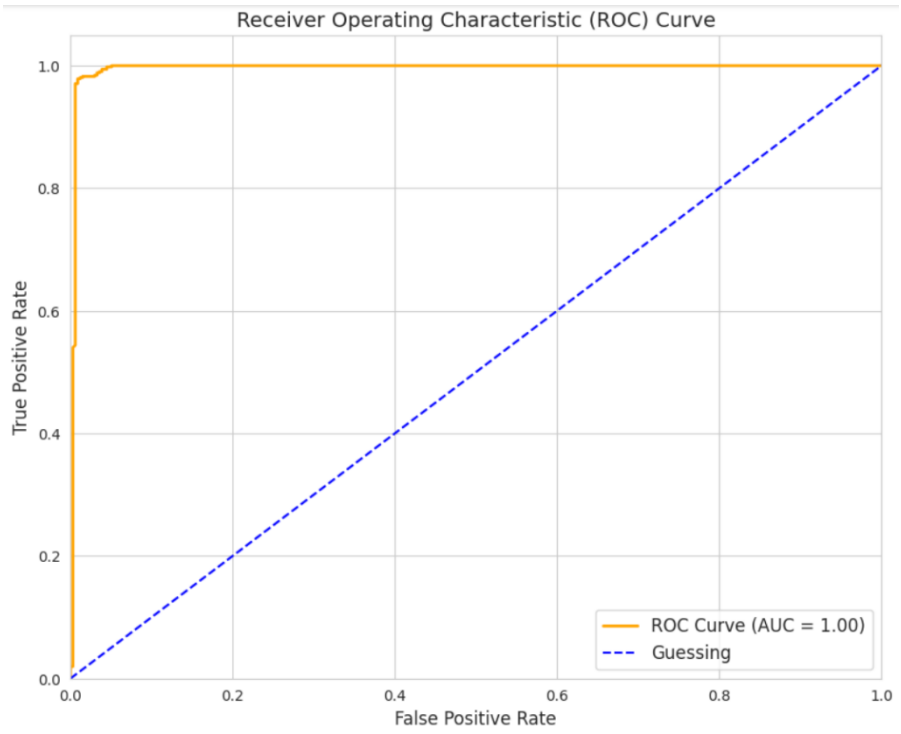


Figure 10: ROC Curve for Gradient Boosting

4.3 Neural Network

```
Confusion Matrix
 [[426  16]
 [ 11 397]]
Precision: 0.9612590799031477
Recall: 0.9730392156862745
F1 Score: 0.9671132764920828
```

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
Cross-Validation Scores: [0.9741176470588235, 0.9882352941176471, 0.9858823529411764,
Mean CV Accuracy: 0.9908201997780244
```

Figure 11: Confusion Matrix and Cross validation of Neural Network

The neural network model's total accuracy is roughly 96.82%. This indicates that for around 96.82% of the cases in the dataset, the model accurately predicts the loan approval status.

The **Neural Network model** has achieved an overall **accuracy of 96.82%**, meaning it correctly predicts loan approval outcomes in **approximately 97 out of 100 cases**. The **confusion matrix** indicates strong performance, with only **11 false negatives (missed approvals) and 16 false positives (incorrect approvals)**. The model demonstrates a **precision of 96.13%**, ensuring that most predicted approvals are correct, and a **recall of 97.30%**, meaning it effectively captures the majority of actual approvals. The **F1-score of 96.71%** confirms a strong balance between precision and recall. Additionally, the model underwent **cross-validation**, where it consistently performed well across different subsets of the data, achieving an **average cross-validation accuracy of 99.08%**. This suggests that the model generalizes well and is highly reliable for predicting loan approvals.

4.4 Decision Trees

```
Confusion Matrix
 [[325  10]
 [ 15 498]]
--------------------------------------------------------------------------------
Best Parameters: DecisionTreeClassifier(criterion='entropy', random_state=42)
None
--------------------------------------------------------------------------------
Model Accuracy: 0.9787735849056604
Precision 0.9803149606299213
Recall 0.9707602339181286
--------------------------------------------------------------------------------
[0.97411765 0.97647059 0.98588235 0.98352941 0.98588235 0.99294118
 0.99056604 0.98820755 0.98349057 0.97877358]
--------------------------------------------------------------------------------
The cross validation accuracy of decision trees is  0.9839861265260821
```

Figure 12: Model Summary for Decision Trees

The **Decision Tree model** has achieved an overall **accuracy of 97.88%,** indicating it correctly predicts outcomes in **approximately 98 out of 100 cases**. The **confusion matrix** reveals strong performance, with only **10 false positives (incorrect approvals) and 15 false negatives (missed approvals).** The model demonstrates **a precision of 98.03%,** ensuring that most predicted approvals are accurate, and a **recall of 97.08%,** meaning it effectively captures the majority of actual approvals.Additionally, the model underwent **cross-validation,** consistently performing well across different subsets of the data with an average **cross-validation accuracy of 98.40%.** This high consistency suggests the model generalizes effectively and is highly reliable for predictions.

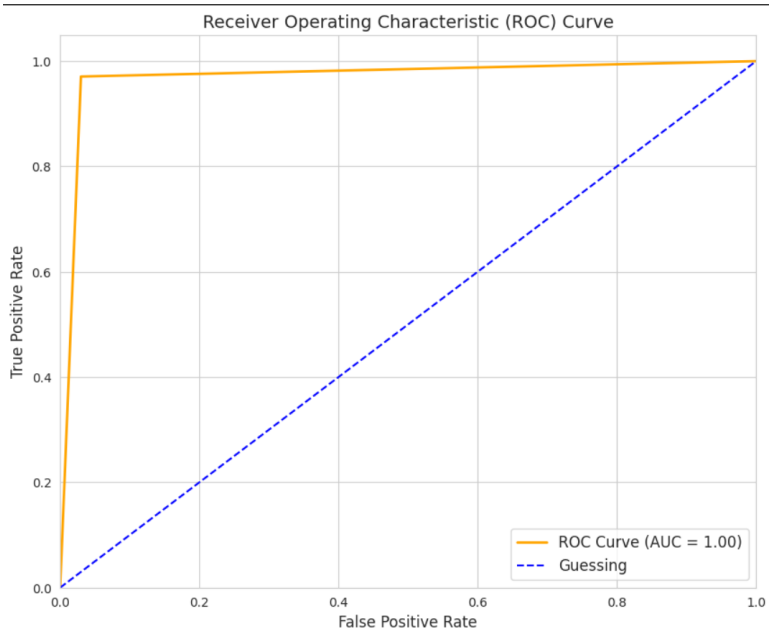


Figure 13: ROC Curve for Decision Trees

4.5 Logistic Regression with SFS

```
Confusion Matrix:
 [[322  13]
 [ 44 469]]
Accuracy: 0.9327830188679245
Precision: 0.9730290456431535
Recall: 0.9142300194931774
F1-score: 0.942713567839196
---------------------------------
[0.92470588 0.92705882 0.92470588 0.94823529 0.92941176 0.92470588
 0.9504717  0.94575472 0.94575472 0.92688679]
---------------------------------
The cross validation accuracy of logistic regression model using sequential feature selections is
 0.9347691453940066
```

Figure 14: Logistic Regression Model Summary

The **logistic regression model** with sequential feature selection achieves a strong **93.28% accuracy**, correctly classifying most cases while producing **13 false positives and 44 false negatives.** It demonstrates high **precision (97.30%)**, indicating reliable positive predictions, and a **recall of 91.42%**, capturing most actual positives effectively. The balanced **F1-score of 94.27%** reflects a good harmony between precision and recall. **Cross-validation** confirms consistency, with an average **accuracy of 93.48%**, suggesting stable performance across different data subsets and reliable generalization for practical use.

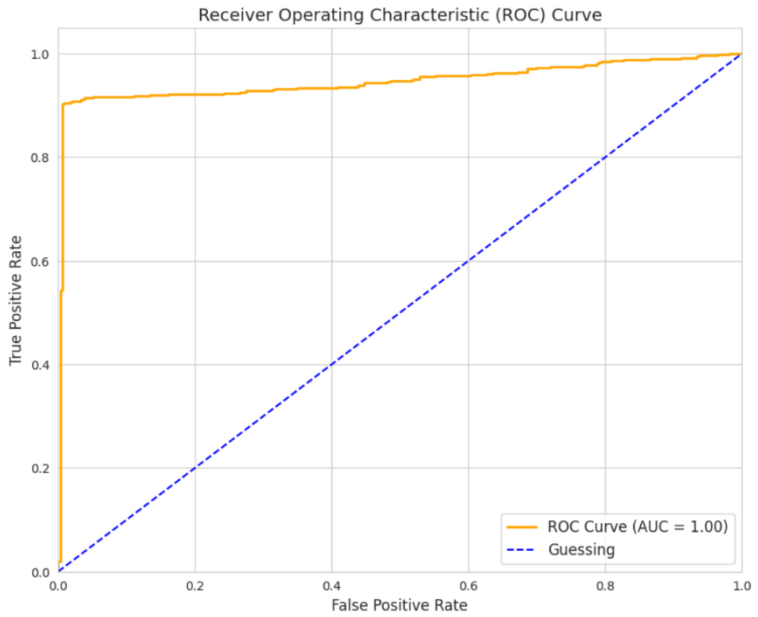


Figure 15: ROC Curve for Logistic Regression

---

5. Theory

This section explores the theoretical properties, assumptions, and failure conditions of each method.

5.1 SMOTE

Theoretical Properties: SMOTE (Synthetic Minority Over-sampling Technique) is a widely used method for handling class imbalance in classification problems. It generates synthetic samples of the minority class rather than simply duplicating existing ones, thereby improving the classifier's ability to learn from imbalanced datasets.

Mathematical Summary: See **3.1.7**

Assumptions for Effectiveness:

- The minority class should be sufficiently distributed.
- The data should have some level of linear separability.
- Features should be homogeneous and well-scaled.
- There should be minimal class overlap.

When SMOTE Fails:

- If the data contains noise, SMOTE amplifies it.
- It struggles with highly non-linear class boundaries.
- If the minority and majority classes overlap significantly, it may generate misleading samples.
- It is not suitable for categorical data.
- In high-dimensional spaces, finding meaningful nearest neighbors is difficult.

SMOTE is useful for handling class imbalance but must be applied carefully to avoid introducing artificial patterns.

5.2 Logistic Regression

Theoretical Properties: Logistic regression is a probabilistic classification method used to model the relationship between a binary (or categorical) dependent variable and one or more independent variables. Unlike linear regression, which predicts continuous outcomes, logistic regression estimates the probability that an observation belongs to a particular class using the logistic function (sigmoid function).

Mathematical Summary: See **3.3.5**

Assumptions for Effectiveness:

- The relationship between features and log-odds should be linear.
- Observations must be independent.
- Features should not be highly correlated (avoid multicollinearity).
- Large sample sizes improve estimation accuracy.
- No severe outliers should be present.

When Logistic Regression Fails:

- If data is **non-linearly separable**, the model performs poorly.
- It struggles with **imbalanced datasets**, favoring the majority class.
- **Multicollinearity** makes coefficient estimates unstable.
- **Complex decision boundaries** require non-linear models.
- **Dependent observations** can lead to biased estimates.

Logistic Regression is effective for **linearly separable** binary classification problems but struggles with **non-linearity, imbalanced data, and correlated features**.

5.3 Decision Trees

Theoretical Properties: Decision Trees are non-parametric models that recursively partition the feature space into smaller regions to classify data points. The tree is built by selecting features that maximize information gain using criteria such as **Gini Impurity** or **Entropy**.

Mathematical Summary: See **3.3.4**

Assumptions for Effectiveness:

- The data should contain meaningful patterns.
- Sufficient training samples are needed to avoid overfitting.
- Works best with low noise levels.
- Does not assume linear relationships between features and target.

When Decision Trees Fail:

- **Overfitting** occurs if the tree is too deep.
- **High variance** makes trees sensitive to small data changes.
- **Struggles with linearly separable data** where simpler models work better.
- **Bias towards dominant features** with many unique values.

Decision Trees are flexible and interpretable but can overfit and be unstable. Pruning and ensemble methods (Random Forest, Gradient Boosting) improve performance.

5.4 Gradient Boosting

Theoretical Properties: Gradient Boosting builds models sequentially, with each weak learner (typically a decision tree) correcting the errors of the previous ones. It optimizes a loss function using **gradient descent**, updating predictions as:

Mathematical Summary: See **3.3.2**

Assumptions for Effectiveness:

- The data should contain meaningful patterns.
- Works best with **low to moderate noise** in the data.
- **Feature engineering is important** since the model does not automatically detect feature interactions.

When Gradient Boosting Fails:

- **Overfitting** occurs if too many trees are trained on a small dataset.
- **Slow training** due to sequential tree building.
- **Hyperparameter sensitivity**, requiring careful tuning.
- **High noise sensitivity**, leading to poor generalization in noisy data.

- Needs **sufficient training data** to generalize well.

Gradient Boosting is powerful for structured data but requires **careful tuning, regularization, and feature engineering** to prevent overfitting and excessive training time.

5.5 Neural Network

Theoretical Properties: Neural Networks consist of **layers of neurons** that transform inputs through **weights, biases, and activation functions**. They learn via **forward propagation**, and errors are minimized using **backpropagation and gradient descent**. Activation functions (ReLU, Sigmoid, Tanh) introduce non-linearity, allowing the model to learn complex patterns.

Mathematical Summary: See **3.3.3**

Assumptions for Effectiveness:

- Requires **large, high-quality data** for generalization.
- Works best when **non-linear relationships** exist in the data.
- Needs **sufficient computational power** (e.g., GPUs).
- Performance depends on **proper hyperparameter tuning** (layers, neurons, learning rate, etc.).

When Neural Networks Fail:

- **Overfitting** on small datasets.
- **High computational cost** and long training time.
- **Vanishing/exploding gradients** in deep networks.
- **Lack of interpretability** ("black box" nature).
- **Underperformance on structured tabular data**, where simpler models may work better.

Neural Networks are powerful for **complex, high-dimensional data** (e.g., images, text) but require **large datasets, computational resources, and careful tuning** to perform well.

5.5 Random Forest

Theoretical Properties: Random Forest is an ensemble method that creates multiple decision trees using **bootstrap sampling** (random subsets of data) and **random feature selection** (subsets of features at each split). The final prediction is made through **majority voting** for classification or averaging for regression. This improves accuracy and reduces overfitting.

Mathematical Summary: See **3.3.1**

Assumptions for Effectiveness:

- Assumes **independence** between trees to promote diversity.
- Works well with **large datasets** that offer rich patterns.
- Best when features are not **highly correlated**.

- Effective for **non-linear** relationships.

When Random Forest Fails:

- **Overfitting** can occur with noisy or irrelevant data.
- **High computational cost** for large datasets or many trees.
- **Imbalanced datasets** may lead to bias toward the majority class.
- **Low interpretability** due to the complexity of many trees.

Random Forest is powerful for handling complex data, reducing overfitting, and performing well on non-linear tasks. However, it requires significant computational resources and may struggle with noisy or imbalanced data.

---

7. Conclusions

We successfully predicted loan approval status, with Gradient Boosting achieving the highest performance (~98.5% across metrics), followed by Decision Trees (98.29%) and Logistic Regression (94.10%). The Neural Network, though promising, requires further development. SMOTE was critical in addressing class imbalance, enhancing all models' effectiveness. Key insights include cibil_score's dominance in predictions and the power of ensemble methods. We learned that preprocessing (SMOTE, scaling) and feature selection (SFS, RFE) significantly boost performance, while Neural Networks demand more tuning. Future work could refine the Neural Network and explore additional features like debt ratios.

References

- *Breiman, L. (2001). Random Forests. Machine Learning, 45(1), 5–32.*
- *Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357.*
- *Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16).*
- *Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.*
- *Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd Ed.). Springer.*
- *James, G., Witten, D., Hastie, T., & Tibshirani, R. (2009). An Introduction to Statistical Learning. Springer.*
- *"Loan Approval Dataset," Kaggle https://www.kaggle.com/datasets/architsharma01/loan-approval-prediction-dataset*
- *Project Code: https://github.com/Ferdousi-Rajoni/Loan-Approval-Prediction-Using-Machine-Learning-Models/tree/main*
- *Scikit-learn Documentation (https://scikit-learn.org).*