Comparison of Random Forest Classifier (Scratch vs Scikit-learn)

1. Introduction

Random Forest is an ensemble learning algorithm that builds multiple decision trees and combines their predictions to improve accuracy and reduce overfitting. The goal of this study was to implement a Random Forest Classifier from scratch and compare its performance against the scikit-learn implementation, using a loan approval dataset.

2. Dataset

The dataset used contains loan application records with the following features:
- no_of_dependents
- education
- self_employed
- income_annum
- loan_amount
- loan_term
- cibil_score
- residential_assets_value
- commercial_assets_value
- luxury_assets_value
- bank_asset_value

The target variable is loan_status (loan approved = 1, not approved = 0).

Preprocessing steps included:
- Encoding categorical variables (education, self_employed, loan_status).
- Splitting dataset into 80% training and 20% testing.

3. Random Forest (Scratch Implementation)

The Random Forest was implemented manually by combining multiple decision trees built from scratch. The main steps were:
- Bootstrapping: randomly sampling subsets of the dataset.
- Random feature selection: choosing random subsets of features at each split.
- Decision tree building: using entropy/variance as splitting criteria.
- Aggregation: majority voting for classification.

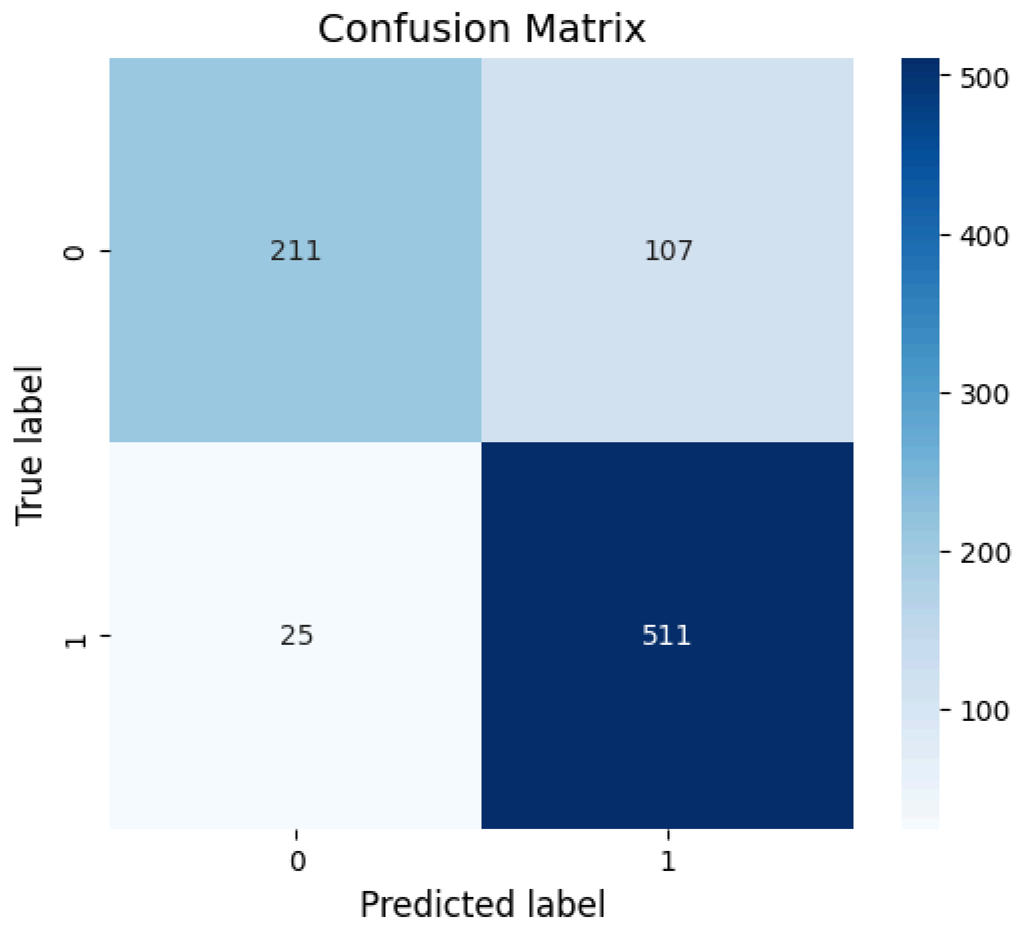
Limitations:
- Slower training due to Python loops.
- Limited optimization (no parallelization, no pruning).
- Requires manual handling of missing values and categorical encoding.

4. Random Forest (Scikit-learn Implementation)

Scikit-learn provides an optimized RandomForestClassifier. For this experiment, the following parameters were used:
- n_estimators=100
- random_state=42

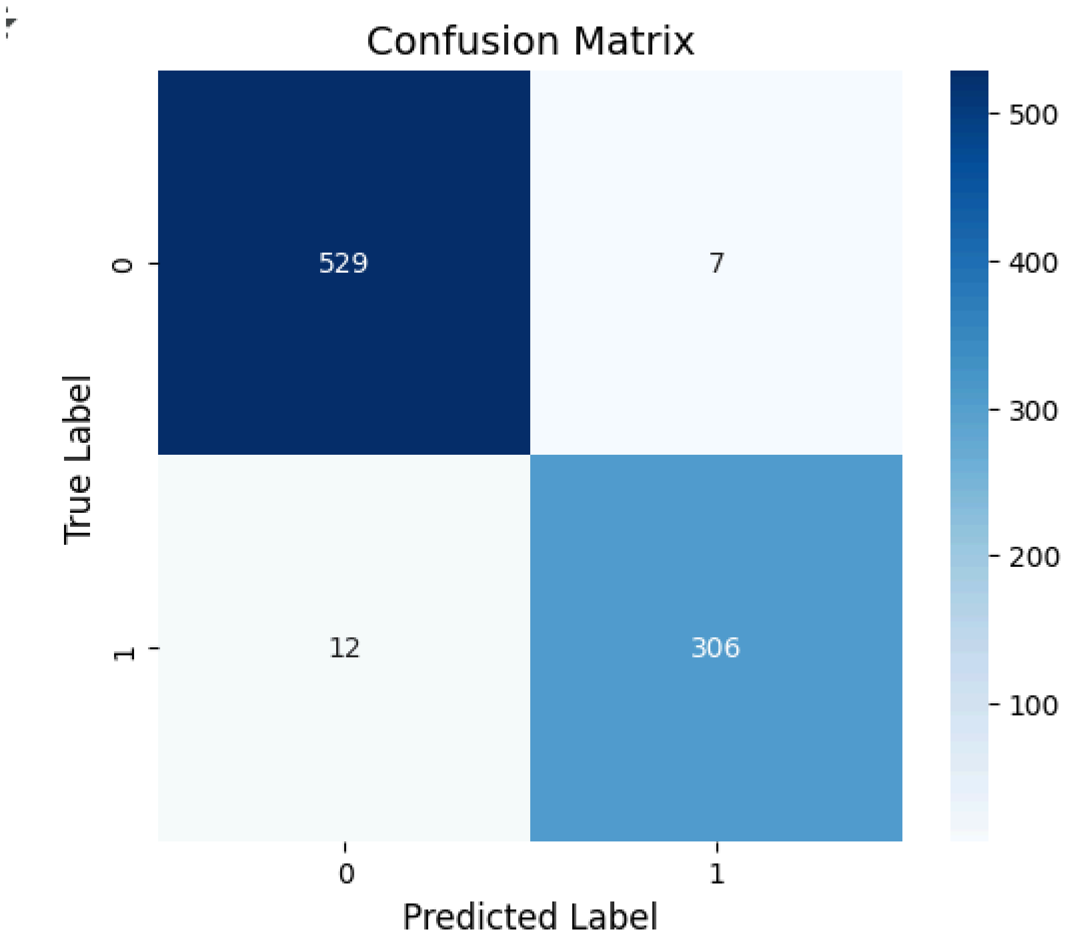
5. Evaluation Metrics

Scratch Implementation (Accuracy ~ 84.5%)



```
Accuracy : 0.8454332552693209
Precision: 0.8518863821138994
Recall   : 0.8454332552693209
F1-Score : 0.839485739611376
```

Scikit-learn Implementation (Accuracy ~ 97.8%)



```
✅ Model Performance
Accuracy : 0.977751756440281
Precision: 0.9777506845946926
Recall   : 0.977751756440281
F1-Score : 0.977715253694394
```

---

6. Comparison

Aspect	Scratch Implementation	Scikit-learn Implementation
Accuracy	84.5%	97.8%
Precision	85.2%	97.8%
Recall	84.5%	97.8%
F1-score	83.9%	97.8%
Training Time	High	Low

Ease of Implementation	Harder (manual math)	Very easy (one line)
Flexibility	Low	High (many hyperparameters)

7. Conclusion

Both implementations show the effectiveness of Random Forest in classification. The scratch model is useful for learning the underlying concepts but was slower (due to manual Python loops and lack of optimizations) and less accurate (~84.5%) with fewer trees, simpler splits, and limited randomness. The scikit-learn version, on the other hand, is highly optimized, scalable, and achieved ~98% accuracy with superior precision, recall, and F1-score. Overall, sklearn is more practical for real-world applications, while the scratch version is valuable for conceptual understanding.