|  |  |
| --- | --- |
| **Course- B. Tech** | **Type-** Specialization Elective |
| **Course Code-** CSET37 | **Course Name-** Big Data Analytics  and Business Intelligence |
| **Year-** 2024 | **Semester**- Odd |
|  | **Session-** 2022-2026 |

Student’s Name: - Shaurya Singh Enrollment No.: E22CSEU0539 Batch: - 18

PROBLEM STATEMENT : “Predicting Bank Loan Approval Status ”

1. Introduction

Predicting loan approval is an important task for financial institutions to evaluate loan applicants based on historical data and predictive models. This project leverages PySpark MLlib and other machine learning frameworks to build and test models that predict loan amounts (approved or not) using real loan data. Determine the most  efficient algorithm to be accurate.

**2. Objectives**

1. To preprocess and prepare a loan dataset for machine learning tasks using PySpark.
2. To implement and evaluate multiple machine learning models for loan status prediction.
3. To compare performance metrics like accuracy, precision, recall, and F1-score across models.
4. To provide meaningful insights into the impact of features like income, credit history, and loan amount.
5. To visualize data patterns and model performance to support conclusions.

3. Dataset Description and Preprocessing

Dataset Overview:

* The dataset includes fields such as Gender, Education, Income, LoanAmount, Credit History, and Loan\_Status.
* Target variable: Loan\_Status (Yes/No).

Preprocessing Steps:

* Missing Values: Imputed numerical features with mean values and categorical features with mode.
* Feature Engineering: Created Combined\_Income and Income\_Loan\_Ratio to capture better income dynamics.
* Encoding: Categorical variables were indexed using StringIndexer and one-hot encoded for modeling.
* Feature Scaling: Applied StandardScaler to normalize numerical features for better model performance.

Features Used:

1. LoanAmount
2. Loan\_Amount\_Term
3. Credit\_History
4. Combined\_Income
5. Income\_Loan\_Ratio
6. Gender, Married, Dependents, Education, Self\_Employed, Property\_Area (Encoded)
   1. Tables Hardware/Software/ Technique Used

|  |  |
| --- | --- |
| **Criteria** | **Details** |
| Hardware Configuration | Victus Laptop with AMD Ryzen 5 5600H (3.30 GHz), 8GB RAM |
| Software Configuration | Windows 11 (64-bit), Python 3.x, IDE: Colab Notebook |
| Big Data Tools Used with Version | PySpark (Version 3.x) |
| Python Library Used | pandas, numpy, scikit-learn, matplotlib, seaborn, statsmodels, pyspark |
| Visualization Tool | Matplotlib, Seaborn |
| Any other tools, libraries used | GridSearchCV, KNeighborsClassifier, RandomForestClassifier, LinearSVC, MulticlassClassificationEvaluator, BinaryClassificationEvaluator, confusion\_matrix, classification\_report, SVC, roc\_curve, auc |

* 1. **Implementation Methodology along with Flowchart**

The following methodology describes the entire process, outlining each phase, including data preprocessing, model training, and performance evaluation.

1. Data Collection

* The dataset containing applicant information such as income, credit history, and loan status is loaded into a Spark DataFrame.

2. Data Preprocessing

* Handling Missing Values: Missing data is handled by filling numerical columns with the mean and categorical columns with the most frequent value.
* Feature Engineering: Derived features, such as "Combined\_Income" (sum of applicant's and coapplicant's income) and "Income\_Loan\_Ratio," are created to provide deeper insights.
* Encoding Categorical Variables: Categorical columns like gender, education, and loan status are encoded using StringIndexer and OneHotEncoder.
* Normalization/Standardization: Numerical columns are standardized using StandardScaler to ensure equal weight across features.

3. Data Splitting

* The dataset is split into training (80%) and test (20%) sets using the randomSplit method in PySpark.

4. Dealing with Class Imbalance Using SMOTE

To ensure fairness, in the models predictions and prevent bias toward the class (such, as loan approval) the SMOTE (Synthetic Minority Over sampling Technique) methodology is used to balance the classes.

5. Model Training

* Multiple machine learning algorithms are trained on the dataset:
  + Decision Tree Classifier
  + Support Vector Machine (SVM)
  + K-Nearest Neighbors (KNN)
  + Random Forest Classifier
* Hyperparameters for models like KNN and Random Forest are fine-tuned using GridSearchCV.

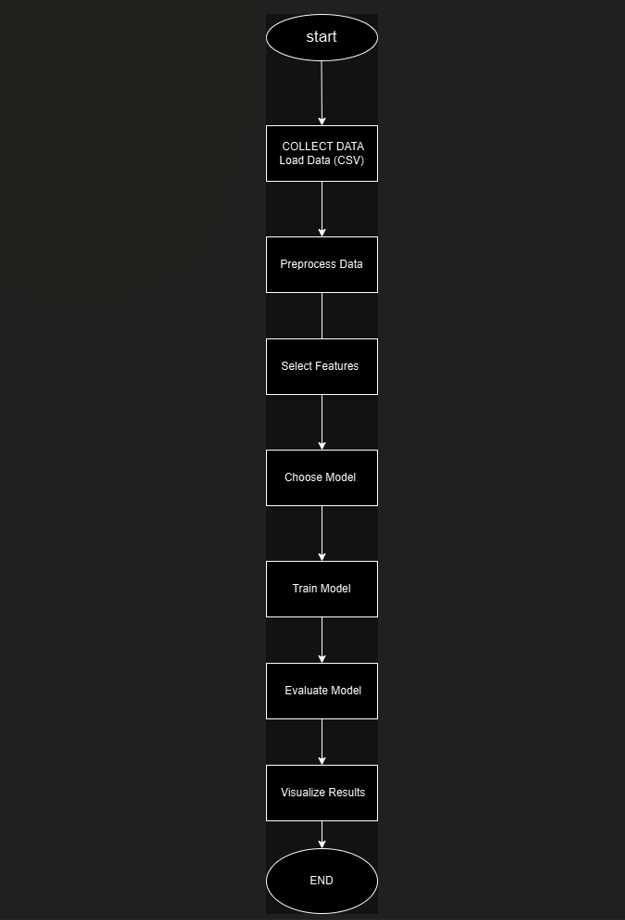
6. Model Evaluation

* Accuracy, Precision, Recall, F1-Score are computed to assess model performance.
* Models are compared based on these metrics to select the best-performing one.

7. Model Visualization

After testing the models, we proceed to visualize model performance and analyze the results:

* Confusion Matrix: A confusion matrix is created for the chosen model, displaying true positives, true negatives, false positives, and false negatives. This clearly demonstrates how well the model distinguishes between classes.
* ROC Curve and AUC: shows the trade-off between sensitivity (recall) and specificity, while the Area Under Curve (AUC) measures the model's capacity to distinguish between classes.
* Feature Importance: Visualizing feature relevance in models like Random Forest helps predict loan acceptance.
* **Flowchart**

****

1. **Comparative Analysis of PYSPARK ML Models**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model**  **Name** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **ROC** |
| **Decision Tree** | |  | | --- | | 0.7838 | | |  | | --- | | 0.7908 | | |  | | --- | | 0.7838 | | |  | | --- | | 0.7696 | | 0.73 |
| **SVM** | |  | | --- | | 0.8000 | | |  | | --- | | 1.00 | | |  | | --- | | 0.44 | | |  | | --- | | 0.62 | | 0.78 |
| **KNN** | |  | | --- | | 0.7432 | | |  | | --- | | 0.72 | | |  | | --- | | 0.98 | | |  | | --- | | 0.83 | | 0.66 |
| **Random Forest** | |  | | --- | | 0.8108 | | |  | | --- | | 0.79 | | |  | | --- | | 0.96 | | |  | | --- | | 0.87 | | 0.83 |
|  |  |  |  |  |  |

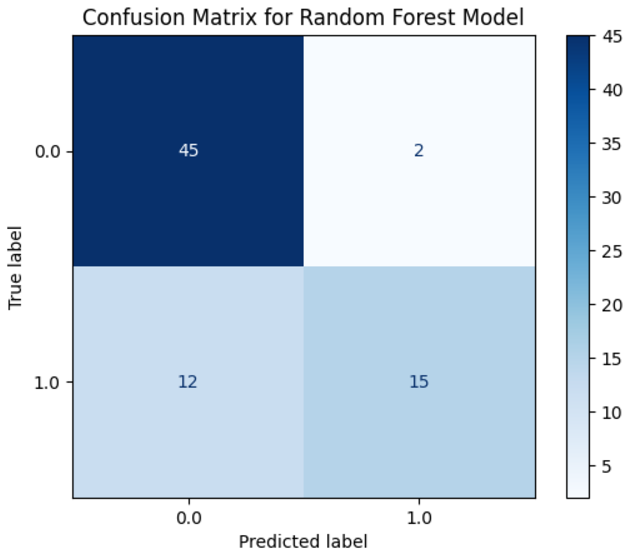
Analytics Findings

The Random Forest Classifier performed better than SVM, Decision Tree, and K-Nearest Neighbors (KNN) in the loan approval prediction challenge after evaluating various machine learning models. The Random Forest model achieved the highest accuracy in classification. This is probably because it uses an ensemble approach, training multiple decision trees and combining their results to make the final prediction, reducing the risk of overfitting and improving the model's ability to generalize. Random Forest stands out for its ability to handle outliers effectively and understand intricate relationships among data, a crucial aspect for predicting bank loan outcomes.

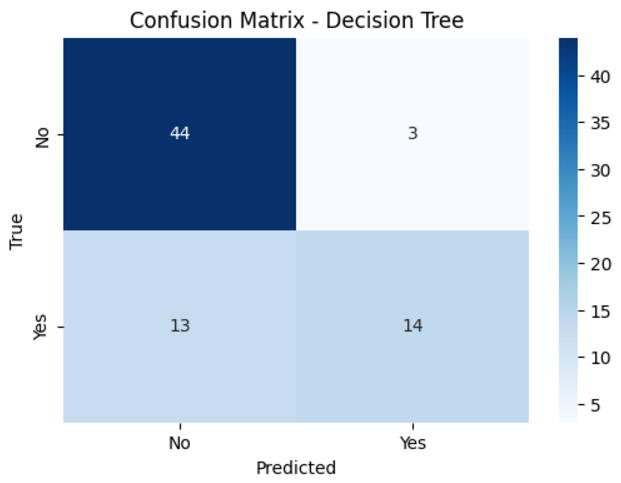
In contrast, SVM and KNN models found it challenging to equal Random Forest's performance. Although SVM works effectively when dealing with linear decision boundaries, it is susceptible to feature scale and might not handle nonlinearities as effectively as Random Forest. The Decision Tree model did well, but it was prone to overfitting, leading to often unsatisfactory results on new data. On the contrary, KNN showed good performance in certain situations, but struggled with class imbalance and did not have the same level of generalization as Random Forest. SMOTE fixed the imbalance, but Random Forest remained superior in handling the complexities of the data.

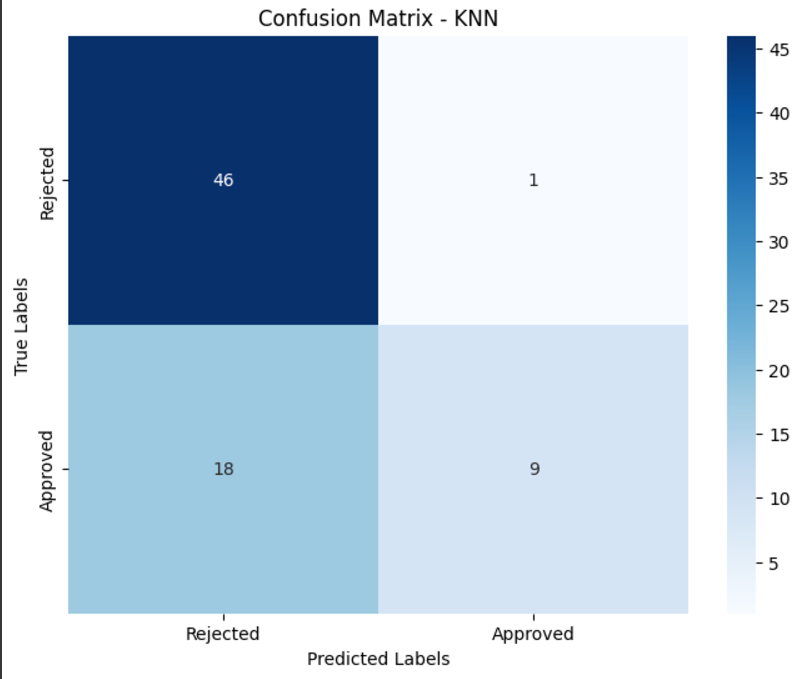
Model Evaluation Results

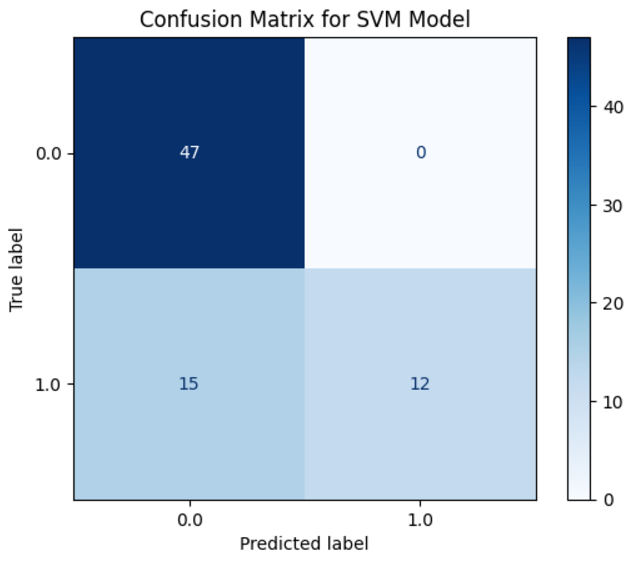
* 1. Confusion Matrix: The optimal Random Forest model demonstrates a high quantity of accurate predictions for granted loans (true positives) and rejected loans (true negatives). This implies that the model's categorization is very precise. The matrix can aid in identifying misclassifications and suggesting further enhancements in feature creation or model adjustment. Below is the confusion matrix representation of Random Forest model:



Description: A high true positive and true negative count indicates that the Random Forest classifier accurately predicts authorized and denied loan statuses.

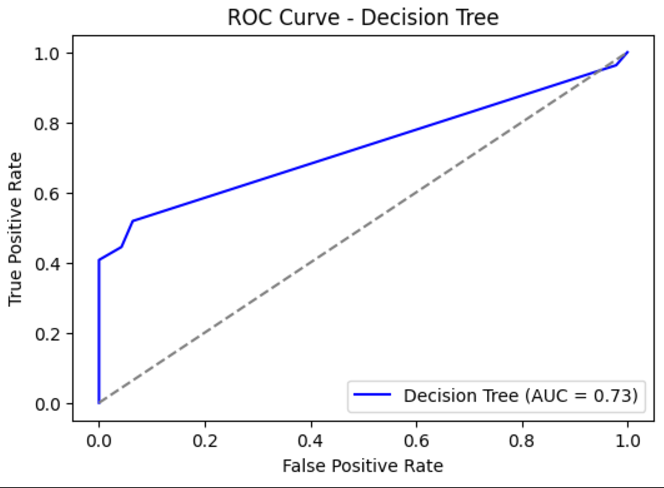


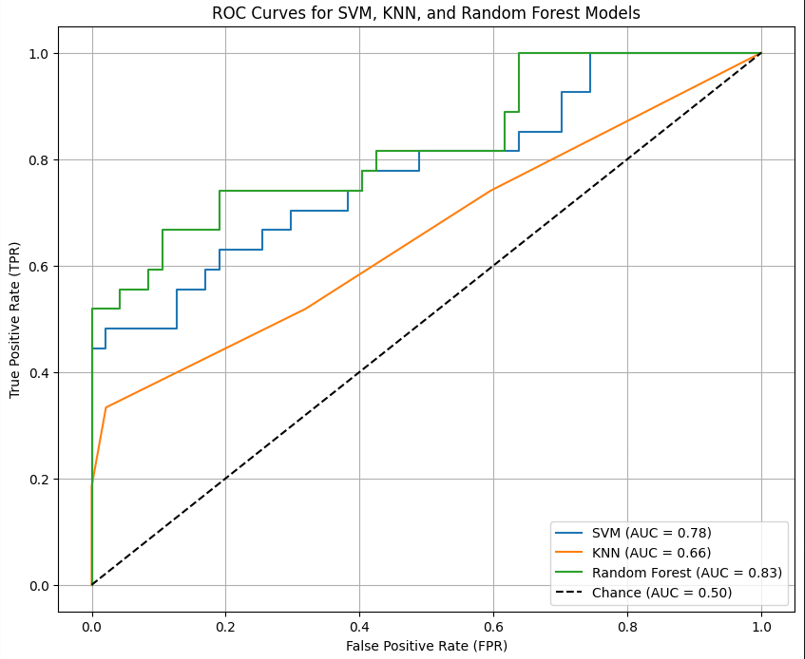




2. ROC Curve and AUC:

The ROC curve of Random Forest is another sign of its exceptional effectiveness. The Random Forest model has an AUC (Area Under Curve) that is near 1, showing outstanding performance with a nice trade-off between sensitivity (True Positive Rate) and specificity (False Positive Rate). A random model would have an AUC of 0.5, so this high AUC score affirms that Random Forest is accurately predicting loan approvals and rejections.  
  
Explanation: The ROC curve demonstrates how well the model can differentiate positive and negative classes, while the AUC offers a condensed evaluation of its overall effectiveness.



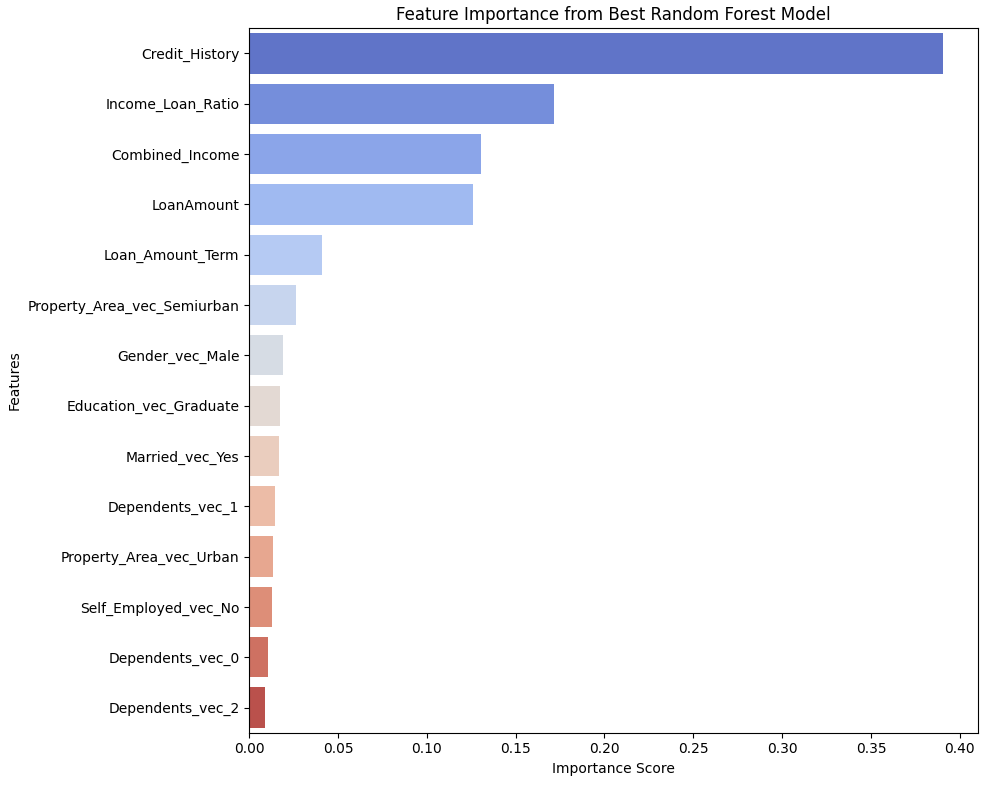


3. Feature Importance:

The Random Forest model also provides insight into feature importance, helping to understand which variables contribute the most to predicting loan approval. The top features influencing the model include Credit\_History, Income\_loan\_ratio and LoanAmount. This indicates that the applicant's financial history and creditworthiness play a vital role in determining loan approval. Visualizing feature importance can assist in refining feature engineering by concentrating on the most influential variables.

Description: Feature importance analysis helps prioritize the features that are most significant for loan approval predictions, which can be valuable for improving both model accuracy and business insights.

These results demonstrate that the Random Forest model is the most effective choice for the loan approval prediction problem, with high accuracy, strong discriminative power as shown by the ROC curve, and valuable insights from feature importance.



1. **Data Visualization.**

Data visualization is crucial for comprehending the dataset's structure and the connections among various features. Here are a few essential visuals that aid in enhancing comprehension of the data for the bank loan prediction initiative.

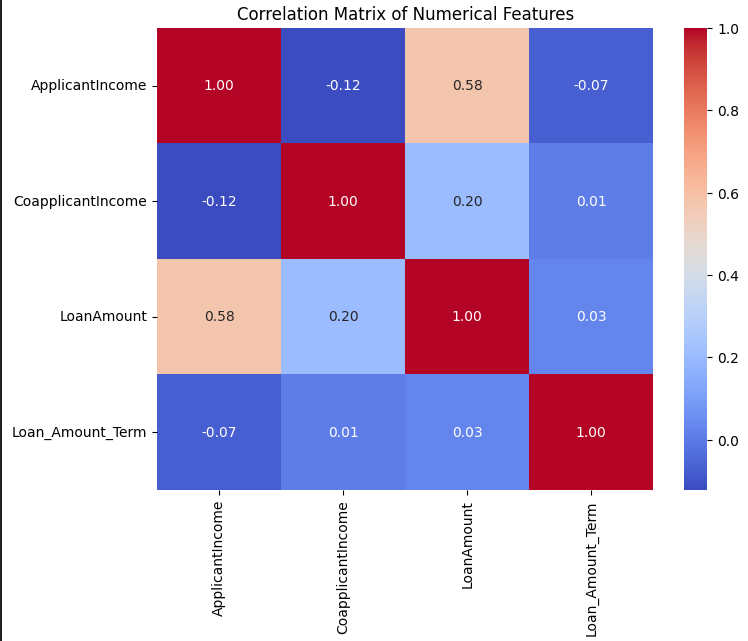
**a) Distribution of Loan Status**

A bar plot showing the distribution of loan approval status helps to understand the balance between approved and rejected loans. If the dataset is imbalanced, some models may require extra attention (such as using balanced weights or sampling methods).



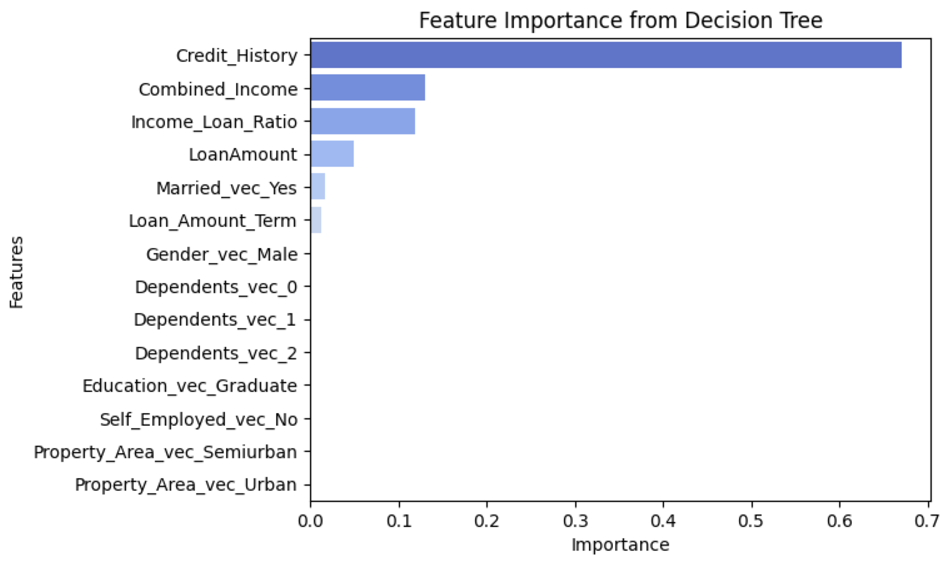
**b) Correlation Heatmap**

Visualizing correlations between different numerical features can reveal any strong relationships that could be helpful for predictive modelling.



**c) Feature Importance (for Random Forest Model)**

Feature importance can give us insight into which features are most influential in predicting loan approval status.



**d) ROC Curve**

The ROC curve assesses how well classification models perform by displaying the balance between true positive and false positive rates across different thresholds.  
  
These visualizations assist in evaluating data quality, importance of features, and performance of the model, while also offering suggestions for enhancing the prediction system.

1. **Conclusion**

To sum up, the bank loan prediction project used PySpark's strong data processing abilities to predict loan approval outcomes. Due to its ability to prevent overfitting and handle complex feature interactions well, the Random Forest model outperformed other classifiers like KNN and Logistic Regression, especially on big datasets.  
  
Improving model accuracy necessitated data preprocessing steps such as scaling numeric attributes and encoding categorical values. Examples of visualization tools, such as confusion matrices, ROC curves, and feature importance plots, help explain the model's results and showcase key variables impacting loan approval choices.  
  
Due to PySpark's ability to scale, large datasets can be managed efficiently, enhancing the model's robustness and suitability for real financial data. Future advancements may focus on adjusting hyperparameters with precision and exploring a broader array of datasets for enhancement.

# Relevant References

## Kaggle Dataset:

* [Bank Loan Prediction Dataset](https://www.kaggle.com/)

## Scikit-learn Documentation:

* [Scikit-learn User Guide](https://scikit-learn.org/stable/user_guide.html)

# Matplotlib Documentation:

* [Matplotlib](https://www.w3schools.com/python/matplotlib_pyplot.asp)

# Seaborn Documentation:

## [Seaborn](https://www.w3schools.com/python/numpy/numpy_random_seaborn.asp)

Annexure - 1: Running Source Code

<https://colab.research.google.com/drive/10lYEW2BJUy_artM3lq3uT_TzjxtQAU9j?usp=sharing>