## **Report**

## **1. Approach Taken**

1. **Data Preprocessing:**
   * Identified and handled different types of columns:
     + Categorical columns (17 columns including 'HDB BRANCH NAME', 'FIRST NAME', 'ASSET CTG', etc.)
     + Date columns ('APPLICATION LOGIN DATE')
     + Boolean columns ('last\_name\_from\_pan')
   * Converted date columns to a suitable numeric format (likely timestamp or ordinal encoding)
   * Converted boolean columns to numeric format
   * Presumably applied one-hot encoding to categorical columns
   * Scaled the numerical features (evidenced by the "Preprocessing and scaling completed successfully!" message)
2. **Model Selection:**
   * Based on the output, a binary classification model is used (likely a Random Forest method)
   * The model was trained to predict two classes: 'APPROVED' and 'DECLINED'
3. **Model Training:**
   * The data was split into training and validation sets (exact split ratio not provided, but commonly 80-20 or 70-30)
   * The model was trained on the preprocessed and scaled data
4. **Model Evaluation:**
   * The model was evaluated using standard classification metrics: precision, recall, f1-score, and support
   * A confusion matrix was generated to visualize the model's performance

## **2. Insights and Conclusions from Data**

1. **Class Distribution:**
   * The dataset is imbalanced:
     + 'APPROVED' class: 1327 samples (66.35% of the dataset)
     + 'DECLINED' class: 673 samples (33.65% of the dataset)
   * This imbalance may affect the model's performance and should be considered when interpreting results
2. **Model Performance:**
   * Overall accuracy: 83% (1661 correct predictions out of 2000 samples)
   * The model performs differently for each class:
     + 'APPROVED' class: High precision (98%) but lower recall (76%)
     + 'DECLINED' class: Lower precision (67%) but high recall (97%)
3. **Interpretation of Results:**
   * The model is very confident when it predicts an application will be approved (98% precision)
   * However, it misses about 24% of the approved applications (76% recall for 'APPROVED')
   * The model catches almost all declined applications (97% recall for 'DECLINED')
   * But it has a higher false positive rate for declined applications (67% precision for 'DECLINED')
4. **Business Implications:**
   * The model is conservative in approving applications, which might be good for risk management
   * However, it might be declining some applications that could have been approved (320 false negatives)
   * Only a small number of actually declined applications are incorrectly approved (19 false positives)

## **3. Performance on Train Data Set**

**Note:** The provided metrics are likely from a validation set, not the training set. Training set performance is typically higher due to potential overfitting. However, we'll analyze the given metrics as they provide valuable insights into the model's performance.

1. **Accuracy:** 0.83 (83%)
   * The model correctly classifies 83% of all samples
2. **Precision:**
   * 'APPROVED': 0.98
   * 'DECLINED': 0.67
   * Macro Average: 0.83
   * Weighted Average: 0.88
3. **Recall:**
   * 'APPROVED': 0.76
   * 'DECLINED': 0.97
   * Macro Average: 0.87
   * Weighted Average: 0.83
4. **F1-Score:**
   * 'APPROVED': 0.86
   * 'DECLINED': 0.79
   * Macro Average: 0.83
   * Weighted Average: 0.84

**Confusion Matrix:** Copy  
[[1007 320]

1. [ 19 654]]  
   * True Negatives (Correctly Predicted Declines): 654
   * False Positives (Incorrectly Predicted Approvals): 19
   * False Negatives (Incorrectly Predicted Declines): 320
   * True Positives (Correctly Predicted Approvals): 1007
2. **Interpretation of Metrics:**
   * The model has high precision for the 'APPROVED' class, meaning when it predicts an approval, it's usually correct
   * It has high recall for the 'DECLINED' class, meaning it catches most of the applications that should be declined
   * The F1-score, which balances precision and recall, is reasonably good for both classes (0.86 and 0.79)
   * The weighted averages are generally higher than the macro averages, indicating better performance on the majority class ('APPROVED')

## **Recommendations and Next Steps**

1. **Address Class Imbalance:** Consider techniques like SMOTE or class weighting to balance the classes and potentially improve the performance of the minority class.
2. **Feature Engineering:** Analyze the importance of different features and consider creating new features that might help distinguish between approved and declined applications.
3. **Model Tuning:** Experiment with hyperparameter tuning to optimize the model's performance, particularly to balance precision and recall for both classes.
4. **Threshold Adjustment:** Consider adjusting the classification threshold to balance false positives and negatives based on business requirements.
5. **Ensemble Methods:** Try combining multiple models or potentially using advanced ensemble techniques to improve overall performance.
6. **Cost-Sensitive Learning:** If the costs of false positives and false negatives are known, incorporate them into the model training process.
7. **Regular Monitoring:** Set up a system to regularly monitor the model's performance on new data and retrain as necessary to prevent performance degradation over time.