Abdulrahman Sherif

البنك العربي
ARAB BANK
Success is a journey

Enhanced Financial Experience Through AI Personalization

Abstract

This document outlines the idea of development that goes beyond basic account management by integrating AI algorithms to provide personalized financial advice based on client behavior and account balances. A decision tree algorithm is employed to trigger specific financial recommendations when certain conditions are met. The project also explores the importance of upgrading algorithms as data volume increases, emphasizing Six Sigma principles for process improvement.

Introduction

In the age of digital transformation, personalized banking experiences are becoming increasingly critical to customer satisfaction and retention. Traditional banking services are evolving, with clients expecting not only account management but also personalized financial advice that helps them make informed decisions. This project aims to develop a web application that serves as a comprehensive banking interface, incorporating advanced AI algorithms to offer tailored financial advice. The application is built using Java Spring Boot and includes features such as balance display, loan reminders, credit card suggestions, and loyalty point management.

Problem Statement

The rapid growth in digital banking services has created a demand for more personalized and intelligent financial advice. Clients often need reminders for due loans, suggestions for taking out new loans, or advice on the best payment options based on their spending patterns. The challenge lies in developing a system that can efficiently analyze client data and provide real-time personalized recommendations. Furthermore, as the amount of data increases, the need for more sophisticated algorithms becomes apparent.

AI-Powered Personalization

To create a truly impactful financial experience, leveraging AI-powered personalization is essential.

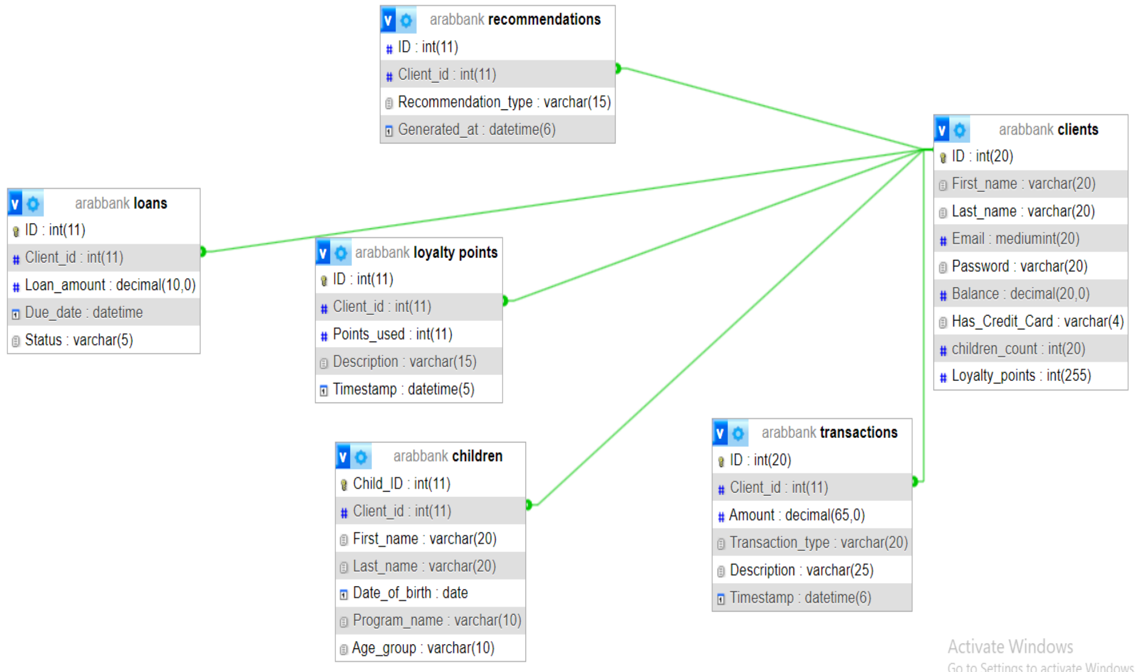
- **Define:** Recognize the need to provide users with personalized financial advice, product recommendations, and alerts based on their behavior and financial goals.
- **Measure:** Evaluate the current effectiveness of personalized services and the impact on user engagement and satisfaction.
- **Analyze:** Analyze user data to identify patterns and behaviors that can be used to deliver more personalized experiences.
- **Improve:** Implement AI-driven personalization to offer tailored financial services, enhancing user experience and engagement.
- **Control:** Monitor the effectiveness of personalization, gather user feedback, and refine AI models to continuously improve the relevance of personalized content.

Database Design

The database for the banking application is designed to store and manage client information, financial transactions, and related data necessary for providing personalized financial advice. The database schema consists of several interconnected tables, as illustrated in the ER diagram provided:

1. **Clients:** Stores personal information about the clients, including their ID, first and last names, email, password, balance, credit card status, children count, and loyalty points. This table serves as the central table linking to others.
2. **Loans:** Contains details about loans taken by clients, such as loan ID, client ID (foreign key), loan amount, due date, and status.
3. **Transactions:** Records the transactions made by clients, including transaction ID, client ID (foreign key), amount, transaction type, description, and timestamp.
4. **Loyalty Points:** Tracks loyalty points earned and used by clients, with fields such as points used, description, and timestamp.
5. **Recommendations:** Stores the financial advice provided to clients, including the recommendation type and the timestamp when it was generated.
6. **Children:** Contains information about the children of clients, including child ID, client ID (foreign key), first name, last name, date of birth, program name, and age group.

This database design ensures efficient storage and retrieval of data, facilitating the implementation of the decision tree algorithm for personalized financial advice. As the dataset grows, more advanced algorithms like Random Forests or Gradient Boosting may be employed to enhance the accuracy and relevance of the advice provided.



Algorithms

1. **External Algorithm** (https://www.cascading.ai/)

   Cascading AI are revolutionizing outdated legacy technology by introducing AI-native systems of record that automate 90% of the manual work previously handled by humans. Their flagship product, an AI-native Loan Origination System, features an AI Agent named Sarah. Sarah seamlessly interacts with small business loan applicants through email or SMS, guiding them through the application process and preparing the loan file for a human underwriter. By streamlining this process, we enable banks to efficiently fund small businesses, which are the backbone of the American economy, without the high manual effort, thereby unlocking affordable and quick funding options.

Key AI Agents:

- **Sarah (Lending & Onboarding):** Engages with small business loan applicants, ensuring they complete their applications by persistently following up via email or SMS.
- **Leo (Lending & Onboarding):** Works tirelessly to collect all necessary documents from loan applicants, ensuring no application is left incomplete.
- **Tom (Customer Service):** Navigates outdated banking systems to surface the right information and drafts accurate responses to customer service inquiries.
- **Anna (Back-Office):** Analyzes unstructured financial data to resolve issues such as payment halts due to minor errors, ensuring smooth operations.

The cascading AI agents integrate with existing IT infrastructure, use computer vision and conversational AI to interact with systems and customers, and align with banking regulations through human-in-the-loop systems. These agents transform banking operations, making processes more efficient and unlocking new opportunities for banks to serve small businesses effectively.

2. Decision Tree (In House Algorithm)

A decision tree is a popular machine learning algorithm that is used to model decisions and their possible consequences, including chance event outcomes, resource costs, and utility. In the context of this banking application, a decision tree algorithm is employed to analyze client data and provide personalized financial advice. The algorithm uses a tree-like model of decisions, where each internal node represents a "test" or condition (e.g., whether the client's balance is below a certain threshold), each branch represents the outcome of the test, and each leaf node represents the final decision or action (e.g., suggesting a credit card).

Six Sigma is a data-driven approach aimed at process improvement by reducing defects and ensuring quality. In the context of this banking application, Six Sigma principles can be applied to improve the accuracy and relevance of the financial advice provided. The project follows the DMAIC (Define, Measure, Analyze, Improve, Control) methodology:

-**Define:** Identify the specific financial advice features that need to be implemented, such as loan reminders, credit card suggestions, and loyalty point management.

- **Measure**: Track the accuracy and relevance of the advice provided to clients based on their financial behavior.

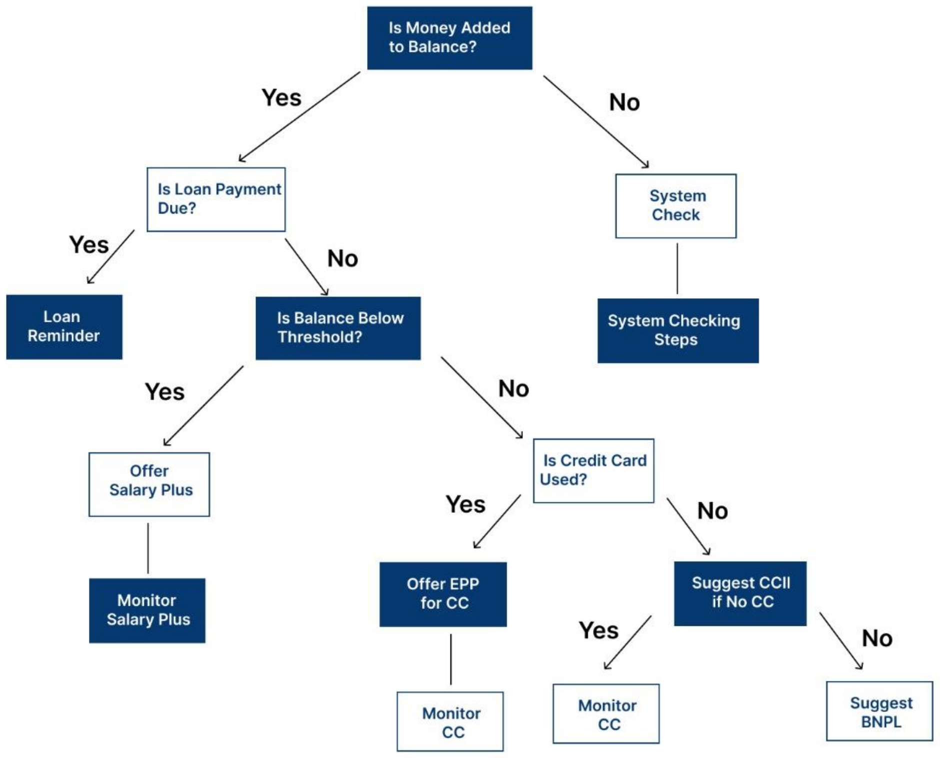
- **Analyze**: Identify any discrepancies or areas where the advice could be more personalized or timely.

- **Improve:** Refine the decision tree algorithm to better analyze client data and provide more accurate recommendations.

- **Control:** Continuously monitor the performance of the algorithm and make adjustments as necessary to maintain a high level of service quality.

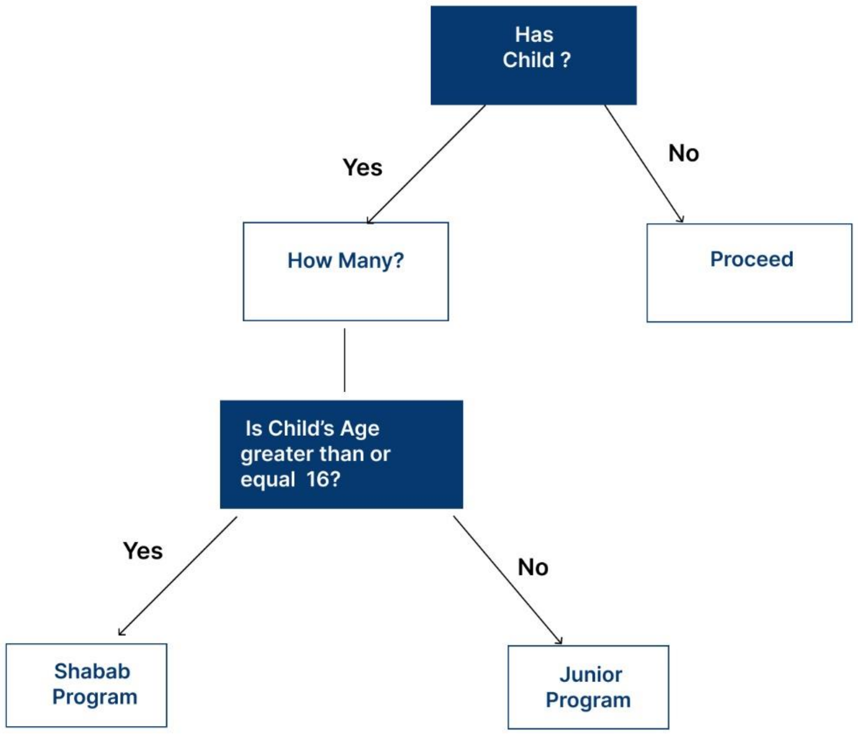
Decision Tree Diagram

Case1:



The system first checks if money has been added to your balance. If so, it verifies if a loan payment is due and sends a reminder if necessary. If there's no loan payment and your balance is sufficient, the system assesses your credit card usage. If you have a credit card, an Extended Payment Plan (EPP) might be offered. However, if you don't use a credit card, the system checks for available Buy Now, Pay Later (BNPL) options. If a BNPL offer exists, it will be presented to you. If not, and you lack a credit card, the system might recommend a Credit Card Issuance (CCII).

Case 2:



The decision tree for determining the appropriate program for the client's children begins by assessing the age of the child. If the child is younger than 16, they are assigned to the Junior Program, which is designed for younger individuals. If the child is 16 or older, they are assigned to the Shabab Program, tailored for older youth. This straightforward process ensures that each child is placed in a program that best fits their age and developmental stage, enhancing the relevance and effectiveness of the offered programs.

Decision Tree Implementation

Case1:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier, plot_tree

from sklearn.metrics import accuracy_score, classification_report

import matplotlib.pyplot as plt

# Define the data

data = {

    "money_added_to_balance": [True, True, True, True, False, False, False, False],

    "loan_payment_due": [True, True, False, False, False, True, False, False],

    "balance_below_threshold": [False, True, True, False, False, False, True, False],

    "credit_card_used": [False, False, True, True, False, False, True, False],

    "has_credit_card": [True, True, True, True, True, False, False, True],

    "recommendation": [

        "Loan Reminder", "Offer Salary Plus", "Offer Salary Plus",

        "Offer EPP for CC", "Suggest CCII", "System Check",

        "System Checking Steps", "Suggest CCII if No CC"

    ]}

df = pd.DataFrame(data)

# Separate the features and the target variable

X = df.drop("recommendation", axis=1)

y = df["recommendation"]

# Split the data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the DecisionTreeClassifier

clf = DecisionTreeClassifier(random_state=42)

clf.fit(X_train, y_train)

# Make predictions on the test set

predictions = clf.predict(X_test)

# Evaluate the accuracy of the model

accuracy = accuracy_score(y_test, predictions)

print("Accuracy:", accuracy)

# Generate classification report

# Use labels that match the classes in predictions and y_test

report_str = classification_report(y_test, predictions, labels=clf.classes_, zero_division=1)

print(report_str)

# Plot the decision tree

plt.figure(figsize=(20, 12))

plot_tree(clf, feature_names=X.columns, class_names=clf.classes_, filled=True, rounded=True)            plt.show()
```

Case 2:

```
from sklearn.tree import DecisionTreeClassifier

import numpy as np


# Sample dataset

# 0 = No, 1 = Yes

data = {

    "has_child": [0, 1, 1, 1, 1, 1, 1, 1],

    "child_age": [0, 10, 12, 14, 15, 16, 17, 18],

    "program": ["Proceed", "Junior Program", "Junior Program", "Junior Program",

                "Junior Program", "Shabab Program", "Shabab Program", "Shabab Program"]

}


# Convert to numpy arrays for sklearn

X = np.array([data["has_child"], data["child_age"]]).T

y = np.array(data["program"])


# Create and train the decision tree

clf = DecisionTreeClassifier()

clf.fit(X, y)


# Predict for a new sample

sample = np.array([[1, 13]])  # Example: has a child aged 13

prediction = clf.predict(sample)

print("Predicted Program:", prediction[0])


# Output the structure of the tree

from sklearn import tree

import matplotlib.pyplot as plt


plt.figure(figsize=(12, 8))

tree.plot_tree(clf, feature_names=["Has Child", "Child Age"], class_names=["Proceed", "Junior Program", "Shabab Program"],
filled=True)

plt.show()
```

For our implementation there are key metrics used in machine learning evaluation, particularly in classification problems as well as other metrics used to evaluate the overall performance of a classification model has been utilized.

Precision

- **Definition:** Measures how many of the positive predictions made by your model were actually correct.
- **Formula:** Precision = True Positives / (True Positives + False Positives)
- **Interpretation:** A high precision indicates that when your model predicts a positive outcome, it's likely to be correct.

Recall

- **Definition:** Measures how many of the actual positive cases your model was able to correctly identify.
- **Formula:** Recall = True Positives / (True Positives + False Negatives)
- **Interpretation:** A high recall indicates that your model is good at identifying all the positive cases.

F1-Score

- **Definition:** A harmonic mean of precision and recall, providing a single metric that balances both.
- **Formula:** F1-Score = 2 * (Precision * Recall) / (Precision + Recall)
- **Interpretation:** A high F1-score indicates that your model is good at both precision and recall.

Support

- **Definition:** The number of instances in a class.
- **Interpretation:** It's important to consider support when evaluating metrics, as imbalanced datasets can skew results.

Accuracy

- **Definition:** The proportion of correct predictions out of the total number of predictions.
- **Formula:** Accuracy = (True Positives + True Negatives) / (Total Instances)
- **Interpretation:** A high accuracy indicates that the model is making correct predictions overall.

Macro Average

- **Definition:** The arithmetic mean of the precision, recall, and F1-score calculated for each class.
- **Formula:** Macro Average = (Precision_class1 + Precision_class2 + ...) / N
- **Interpretation:** This metric gives equal weight to each class, regardless of their support.

Weighted Average

- **Definition:** The average of precision, recall, and F1-score weighted by the support of each class.
- **Formula:** Weighted Average = (Support_class1 * Precision_class1 + Support_class2 * Precision_class2 + ...) / Total Support
- **Interpretation:** This metric takes into account the class distribution, giving more weight to classes with higher support.

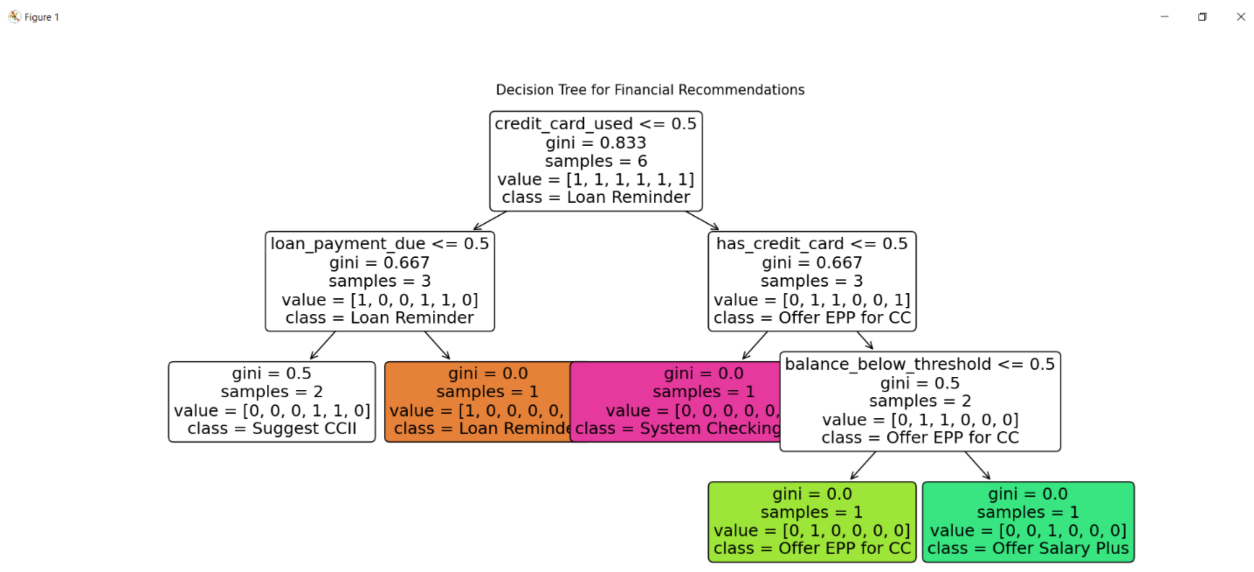
Run Time

- Overall Process: The decision tree works by splitting the data based on the conditions (e.g., credit card usage, loan payment due, etc.), and at each step, it tries to narrow down the recommendations until it reaches a final decision (leaf node).

- Class Prediction: Each leaf node represents a final recommendation based on the decisions made from the root to that leaf.

- Gini Index: At each step, the Gini index indicates how well the decision split the data. Lower Gini index values mean that the node is purer (closer to having all samples of one class).

Case 1:



1. Root Node:

- Condition: The decision tree starts with the condition `credit_card_used <= 0.5`.

- Explanation: This node checks if the client has used a credit card. If `credit_card_used` is `0` (no, which is ≤ 0.5), the left branch is followed. If `credit_card_used` is `1` (yes, which is > 0.5), the right branch is followed.

2. Gini Index:

- Definition: The Gini index is a measure of impurity or diversity. A Gini index of `0` indicates that all cases in the node belong to a single class. A higher Gini index means there is more diversity within the node.

- Interpretation in the Root Node: A Gini index of `0.833` means the node is quite impure, meaning the samples are diverse and could belong to different classes.

3. Samples:

- Definition: This represents the number of data samples or instances that reach this particular node.

- In the Root Node: There are 6 samples, which means 6 instances in the dataset meet this condition (`credit_card_used <= 0.5`).

4. Value:

- Definition: This array shows the count of samples for each class at this node.

-In the Root Node: `[1, 1, 1, 1, 1, 1]` indicates that each class (e.g., "Loan Reminder", "Offer Salary Plus", etc.) has exactly 1 instance among the 6 samples reaching this node.

5. Class:

- Definition: The predicted class for this node, i.e., the class with the majority of instances.

- In the Root Node: The class is "Loan Reminder," meaning this is the most common recommendation for the samples that reach this node.

First Level Nodes

1. Left Child Node (loan_payment_due <= 0.5):

- **Condition:** Checks if a loan payment is due.

-**Gini Index:** `0.667`, indicating some impurity. This means there is a mix of recommendations among the 3 samples at this node.

- **Samples:** There are 3 samples.

- **Value:** `[1, 0, 0, 1, 1, 0]`, meaning one sample is "Loan Reminder", one is "Offer Salary Plus", and one is "Suggest CCII".

- **Class:** The class with the most samples here is "Loan Reminder".

2. Right Child Node (has_credit_card <= 0.5):

- **Condition:** Checks if the client has a credit card.

- **Gini Index:** `0.667`, which also indicates impurity.

- **Samples:** 3 samples reach this node.

- **Value:** `[0, 1, 1, 0, 0, 1]`, meaning there's one instance each of "Offer EPP for CC", "Offer Salary Plus", and "System Check".

- **Class:** The class predicted at this node is "Offer EPP for CC"

Second Level Nodes

1. Left Child of "loan_payment_due <= 0.5" (leaf nodes):

- **Two leaf nodes:**

  - One node has `gini = 0.5`, `samples = 2`, with value `[0, 0, 0, 1, 1, 0]`, and the class is "Suggest CCII".

  - The other node has `gini = 0.0`, `samples = 1`, with value `[1, 0, 0, 0, 0, 0]`, and the class is "Loan Reminder".

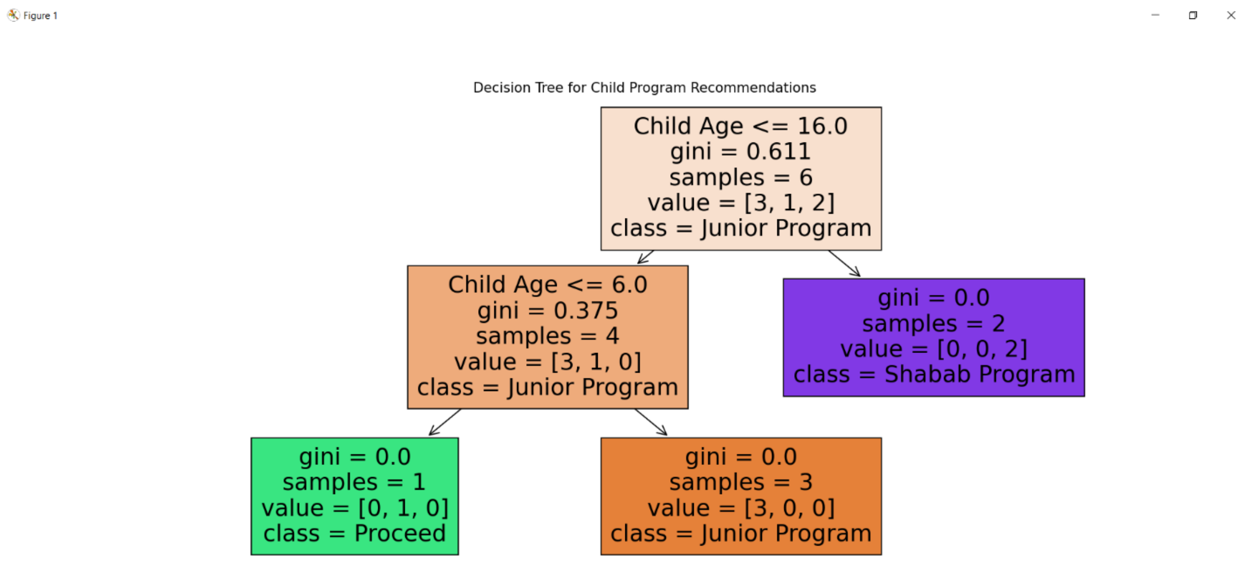
2. Right Child of "has_credit_card <= 0.5" (balance_below_threshold <= 0.5):

- **Condition:** Checks if the client's balance is below a certain threshold.
- **Gini Index:** `0.5`, indicating moderate impurity.
- **Samples:** 2 samples.
- **Value:** `[0, 1, 1, 0, 0, 0]`, with one sample each for "Offer Salary Plus" and "Offer EPP for CC".
- **Class:** "Offer EPP for CC" is predicted here.

3. Leaf Nodes under "balance_below_threshold <= 0.5":

- **Left Leaf Node:**
  - **`Gini Index`**: `0.0`, meaning it is pure**.**
  - **Samples:** 1.
  - **Value:** `[0, 1, 0, 0, 0, 0]`, class is "Offer EPP for CC".
- **Right Leaf Node:**
  - **`Gini Index`:** `0.0`, meaning it is pure.
  - **Samples:** 1**.**
  - **Value:** `[0, 0, 1, 0, 0, 0]`, class is "Offer Salary Plus".\

Case 2:



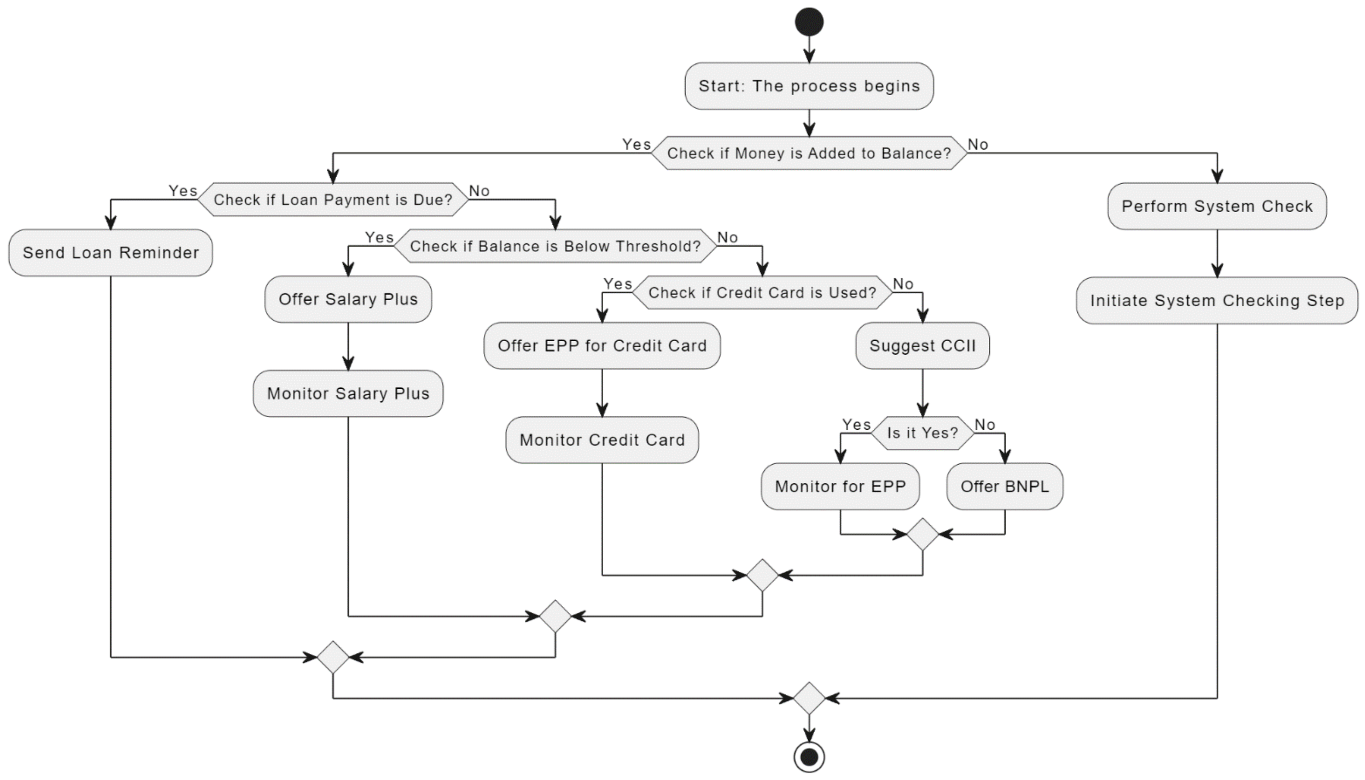
First Level Nodes

1. **Left Child Node (Child Age <= 6.0):**
    - **Condition:** Checks if the child's age is 6 years old or younger.
    - **Gini Index:** 0.375, indicating low impurity but still some diversity.
    - **Samples:** There are 4 samples.
    - **Value:** [3, 1, 0], meaning 3 samples belong to the "Junior Program" class, and 1 sample belongs to the "Proceed" class.
    - **Class:** The predicted class here is "Junior Program."
2. **Right Child Node (Child Age > 16.0):**
    - **Condition:** Represents cases where the child is older than 16 years.
    - **Gini Index:** 0.0, which means this node is pure—all samples belong to the same class.
    - **Samples:** 2 samples reach this node.
    - **Value:** [0, 0, 2], indicating both samples belong to the "Shabab Program" class.
    - **Class:** The predicted class is "Shabab Program."

Second Level Nodes

1. **Left Child of "Child Age <= 6.0" (leaf nodes):**
    - **Two leaf nodes:**
        - **Left Leaf Node:**
            - **Gini Index:** 0.0, meaning it's pure.
            - **Samples:** 1 sample.
            - **Value:** [0, 1, 0], with the class being "Proceed".
        - **Right Leaf Node:**
            - **Gini Index:** 0.0, meaning it's pure.
            - **Samples:** 3 samples.
            - **Value:** [3, 0, 0], with the class being "Junior Program".

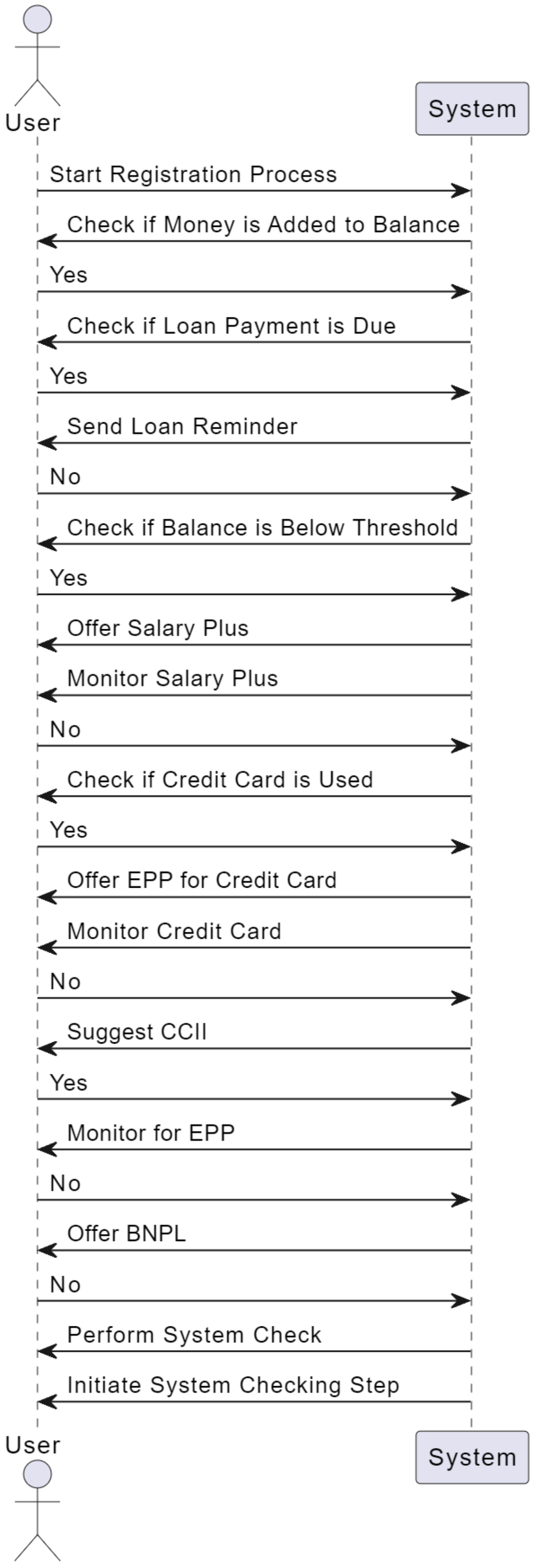
Activity Diagram


Start: The process begins
Check if Money is Added to Balance?
Yes
No
Check if Loan Payment is Due?
Yes
No
Send Loan Reminder
Check if Balance is Below Threshold?
Yes
No
Offer Salary Plus
Monitor Salary Plus
Check if Credit Card is Used?
Yes
No
Offer EPP for Credit Card
Monitor Credit Card
Suggest CCII
Is it Yes?
Yes
No
Monitor for EPP
Offer BNPL
Perform System Check
Initiate System Checking Step

The process begins by checking if money has been added to the user's balance. If so, it verifies if a loan payment is due and sends a reminder if necessary. If there's no loan payment and the balance is sufficient, the system assesses credit card usage. If a credit card is used, an Extended Payment Plan (EPP) might be offered. Otherwise, the system checks for available Buy Now, Pay Later (BNPL) options. If a BNPL offer exists, it will be presented to the user. If not, and the user lacks a credit card, the system might recommend a Credit Card Issuance (CCII). Finally, if the user's financial situation doesn't meet any of these criteria, a system check is performed, followed by the initiation of system checking steps.

Sequence Diagram

The system begins by checking if money has been added to the user's balance. If so, it verifies if a loan payment is due and sends a reminder if necessary. If there's no loan payment and the balance is sufficient, the system assesses credit card usage. If a credit card is used, an Extended Payment Plan (EPP) might be offered. Otherwise, the system checks for available Buy Now, Pay Later (BNPL) options. If a BNPL offer exists, it will be presented to the user. If not, and the user lacks a credit card, the system might recommend a Credit Card Issuance (CCII). Finally, if the user's financial situation doesn't meet any of these criteria, a system check is performed, followed by the initiation of system checking steps.


User
System
Start Registration Process
Check if Money is Added to Balance
Yes
Check if Loan Payment is Due
Yes
Send Loan Reminder
No
Check if Balance is Below Threshold
Yes
Offer Salary Plus
Monitor Salary Plus
No
Check if Credit Card is Used
Yes
Offer EPP for Credit Card
Monitor Credit Card
No
Suggest CCII
Yes
Monitor for EPP
No
Offer BNPL
No
Perform System Check
Initiate System Checking Step
User
System

Scaling with Data Growth

As the client base and associated data grow, the decision tree algorithm may become less efficient due to increased complexity and the potential for overfitting. To address this, more advanced algorithms such as Random Forest or Gradient Boosting can be implemented. These algorithms offer better performance with larger datasets by aggregating the decisions from multiple trees to reduce variance and improve prediction accuracy.

Examples of Advanced Algorithms

- **Random Forest:** An ensemble method that builds multiple decision trees and merges them together to get a more accurate and stable prediction. This method reduces the risk of overfitting.
- **Gradient Boosting:** A technique that builds trees sequentially, where each tree tries to correct the errors of the previous one. This approach is effective for improving the accuracy of predictions.

Conclusion

This project demonstrates the potential of integrating AI algorithms into banking applications to provide personalized financial advice. By applying Six Sigma principles and leveraging decision tree algorithms, the application ensures that clients receive timely and relevant financial recommendations. As the application scales, the adoption of more sophisticated algorithms like Random Forest and Gradient Boosting will be necessary to maintain performance and accuracy.