**Banking Specific**

**Acquisition**

**Acquisition** is the phase where the bank focuses on attracting new customers. The goal is to convert potential prospects into active account holders.

**Opportunities in Campaign Analytics:**

* **Segmentation and Targeting:** Identify and target high-potential prospects through segmentation based on demographics, behavior, and other relevant data.

**ML and Analytics Opportunities:**

* **Look-alike Modelling:** Identify potential new customers who resemble existing high-value customers.
* **Churn Prediction: Predict which newly acquired customers are at risk of early churn and proactively engage them with targeted interventions.**

**Engagement**

**Engagement** focuses on nurturing the relationship with the customer by encouraging them to actively use the bank's products and services.

* **Cross-Selling and Upselling:** Identify opportunities to promote additional products (e.g., loans, credit cards) to existing customers based on their usage patterns and needs.
* **Behavioral Analysis:** Monitor customer interactions and transactions to identify engagement levels and tailor communication strategies accordingly.- CLUSTERING

**ML and Analytics Opportunities:**

* **Recommendation Systems:** Develop recommendation engines to suggest relevant products and services to customers based on their transaction history and preferences.
* **Sentiment Analysis:** Analyze customer feedback and sentiment from various channels (e.g., social media, customer service interactions) to understand and address their concerns.

**Retention**

**Retention** is the phase where the bank aims to keep customers loyal and reduce churn. The focus is on maintaining a long-term relationship with the customer.

**ML and Analytics Opportunities:**

* **Churn Prediction:** Use predictive analytics to identify customers who are likely to churn and intervene with personalized retention strategies.
* **Customer Lifetime Value (CLTV) Analysis:** Calculate and track CLTV to focus retention efforts on the most valuable customers.

Segmentation – start

Eg – Student, Hgh profile customers –

Clusters can be a base point – within custumer

Cluster + Alike(Profiling) -> Uplift – how much revenue can be generated

Link -> Cycle + Flow

What level/ stage customer

Top3 opportunites –> which attributes can be used -> target variables

Possible attributes important

### **Integrated Flow**

1. **Acquisition:**
   * **Segmentation and Targeting:** Use K-Means Clustering to segment potential customers based on demographics and behavior.
   * **Look-alike Modelling:** Apply Logistic Regression to identify high-potential prospects similar to existing high-value customers.
2. **Engagement:**
   * **Cross-Selling and Upselling:** Use Decision Trees to identify opportunities for promoting additional products to existing customers.
   * **Recommendation Systems:** Implement Collaborative Filtering to provide personalized product recommendations based on transaction history.
3. **Retention:**
   * **Churn Prediction:** Apply Gradient Boosting to predict the likelihood of customer churn and intervene proactively.
   * **CLTV Analysis:** Use Linear Regression to calculate and track customer lifetime value, focusing retention efforts on high-value customers.

# Segmentation and Targeting in Banking

**Objective:** Segment potential customers into meaningful groups such as students, seniors, high-profile customers, etc., to tailor marketing strategies effectively.

### Key Segmentation Algorithms

1. **K-Means Clustering:**
   * **Description:** A popular and straightforward clustering algorithm that partitions customers into K distinct non-overlapping subgroups.
   * **Use Case:** Effective for identifying broad segments like students, seniors, high-profile customers.
   * **Steps:**
     1. Standardize data (mean=0, variance=1).
     2. Choose the number of clusters K.
     3. Assign each data point to the nearest centroid.
     4. Recompute centroids as the mean of all points in a cluster.
     5. Repeat steps 3-4 until convergence.
   * **Example:**
   * from sklearn.preprocessing import StandardScaler
   * from sklearn.cluster import KMeans
   * scaler = StandardScaler()
   * df\_scaled = scaler.fit\_transform(df)
   * kmeans = KMeans(n\_clusters=5, random\_state=0).fit(df\_scaled)

df['segment'] = kmeans.labels\_

1. **Hierarchical Clustering:**
   * **Description:** Builds a tree of clusters (dendrogram) by progressively merging or splitting clusters.
   * **Use Case:** Useful when the number of clusters is not known a priori and for visualizing the clustering process.
   * **Steps:**
     1. Compute the distance matrix.
     2. Merge the closest pair of clusters.
     3. Repeat until all points are merged into a single cluster.
   * **Example:**
   * from scipy.cluster.hierarchy import dendrogram, linkage
   * import matplotlib.pyplot as plt
   * linked = linkage(df\_scaled, method='ward')
   * dendrogram(linked)

plt.show()

1. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):**
   * **Description:** Identifies clusters based on the density of data points, allowing for the identification of arbitrarily shaped clusters and noise.
   * **Use Case:** Effective for discovering clusters in data with varying densities and for outlier detection.
   * **Steps:**
     1. Choose parameters eps (maximum distance between points in a cluster) and min\_samples (minimum number of points to form a dense region).
     2. Classify points as core, border, or noise.
     3. Expand clusters from core points.
   * **Example:**
   * from sklearn.cluster import DBSCAN
   * dbscan = DBSCAN(eps=0.5, min\_samples=5)

df['segment'] = dbscan.fit\_predict(df\_scaled)

### Important Variables for Segmentation (Indian Banking Perspective)

1. **Demographic Information:**
   * **Age:** Important for segmenting customers into groups like students, working professionals, and retirees.
   * **Income Level:** Helps differentiate between high-profile customers and others.
   * **Location:** Urban vs. rural, regional differences in banking needs.
2. **Behavioral Data:**
   * **Transaction History:** Frequency, volume, and type of transactions.
   * **Product Usage:** Types of banking products used (savings accounts, credit cards, loans).
   * **Channel Preference:** Online banking, mobile app usage, branch visits.
3. **Socio-economic Indicators:**
   * **Occupation:** Differentiates between salaried employees, self-employed individuals, etc.
   * **Education Level:** Can influence financial product preferences and needs.
4. **Credit Information:**
   * **CIBIL Score:** Indicative of creditworthiness, important for segments focused on lending products.
   * **Loan History:** Existing loans, repayment behavior.
5. **Customer Interaction Data:**
   * **Customer Service Interactions:** Frequency and type of inquiries or complaints.
   * **Feedback and Surveys:** Sentiment analysis from customer feedback.

### Detailed Example of Segmentation Implementation

### Step-by-Step Implementation Using K-Means Clustering

1. **Data Preparation:**
   * Collect data from various sources (transaction history, demographic data, credit information).
   * Handle missing values and outliers.
2. **Feature Engineering:**
   * Create relevant features (e.g., average monthly transaction volume, total number of products held).
   * Standardize numerical features to have mean = 0 and variance = 1.
3. **K-Means Clustering Implementation:**
4. import pandas as pd
5. from sklearn.preprocessing import StandardScaler
6. from sklearn.cluster import KMeans
7. # Sample data preparation
8. data = {
9. 'age': [23, 45, 56, 67, 34],
10. 'income': [30000, 80000, 120000, 150000, 50000],
11. 'transaction\_volume': [5000, 15000, 30000, 50000, 12000],
12. 'products\_held': [1, 3, 4, 5, 2],
13. 'cibil\_score': [700, 750, 800, 650, 720]
14. }
15. df = pd.DataFrame(data)
16. # Standardize the data
17. scaler = StandardScaler()
18. df\_scaled = scaler.fit\_transform(df)
19. # Apply K-Means Clustering
20. kmeans = KMeans(n\_clusters=3, random\_state=0).fit(df\_scaled)
21. df['segment'] = kmeans.labels\_

print(df)

1. **Interpretation and Actionable Insights:**
   * Analyze the characteristics of each segment.
   * Develop targeted marketing strategies for each segment (e.g., student loans for the student segment, investment products for high-income segment).

### Variables for Targeting

1. **Student Segment:**
   * Age: Typically 18-25
   * Education: Enrolled in higher education
   * Transaction Patterns: Frequent small transactions, usage of educational loans
2. **Senior Segment:**
   * Age: Above 60
   * Income: Pensioners, fixed income
   * Transaction Patterns: Less frequent transactions, higher focus on savings and fixed deposits
3. **High-Profile Customers:**
   * Income: Above a certain threshold (e.g., INR 1,00,000/month)
   * Occupation: Senior executives, business owners
   * Credit Information: High CIBIL score, high credit limits

By focusing on these key segmentation techniques and relevant variables, banks can effectively segment their customer base and tailor their marketing campaigns to meet the specific needs of each group.

# Look-alike Modelling in Banking

**Objective:** Identify potential new customers who resemble existing high-value customers to target them with tailored marketing campaigns effectively.

### Key Modelling Algorithm

1. **Random Forest:**
   * **Description:** Ensemble learning method that builds multiple decision trees and merges their predictions to improve accuracy.
   * **Use Case:** Effective for classifying potential customers as similar or dissimilar to high-value customers.

### Important Variables for Look-alike Modelling (Indian Banking Perspective)

1. **Profiles of High-value Customers:**
   * **Attributes:** Transaction history, product holdings, creditworthiness, demographic information.
   * **Derived Attributes:** Lifetime value, profitability score, churn propensity.
2. **Transaction History of Potential Customers:**
   * **Attributes:** Transaction frequency, transaction amount, product usage.

### Detailed Example of Look-alike Modelling Implementation

### Step-by-Step Implementation Using Random Forest

1. **Data Preparation:**
   * Utilize existing campaign data (seed data) containing client attributes and campaign-related information.
   * Handle missing values and outliers, preprocess data for modelling.
2. **Feature Engineering:**
   * **Attributes:**
     + Transaction history
     + Product holdings
     + Creditworthiness
     + Demographic information
   * **Derived Attributes:**
     + Lifetime value
     + Profitability score
     + Churn propensity
3. **Model Training and Evaluation:**
4. import pandas as pd
5. from sklearn.model\_selection import train\_test\_split
6. from sklearn.ensemble import RandomForestClassifier
7. from imblearn.over\_sampling import SMOTE
8. from sklearn.metrics import accuracy\_score, f1\_score, roc\_auc\_score, precision\_score
9. # Sample data preparation
10. # Assume df is the preprocessed and sampled data
11. X = df.drop('high\_value\_customer', axis=1)
12. y = df['high\_value\_customer']
13. # Split data into train and test sets
14. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)
15. # Apply SMOTE to handle class imbalance
16. smote = SMOTE(random\_state=42)
17. X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train, y\_train)
18. # Train Random Forest model
19. model = RandomForestClassifier(n\_estimators=100, random\_state=42)
20. model.fit(X\_train\_resampled, y\_train\_resampled)
21. # Predictions on test data
22. y\_pred = model.predict(X\_test)
23. # Evaluation metrics
24. accuracy = accuracy\_score(y\_test, y\_pred)
25. f1 = f1\_score(y\_test, y\_pred)
26. roc\_auc = roc\_auc\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

1. **Interpretation and Actionable Insights:**
   * Analyze model performance using evaluation metrics like Accuracy, F1-Score, ROC-AUC, Precision.
   * Use the trained model to predict potential investors among the remaining client base (pool data).
   * Target the predicted look-alikes with personalized marketing campaigns to increase conversion rates and build a loyal and profitable customer base.

### Variables for Targeting

1. **Look-alike Segment:**
   * **Attributes:** Transaction history, product holdings, creditworthiness, demographic information.
   * **Derived Attributes:** Lifetime value, profitability score, churn propensity.
   * **Target Variables:** Likelihood of becoming a high-value customer, expected lifetime value.

By following this approach, the bank can effectively identify and target potential customers who resemble their high-value customers, leading to improved marketing campaign effectiveness and overall business performance.

# Uplift Modelling in Banking

**Objective:** Identify customers who are likely to be positively influenced by targeted marketing interventions, thereby driving incremental transactions and improving overall campaign effectiveness.

### Key Modelling Algorithm

1. **Uplift Modelling:**
   * **Description:** Predicts the causal effect of a treatment (e.g., a targeted ad) on an individual's behavior. It aims to estimate the difference in outcomes between treated and untreated groups.
   * **Use Case:** Determine which customers are most likely to respond positively to marketing campaigns and tailor interventions accordingly.

### Important Variables for Uplift Modelling (Indian Banking Perspective)

1. **Customer Profile:**
   * **Attributes:** Age, income, occupation, credit score (e.g., CIBIL score).
   * **Derived Attributes:** Risk profile, financial stability, spending patterns.
2. **Behavioral Patterns:**
   * **Attributes:** Purchase history, response to previous marketing campaigns, online activity.
   * **Derived Attributes:** Engagement score, response likelihood, churn propensity.
3. **Campaign Data:**
   * **Attributes:** Type of campaign, channel of communication, offer details, frequency of contact.
   * **Derived Attributes:** Campaign effectiveness score, historical uplift response.

### Detailed Example of Uplift Modelling Implementation

### Step-by-Step Implementation Using Uplift Modelling

1. **Data Preparation:**
   * Collect historical campaign data including both treated (those who received the marketing intervention) and control (those who did not receive the intervention) groups.
   * Preprocess data to handle missing values, encode categorical variables, and normalize numerical features.
2. **Feature Engineering:**
   * **Attributes:**
     + Age
     + Income
     + Occupation
     + Credit score (CIBIL score)
     + Purchase history
     + Response to previous marketing campaigns
     + Online activity
     + Type of campaign
     + Channel of communication
     + Offer details
     + Frequency of contact
   * **Derived Attributes:**
     + Risk profile
     + Financial stability
     + Spending patterns
     + Engagement score
     + Response likelihood
     + Churn propensity
     + Campaign effectiveness score
     + Historical uplift response
3. **Uplift Modelling Implementation:**
4. import pandas as pd
5. from causalml.inference.tree import UpliftRandomForestClassifier
6. from sklearn.model\_selection import train\_test\_split
7. from sklearn.metrics import classification\_report
8. # Sample data preparation
9. data = {
10. 'age': [35, 40, 45, 30, 50],
11. 'income': [60000, 80000, 120000, 50000, 110000],
12. 'occupation': ['salaried', 'business', 'salaried', 'student', 'retired'],
13. 'cibil\_score': [750, 700, 800, 650, 780],
14. 'purchase\_history': [5, 2, 4, 1, 3],
15. 'response\_to\_previous\_campaigns': [1, 0, 1, 0, 1],
16. 'online\_activity': [10, 5, 15, 3, 8],
17. 'campaign\_type': ['email', 'sms', 'email', 'sms', 'email'],
18. 'channel\_of\_communication': ['email', 'sms', 'email', 'sms', 'email'],
19. 'offer\_details': ['discount', 'cashback', 'discount', 'cashback', 'discount'],
20. 'frequency\_of\_contact': [2, 1, 3, 1, 2],
21. 'treatment': [1, 0, 1, 0, 1], # 1 for treated, 0 for control
22. 'response': [1, 0, 1, 0, 1] # 1 for positive response, 0 for no response
23. }
24. df = pd.DataFrame(data)

# Split data into train and

# Split data into train and test sets

X = df.drop(['response'], axis=1)

y = df['response']

# Split 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)

# Train UpliftRandomForestClassifier

uplift\_model = UpliftRandomForestClassifier(n\_estimators=100, control\_name='control')

uplift\_model.fit(X\_train, treatment='treatment', y=y\_train)

# Predict uplift on test set

uplift\_predictions = uplift\_model.predict(X\_test)

# Evaluate the model

print(classification\_report(y\_test, uplift\_predictions))

1. **Interpretation and Actionable Insights:**
   * Analyze uplift scores to identify which customers are likely to be positively influenced by the campaign.
   * Focus marketing efforts on these high-uplift customers to maximize incremental transactions and campaign effectiveness.
   * Adjust marketing strategies based on the model's insights to reduce negative impacts on customers who may be adversely affected by the campaign.

### Variables for Targeting

1. **Uplift Segment:**
   * **Attributes:**
     + Age
     + Income
     + Occupation
     + Credit score (CIBIL score)
     + Purchase history
     + Response to previous marketing campaigns
     + Online activity
     + Type of campaign
     + Channel of communication
     + Offer details
     + Frequency of contact
   * **Derived Attributes:**
     + Risk profile
     + Financial stability
     + Spending patterns
     + Engagement score
     + Response likelihood
     + Churn propensity
     + Campaign effectiveness score
     + Historical uplift response
   * **Target Variables:**
     + Uplift score (likelihood of positive response to the campaign)
     + Incremental transaction likelihood
     + Expected increase in customer lifetime value (CLV)

### Integrating Uplift Modelling with Look-alike Modelling

To further enhance campaign effectiveness, we can combine uplift modelling with look-alike modelling:

1. **Look-alike Modelling:**
   * Identify potential new customers who resemble existing high-value customers.
   * Use attributes like transaction history, product holdings, creditworthiness, and demographic information.
2. **Uplift Modelling:**
   * Determine which of these look-alike customers are likely to respond positively to marketing interventions.
   * Use uplift scores to target high-potential customers with personalized campaigns.

### Implementation Flow

1. **Segmentation and Targeting:**
   * Segment customers based on demographic, behavioral, and transactional data.
   * Identify key segments such as students, high-profile customers, and seniors.
2. **Look-alike Modelling:**
   * Train a model on seed data (existing high-value customers) to find similar customers in the pool data.
   * Use attributes like age, income, occupation, credit score, transaction history, product holdings, and engagement levels.
3. **Uplift Modelling:**
   * Predict the causal effect of marketing interventions on customer behavior.
   * Target customers with high uplift scores to maximize incremental transactions and campaign ROI.
   * Use attributes like age, income, occupation, credit score, purchase history, response to previous campaigns, online activity, campaign type, channel of communication, offer details, and frequency of contact.
4. **Cross-Selling and Upselling:**
   * Implement recommendation systems to suggest relevant products to customers based on their transaction history and preferences.
   * Use attributes like transaction frequency, transaction amount, product usage, average transaction value, product affinity score, product usage trend, engagement score, response likelihood, and churn propensity.

By integrating these methodologies, the bank can create a comprehensive campaign strategy that effectively targets high-potential customers, maximizes positive responses, and optimizes marketing spend.

# Choosing between uplift modeling and look-alike

Choosing between uplift modeling and look-alike modeling depends on the specific objectives of the bank's marketing campaigns and the nature of the customer data. Below is a comparison to help determine which approach might be better suited for different scenarios:

### Look-alike Modeling

**Purpose:** Identify new customers who are similar to existing high-value customers.

**Advantages:**

1. **Customer Acquisition:** Ideal for expanding the customer base by finding new customers who are likely to exhibit similar behaviors and preferences as existing profitable customers.
2. **Simpler Implementation:** Generally easier to implement since it focuses on identifying similarities in customer attributes.
3. **Broad Reach:** Helps in scaling marketing efforts to reach a larger audience that resembles the current best customers.

**Disadvantages:**

1. **No Causal Insight:** Does not provide insights into how marketing interventions affect customer behavior.
2. **Risk of Mis-targeting:** Can result in targeting customers who might not be influenced by the marketing efforts despite their similarities to existing customers.

### Uplift Modeling

**Purpose:** Predict how different marketing interventions will affect individual customer behavior.

**Advantages:**

1. **Campaign Effectiveness:** Focuses on identifying customers who are most likely to be positively influenced by specific marketing actions, leading to more efficient and effective campaigns.
2. **Cost Efficiency:** Reduces marketing spend by targeting only those customers who are likely to respond positively to the intervention.
3. **Causal Insights:** Provides a deeper understanding of the causal impact of marketing actions, helping to refine strategies over time.

**Disadvantages:**

1. **Complex Implementation:** Requires more sophisticated modeling techniques and a thorough understanding of causal inference.
2. **Data Intensive:** Needs detailed data on past campaigns and customer responses to accurately model the uplift effect.

### When to Use Look-alike Modeling

* **Customer Acquisition Goals:** When the primary goal is to expand the customer base by finding new customers who resemble existing high-value customers.
* **Limited Campaign Data:** If there is insufficient data on past campaigns and customer responses, making it difficult to build reliable uplift models.
* **Broad Marketing Strategies:** When the marketing strategy aims to reach a wide audience with similar characteristics to the current customer base.

### When to Use Uplift Modeling

* **Improving Campaign ROI:** When the primary goal is to improve the effectiveness and efficiency of marketing campaigns by targeting customers who are likely to be influenced by specific interventions.
* **Rich Campaign Data:** When there is sufficient historical data on customer responses to past campaigns, allowing for robust uplift modeling.
* **Targeted Marketing Strategies:** When the marketing strategy focuses on personalized and highly targeted campaigns to maximize positive responses and minimize waste.

### Integrated Approach

For optimal results, combining both methodologies can be beneficial:

1. **Initial Expansion with Look-alike Modeling:**
   * Use look-alike modeling to identify a broad pool of potential new customers who resemble existing high-value customers.
2. **Refinement with Uplift Modeling:**
   * Apply uplift modeling to this pool to further refine and target those customers who are most likely to respond positively to specific marketing interventions.

### Example Implementation Flow

1. **Look-alike Modeling:**
   * Identify potential new customers based on similarities to existing high-value customers using attributes like age, income, occupation, credit score, transaction history, and product holdings.
2. **Segmentation and Targeting:**
   * Segment the identified look-alike customers based on key attributes.
3. **Uplift Modeling:**
   * Apply uplift modeling to predict the impact of different marketing interventions on these segments.
   * Focus marketing efforts on high-uplift segments to maximize campaign effectiveness.
4. **Cross-Selling and Upselling:**
   * Use recommendation systems to suggest relevant products to these high-potential customers, enhancing engagement and increasing customer lifetime value.

By combining both approaches, the bank can effectively expand its customer base while ensuring that marketing efforts are focused on those most likely to respond positively, thus optimizing both acquisition and engagement strategies.

# Engagement Stage: Cross-Selling and Upselling

After identifying and targeting potential new customers through look-alike or uplift modeling, the next step is to enhance engagement by focusing on cross-selling and upselling opportunities. This stage aims to deepen the relationship with existing customers by encouraging them to purchase additional products or services.

### Cross-Selling and Upselling

**Objective:**

* Increase the average revenue per customer by recommending relevant additional products or services based on customer needs and behavior.

**Methodology:**

1. **Data Preparation:**
   * Collect and preprocess customer data to ensure it is clean and suitable for analysis.
   * Data sources include transaction history, product holdings, demographic information, and engagement metrics.
2. **Attribute Selection:**
   * **Raw Attributes:**
     + Age
     + Income
     + Occupation
     + Credit score (CIBIL score)
     + Existing product holdings (e.g., types of accounts, loans, credit cards)
     + Transaction history (frequency, recency, monetary value)
     + Customer interaction data (e.g., customer service interactions, website visits)
   * **Derived Attributes:**
     + Product affinity score: Likelihood of interest in a specific product based on past behavior and similar customer profiles.
     + Engagement score: Measure of overall engagement with the bank's services.
     + Transaction trends: Patterns in spending and saving behavior.
     + Churn propensity: Likelihood of customer attrition.
     + Lifetime value (LTV): Projected long-term value of the customer to the bank.
   * **Target Variables:**
     + Response to cross-sell or upsell offers (binary: yes/no)
     + Purchase of additional products (e.g., loans, credit cards)
3. **Model Selection:**
   * **Recommendation System:** Collaborative Filtering, Content-Based Filtering, or Hybrid models.
   * **Machine Learning Algorithms:** Random Forest, Gradient Boosting Machines (GBM), Neural Networks.

### Implementation Steps

1. **Data Collection and Preparation:**
2. import pandas as pd
3. from sklearn.model\_selection import train\_test\_split
4. from sklearn.preprocessing import OneHotEncoder, StandardScaler
5. # Sample customer data
6. data = {
7. 'age': [25, 45, 35, 50, 23],
8. 'income': [50000, 120000, 75000, 100000, 45000],
9. 'cibil\_score': [700, 800, 750, 780, 690],
10. 'existing\_products': ['savings', 'loan', 'credit card', 'savings', 'loan'],
11. 'transaction\_frequency': [15, 50, 25, 30, 10],
12. 'average\_transaction\_value': [2000, 5000, 3000, 4000, 1500],
13. 'engagement\_score': [3, 5, 4, 4, 2],
14. 'response\_to\_cross\_sell': [1, 0, 1, 0, 1]
15. }
16. df = pd.DataFrame(data)
17. # Encode categorical variables
18. encoder = OneHotEncoder()
19. encoded\_products = encoder.fit\_transform(df[['existing\_products']]).toarray()
20. df = df.join(pd.DataFrame(encoded\_products, columns=encoder.get\_feature\_names\_out(['existing\_products']))).drop('existing\_products', axis=1)
21. # Standardize numerical features
22. scaler = StandardScaler()
23. numerical\_features = ['age', 'income', 'cibil\_score', 'transaction\_frequency', 'average\_transaction\_value', 'engagement\_score']

df[numerical\_features] = scaler.fit\_transform(df[numerical\_features])

1. **Model Training and Evaluation:**
2. from sklearn.ensemble import RandomForestClassifier
3. from sklearn.metrics import classification\_report
4. # Split data into features and target variable
5. X = df.drop(['response\_to\_cross\_sell'], axis=1)
6. y = df['response\_to\_cross\_sell']
7. # Train-test split
8. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)
9. # Train Random Forest model
10. model = RandomForestClassifier(n\_estimators=100, random\_state=42)
11. model.fit(X\_train, y\_train)
12. # Predict and evaluate
13. y\_pred = model.predict(X\_test)

print(classification\_report(y\_test, y\_pred))

1. **Recommendation System (Alternative Approach):**

**Collaborative Filtering Example:**

from sklearn.neighbors import NearestNeighbors

import numpy as np

# Sample matrix of customer-product interactions

interaction\_matrix = np.array([

[1, 1, 0, 0, 0],

[1, 0, 1, 0, 0],

[0, 1, 1, 0, 0],

[0, 0, 0, 1, 1],

[1, 0, 0, 1, 0]

])

# Train a KNN model for collaborative filtering

model\_cf = NearestNeighbors(n\_neighbors=2, algorithm='auto')

model\_cf.fit(interaction\_matrix)

# Find similar customers

customer\_index = 0 # Example customer index

distances, indices = model\_cf.kneighbors(interaction\_matrix[customer\_index].reshape(1, -1), n\_neighbors=2)

similar\_customers = indices.flatten()

print(f"Customers similar to customer {customer\_index}: {similar\_customers}")

1. **Deploy and Monitor:**
   * Deploy the trained models to the bank's marketing system.
   * Monitor customer responses to cross-sell and upsell offers.
   * Continuously refine the models based on new data and feedback.

### Variables for Cross-Selling and Upselling

1. **Customer Attributes:**
   * **Age:** Influences financial needs and product preferences.
   * **Income:** Indicates financial capacity and potential product interest.
   * **Occupation:** Provides insights into lifestyle and financial requirements.
   * **Credit Score:** Reflects creditworthiness and financial behavior.
   * **Existing Products:** Shows current product holdings and potential gaps.
2. **Transactional and Behavioral Data:**
   * **Transaction Frequency:** Indicates engagement level with banking services.
   * **Average Transaction Value:** Suggests spending capacity and behavior.
   * **Engagement Score:** Measures overall interaction and activity with the bank.
   * **Response to Previous Campaigns:** Historical data on response rates to past marketing efforts.
3. **Derived Attributes:**
   * **Product Affinity Score:** Calculated based on similarity to other customers who have purchased the product.
   * **Churn Propensity:** Probability of the customer leaving the bank.
   * **Lifetime Value (LTV):** Estimated total value the customer will bring to the bank over time.
4. **Target Variables:**
   * **Response to Cross-Sell/Upsell Offers:** Binary indicator of whether the customer accepted the offer.
   * **Additional Product Purchases:** Details of additional products purchased as a result of cross-selling or upselling efforts.

### Conclusion

By implementing a robust cross-selling and upselling strategy using advanced machine learning techniques, the bank can significantly enhance customer engagement, increase revenue per customer, and build long-term customer relationships. This approach ensures personalized and effective marketing efforts, aligning with customer needs and preferences, ultimately driving growth and customer satisfaction.

# Engagement Stage: Recommendation Systems

After identifying and targeting potential new customers through look-alike or uplift modeling, and enhancing engagement through cross-selling and upselling, the next step is to implement recommendation systems. These systems aim to suggest relevant products and services to customers based on their past behaviors and preferences, further enhancing customer engagement and satisfaction.

### Recommendation Systems

**Objective:**

* Personalize customer experience by recommending products and services that align with their preferences and needs.
* Increase product adoption and customer satisfaction through targeted recommendations.

**Methodology:**

1. **Data Preparation:**
   * Collect and preprocess customer data, including transaction history, product holdings, and interaction data.
   * Data sources include transactional logs, customer profiles, and engagement metrics.
2. **Attribute Selection:**
   * **Raw Attributes:**
     + Customer ID
     + Age
     + Income
     + Occupation
     + Credit score (CIBIL score)
     + Product holdings (e.g., savings account, loan, credit card)
     + Transaction history (frequency, recency, monetary value)
     + Customer interactions (e.g., website visits, customer service interactions)
   * **Derived Attributes:**
     + Product affinity score: Likelihood of interest in specific products based on past behavior and similar customer profiles.
     + Engagement score: Measure of overall engagement with the bank's services.
     + Spending patterns: Trends in spending and saving behavior.
     + Customer lifetime value (CLTV): Projected long-term value of the customer to the bank.
   * **Target Variables:**
     + Recommended product uptake (binary: yes/no)
     + Customer satisfaction score after recommendation (optional)
3. **Model Selection:**
   * **Recommendation System Types:**
     + **Collaborative Filtering:** Based on similarities between users or items.
     + **Content-Based Filtering:** Based on similarities in product attributes.
     + **Hybrid Models:** Combines collaborative and content-based approaches.
   * **Algorithms:** Matrix Factorization, k-Nearest Neighbors (k-NN), Singular Value Decomposition (SVD), Neural Collaborative Filtering.

### Implementation Steps

1. **Data Collection and Preparation:**
2. import pandas as pd
3. from sklearn.preprocessing import OneHotEncoder, StandardScaler
4. # Sample customer data
5. data = {
6. 'customer\_id': [1, 2, 3, 4, 5],
7. 'age': [25, 45, 35, 50, 23],
8. 'income': [50000, 120000, 75000, 100000, 45000],
9. 'cibil\_score': [700, 800, 750, 780, 690],
10. 'product\_holdings': ['savings,loan', 'savings,credit card', 'savings,loan', 'savings', 'loan'],
11. 'transaction\_frequency': [15, 50, 25, 30, 10],
12. 'average\_transaction\_value': [2000, 5000, 3000, 4000, 1500],
13. 'engagement\_score': [3, 5, 4, 4, 2]
14. }
15. df = pd.DataFrame(data)
16. # Encode categorical variables
17. encoder = OneHotEncoder()
18. encoded\_products = encoder.fit\_transform(df[['product\_holdings']]).toarray()
19. df = df.join(pd.DataFrame(encoded\_products, columns=encoder.get\_feature\_names\_out(['product\_holdings']))).drop('product\_holdings', axis=1)
20. # Standardize numerical features
21. scaler = StandardScaler()
22. numerical\_features = ['age', 'income', 'cibil\_score', 'transaction\_frequency', 'average\_transaction\_value', 'engagement\_score']

df[numerical\_features] = scaler.fit\_transform(df[numerical\_features])

1. **Collaborative Filtering Example:**

**Matrix Factorization (using Singular Value Decomposition - SVD):**

from surprise import Dataset, Reader, SVD

from surprise.model\_selection import train\_test\_split, cross\_validate

# Sample interaction data (user\_id, item\_id, rating)

interaction\_data = {

'user\_id': [1, 1, 1, 2, 2, 3, 3, 4, 4, 5],

'item\_id': [101, 102, 103, 101, 103, 102, 104, 101, 104, 103],

'rating': [5, 4, 3, 4, 5, 3, 4, 2, 5, 3]

}

interaction\_df = pd.DataFrame(interaction\_data)

# Convert data into Surprise format

reader = Reader(rating\_scale=(1, 5))

data = Dataset.load\_from\_df(interaction\_df[['user\_id', 'item\_id', 'rating']], reader)

# Train-test split

trainset, testset = train\_test\_split(data, test\_size=0.2)

# Train SVD model

svd = SVD()

svd.fit(trainset)

# Evaluate model

predictions = svd.test(testset)

cross\_validate(svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)

1. **Content-Based Filtering Example:**

**Using TF-IDF for Product Descriptions:**

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import linear\_kernel

# Sample product descriptions

products = {

'product\_id': [101, 102, 103, 104],

'description': [

"Savings account with high interest rates",

"Personal loan with flexible repayment options",

"Credit card with cashback offers",

"Home loan with low interest rates"

]

}

product\_df = pd.DataFrame(products)

# Compute TF-IDF matrix

tfidf = TfidfVectorizer(stop\_words='english')

tfidf\_matrix = tfidf.fit\_transform(product\_df['description'])

# Compute similarity matrix

cosine\_sim = linear\_kernel(tfidf\_matrix, tfidf\_matrix)

# Function to get recommendations based on product description

def get\_recommendations(product\_id, cosine\_sim=cosine\_sim):

idx = product\_df[product\_df['product\_id'] == product\_id].index[0]

sim\_scores = list(enumerate(cosine\_sim[idx]))

sim\_scores = sorted(sim\_scores, key=lambda x: x[1], reverse=True)

sim\_scores = sim\_scores[1:4] # Get top 3 similar products

product\_indices = [i[0] for i in sim\_scores]

return product\_df['product\_id'].iloc[product\_indices]

# Example usage

print(get\_recommendations(101))

1. **Hybrid Recommendation System Example:**

**Combining Collaborative and Content-Based Filtering:**

# Assuming collaborative\_filtering\_predictions and content\_based\_predictions

# are dataframes with customer\_id and recommended product\_id columns

# Merge predictions

hybrid\_recommendations = pd.concat([collaborative\_filtering\_predictions, content\_based\_predictions]).drop\_duplicates()

# Example to show recommendations for a specific customer

customer\_id = 1

customer\_recommendations = hybrid\_recommendations[hybrid\_recommendations['customer\_id'] == customer\_id]

print(customer\_recommendations)

### Variables for Recommendation Systems

1. **Customer Attributes:**
   * **Customer ID:** Unique identifier for each customer.
   * **Age:** Influences financial needs and product preferences.
   * **Income:** Indicates financial capacity and potential product interest.
   * **Occupation:** Provides insights into lifestyle and financial requirements.
   * **Credit Score:** Reflects creditworthiness and financial behavior.
   * **Product Holdings:** Current products owned by the customer.
2. **Transactional and Behavioral Data:**
   * **Transaction Frequency:** Indicates engagement level with banking services.
   * **Average Transaction Value:** Suggests spending capacity and behavior.
   * **Engagement Score:** Measures overall interaction and activity with the bank.
   * **Customer Interactions:** Data from touchpoints like website visits and customer service.
3. **Derived Attributes:**
   * **Product Affinity Score:** Calculated based on similarity to other customers who have purchased the product.
   * **Spending Patterns:** Trends in spending and saving behavior.
   * **Customer Lifetime Value (CLTV):** Estimated total value the customer will bring to the bank over time.
4. **Target Variables:**
   * **Recommended Product Uptake:** Binary indicator of whether the customer accepted the recommendation.
   * **Customer Satisfaction Score:** Optional measure of customer satisfaction post-recommendation.

### Conclusion

By implementing a sophisticated recommendation system, the bank can personalize customer interactions, increase product adoption, and improve customer satisfaction. These systems leverage both collaborative and content-based approaches to provide tailored recommendations, thereby enhancing engagement and fostering long-term customer relationships.

# Retention Stage: Churn Prediction Model

Churn prediction models are crucial in the retention phase to identify customers who are likely to leave the bank. By proactively addressing their needs and concerns, the bank can implement targeted retention strategies to reduce churn and maintain a loyal customer base.

### Churn Prediction Model

**Objective:**

* Predict which customers are likely to churn (leave the bank) and take preemptive actions to retain them.
* Enhance customer loyalty and reduce churn rate by addressing issues before customers decide to leave.

**Methodology:**

1. **Data Preparation:**
   * Collect and preprocess customer data, including demographics, transaction history, product usage, and engagement metrics.
   * Data sources include transactional logs, customer service interactions, and historical churn data.
2. **Attribute Selection:**
   * **Raw Attributes:**
     + Customer ID
     + Age
     + Income
     + Occupation
     + Credit score (CIBIL score)
     + Product holdings (e.g., savings account, loan, credit card)
     + Transaction history (frequency, recency, monetary value)
     + Customer interactions (e.g., website visits, customer service interactions)
     + Complaint logs and resolution times
     + Tenure with the bank
   * **Derived Attributes:**
     + Engagement score: Measure of overall engagement with the bank's services.
     + Customer satisfaction score: Derived from surveys or feedback forms.
     + Churn score: Probability score indicating likelihood of churn.
     + Product utilization rate: Frequency and extent of product usage.
   * **Target Variable:**
     + Churn status (binary: 1 if the customer churned, 0 otherwise)
3. **Model Selection:**
   * **Classification Algorithms:**
     + Logistic Regression
     + Random Forest
     + Gradient Boosting Machines (GBM)
     + Support Vector Machine (SVM)
     + Neural Networks
   * **Evaluation Metrics:**
     + Accuracy
     + Precision
     + Recall
     + F1-Score
     + Area Under the ROC Curve (AUC-ROC)

### Implementation Steps

1. **Data Collection and Preparation:**
2. import pandas as pd
3. from sklearn.preprocessing import StandardScaler, OneHotEncoder
4. from sklearn.model\_selection import train\_test\_split
5. # Sample customer data
6. data = {
7. 'customer\_id': [1, 2, 3, 4, 5],
8. 'age': [25, 45, 35, 50, 23],
9. 'income': [50000, 120000, 75000, 100000, 45000],
10. 'cibil\_score': [700, 800, 750, 780, 690],
11. 'product\_holdings': ['savings,loan', 'savings,credit card', 'savings,loan', 'savings', 'loan'],
12. 'transaction\_frequency': [15, 50, 25, 30, 10],
13. 'average\_transaction\_value': [2000, 5000, 3000, 4000, 1500],
14. 'engagement\_score': [3, 5, 4, 4, 2],
15. 'tenure': [2, 10, 5, 8, 1],
16. 'churn\_status': [0, 1, 0, 0, 1]
17. }
18. df = pd.DataFrame(data)
19. # Encode categorical variables
20. encoder = OneHotEncoder()
21. encoded\_products = encoder.fit\_transform(df[['product\_holdings']]).toarray()
22. df = df.join(pd.DataFrame(encoded\_products, columns=encoder.get\_feature\_names\_out(['product\_holdings']))).drop('product\_holdings', axis=1)
23. # Standardize numerical features
24. scaler = StandardScaler()
25. numerical\_features = ['age', 'income', 'cibil\_score', 'transaction\_frequency', 'average\_transaction\_value', 'engagement\_score', 'tenure']
26. df[numerical\_features] = scaler.fit\_transform(df[numerical\_features])
27. # Split data into train and test sets
28. X = df.drop(['customer\_id', 'churn\_status'], axis=1)
29. y = df['churn\_status']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

1. **Model Training and Evaluation:**

**Random Forest Classifier:**

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score

# Train Random Forest model

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

# Predict on test data

y\_pred = model.predict(X\_test)

y\_prob = model.predict\_proba(X\_test)[:, 1]

# Evaluate model performance

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred)

recall = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

auc\_roc = roc\_auc\_score(y\_test, y\_prob)

print(f'Accuracy: {accuracy}')

print(f'Precision: {precision}')

print(f'Recall: {recall}')

print(f'F1-Score: {f1}')

print(f'ROC AUC: {auc\_roc}')

### Variables for Churn Prediction

1. **Customer Attributes:**
   * **Customer ID:** Unique identifier for each customer.
   * **Age:** Different age groups may have varying churn rates.
   * **Income:** Financial stability may influence customer loyalty.
   * **Occupation:** Reflects lifestyle and financial needs.
   * **Credit Score:** Higher scores may correlate with lower churn rates.
   * **Product Holdings:** Number and type of products owned by the customer.
2. **Transactional and Behavioral Data:**
   * **Transaction Frequency:** Frequency of transactions can indicate engagement.
   * **Average Transaction Value:** Monetary value of transactions.
   * **Engagement Score:** Overall interaction and activity with the bank.
   * **Tenure:** Duration of the customer's relationship with the bank.
   * **Complaint Logs:** History of customer complaints and resolution times.
3. **Derived Attributes:**
   * **Churn Score:** Probability of the customer leaving the bank.
   * **Engagement Score:** Comprehensive measure of customer activity.
   * **Customer Satisfaction Score:** Derived from feedback or survey responses.
   * **Product Utilization Rate:** Frequency and extent of product usage.
4. **Target Variable:**
   * **Churn Status:** Binary indicator of whether the customer has churned (1) or not (0).

### Conclusion

By implementing a churn prediction model, the bank can proactively identify at-risk customers and implement targeted retention strategies. This approach helps in reducing churn rates, maintaining customer loyalty, and enhancing overall customer satisfaction. Using advanced machine learning techniques and a comprehensive set of attributes, the bank can effectively predict and address customer churn, ensuring a stable and loyal customer base.

# Retention Stage: Customer Lifetime Value (CLTV) Prediction Model

Customer Lifetime Value (CLTV) prediction is a critical aspect of customer retention strategy. It helps in identifying the long-term value of a customer to the bank, allowing for strategic decisions on where to invest marketing and retention efforts.

### Customer Lifetime Value (CLTV) Prediction Model

**Objective:**

* Predict the lifetime value of customers to prioritize high-value customers for retention and targeted marketing campaigns.
* Optimize resource allocation by focusing on customers with the highest potential value.

**Methodology:**

1. **Data Preparation:**
   * Collect and preprocess customer data, including transactional, behavioral, and demographic information.
   * Historical transaction data and customer interaction logs are crucial.
2. **Attribute Selection:**
   * **Raw Attributes:**
     + Customer ID
     + Age
     + Income
     + Occupation
     + Credit score (CIBIL score)
     + Product holdings (e.g., savings account, loan, credit card)
     + Transaction history (frequency, recency, monetary value)
     + Customer interactions (e.g., website visits, customer service interactions)
     + Complaint logs and resolution times
     + Tenure with the bank
   * **Derived Attributes:**
     + Average purchase value: Average value of transactions.
     + Purchase frequency: Number of purchases in a given period.
     + Recency: Time since the last purchase.
     + Engagement score: Measure of overall engagement with the bank's services.
     + Customer satisfaction score: Derived from surveys or feedback forms.
     + Churn probability: Probability of the customer leaving the bank.
   * **Target Variable:**
     + CLTV: Predicted monetary value of a customer over their entire relationship with the bank.
3. **Model Selection:**
   * **Regression Algorithms:**
     + Linear Regression
     + Random Forest Regressor
     + Gradient Boosting Regressor
     + XGBoost
     + Neural Networks
   * **Evaluation Metrics:**
     + Mean Absolute Error (MAE)
     + Mean Squared Error (MSE)
     + Root Mean Squared Error (RMSE)
     + R-squared (R²)

### Implementation Steps

1. **Data Collection and Preparation:**
2. import pandas as pd
3. from sklearn.preprocessing import StandardScaler, OneHotEncoder
4. from sklearn.model\_selection import train\_test\_split
5. # Sample customer data
6. data = {
7. 'customer\_id': [1, 2, 3, 4, 5],
8. 'age': [25, 45, 35, 50, 23],
9. 'income': [50000, 120000, 75000, 100000, 45000],
10. 'cibil\_score': [700, 800, 750, 780, 690],
11. 'product\_holdings': ['savings,loan', 'savings,credit card', 'savings,loan', 'savings', 'loan'],
12. 'transaction\_frequency': [15, 50, 25, 30, 10],
13. 'average\_transaction\_value': [2000, 5000, 3000, 4000, 1500],
14. 'engagement\_score': [3, 5, 4, 4, 2],
15. 'tenure': [2, 10, 5, 8, 1],
16. 'cltv': [20000, 120000, 75000, 100000, 45000]
17. }
18. df = pd.DataFrame(data)
19. # Encode categorical variables
20. encoder = OneHotEncoder()
21. encoded\_products = encoder.fit\_transform(df[['product\_holdings']]).toarray()
22. df = df.join(pd.DataFrame(encoded\_products, columns=encoder.get\_feature\_names\_out(['product\_holdings']))).drop('product\_holdings', axis=1)
23. # Standardize numerical features
24. scaler = StandardScaler()
25. numerical\_features = ['age', 'income', 'cibil\_score', 'transaction\_frequency', 'average\_transaction\_value', 'engagement\_score', 'tenure']
26. df[numerical\_features] = scaler.fit\_transform(df[numerical\_features])
27. # Split data into train and test sets
28. X = df.drop(['customer\_id', 'cltv'], axis=1)
29. y = df['cltv']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

1. **Model Training and Evaluation:**

**Gradient Boosting Regressor:**

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

import numpy as np

# Train Gradient Boosting model

model = GradientBoostingRegressor(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

# Predict on test data

y\_pred = model.predict(X\_test)

# Evaluate model performance

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

print(f'Mean Absolute Error: {mae}')

print(f'Mean Squared Error: {mse}')

print(f'Root Mean Squared Error: {rmse}')

print(f'R-squared: {r2}')

### Variables for CLTV Prediction

1. **Customer Attributes:**
   * **Customer ID:** Unique identifier for each customer.
   * **Age:** Different age groups may have varying lifetime values.
   * **Income:** Financial stability can impact spending and saving behaviors.
   * **Occupation:** Reflects lifestyle and financial needs.
   * **Credit Score:** Higher scores may correlate with higher lifetime values.
   * **Product Holdings:** Number and type of products owned by the customer.
2. **Transactional and Behavioral Data:**
   * **Transaction Frequency:** Frequency of transactions can indicate engagement.
   * **Average Transaction Value:** Monetary value of transactions.
   * **Engagement Score:** Overall interaction and activity with the bank.
   * **Tenure:** Duration of the customer's relationship with the bank.
   * **Complaint Logs:** History of customer complaints and resolution times.
3. **Derived Attributes:**
   * **Average Purchase Value:** Average value of transactions.
   * **Purchase Frequency:** Number of purchases in a given period.
   * **Recency:** Time since the last purchase.
   * **Engagement Score:** Comprehensive measure of customer activity.
   * **Customer Satisfaction Score:** Derived from feedback or survey responses.
   * **Churn Probability:** Probability of the customer leaving the bank.
4. **Target Variable:**
   * **CLTV:** Predicted monetary value of a customer over their entire relationship with the bank.

### Conclusion

By implementing a CLTV prediction model, the bank can effectively identify high-value customers and focus its retention and marketing efforts on them. This strategic approach ensures that resources are allocated efficiently, maximizing long-term profitability and enhancing customer satisfaction. Using advanced regression techniques and a comprehensive set of attributes, the bank can accurately predict customer lifetime value and make informed business decisions.