Personal Loan Predictive Model

Machine Learning: Personal Loan Campaign

Isaac Gross - Mar 7, 2025

*Without Stratify=Y when splitting train & test data (ccp_alpha = 0.0)

Contents / Agenda

Executive Summary: Conclusions

- **Model Selection & Performance:**
    - We built three Decision Tree models: Default (Unpruned), Pre-Pruned, and Post-Pruned.
    - The Pre-Pruned tree was chosen as the final model, achieving perfect recall on test data.
- **Key Insights:**
    - Income is the most important predictor, followed by Credit Card Average Spending and Family Size.
    - Customers with higher income and moderate credit card usage are more likely to take loans.
    - Education level and other financial accounts had minimal impact on loan acceptance.
- **Business Impact:**
    - Accurately identifying loan-ready customers can improve marketing efficiency.
    - Minimizing false negatives (high recall) ensures that all potential loan customers are captured.

Executive Summary: Insights & Recommendations

Why Use Recall?

- Since loan acceptance is imbalanced (~9% conversion rate), recall ensures we don't miss potential loan customers.
- High recall (1.0) means we capture every loan-accepting customer, reducing lost revenue opportunities.

Recommendations:

- Target High-Income Customers
- Segment Customers by Higher Spending Habits:
- Optimize Loan Offer Strategy:
  - Offer personalized promotions to families with 2-4 members, as family size has some impact.
- Limit Spending on Non-Impactful Segments:
  - Avoid marketing efforts on these groups to reduce costs.

By leveraging high-recall targeting strategies, we can maximize customer conversions, reduce missed opportunities, and optimize marketing efficiency for greater revenue growth.

Business Problem Overview

- **Problem Statement:**
  - AllLife Bank aims to convert liability customers (depositors) into personal loan customers.
  - The goal is to identify which customers are most likely to accept a personal loan to improve targeted marketing efforts.
- **Key Business Challenges:**
  - Only ~9% of customers accept personal loans, making it hard to identify the right audience.
  - Marketing campaigns must be optimized to increase conversion rates while minimizing costs.
- **Objective:**
  - Develop a predictive model to classify which customers are most likely to accept a loan.
  - Ensure the model has high recall so that no potential loan customers are missed.

Solution Approach

1️⃣ Data Preprocessing

- Cleaned data, fixed anomalies (e.g., negative experience values), and handled categorical variables.
- Identified outliers but found no major impact requiring removal.
- Used a 70/30 train-test split for model evaluation.

2️⃣ Model Building & Selection

- Built Default (Unpruned), Pre-Pruned, and Post-Pruned Decision Trees.
- Final Model: Pre-Pruned Decision Tree (Best recall, avoids overfitting).
- Key Parameters: Max depth = 2, Min samples split = 10, Recall = 1.0 (Perfect on test data).

3️⃣ Evaluation & Business Impact

- Used recall as the key metric to ensure all potential loan customers were identified.
- Insights from feature importance guided targeted marketing strategies for high-income, high-credit-spending customers.

EDA: Univariate Analysis

- **Most of our clients have the following traits:**
  - Income under 100k (~65%)
  - CC Usage under 2k per month
  - No Mortgage (~65%)
  - No Securities Account with us (~90%)
  - No CD Account with us (~94%)
  - No CC from another Bank (~70%)
  - Multi person Family (~70%)
  - Under 50 years old
  - Graduate or Advanced Degree (~58%)

EDA: Multivariate Analysis

- **Found Correlation between the following:**
  - Experience & Age (+99%)
  - Income & CC Average (+65%)
  - Income & Mortgage (+21%)
  - Family & Income (-16%)

EDA: Multivariate Analysis

When comparing the the amount of people who accepted a loan by various variables I found valuable information.

- For anyone who has a CD Account with us, 50% of those people have taken a personal loan with us
- People who take out loans generally have a higher income
  - At minimum ~60k & At max ~200k
  - No outliers
- People who do not take out loans generally have a lower income
  - At max ~155k
  - Some outlier that go past 200k
- A slightly higher % of clients took a loan if they completed more than an undergrad degree
- A small gradual increase % of clients took a loan if they had an additional person in their family
- The following variables had little to no effect on which clients took out a loan:
  - Securities Account, Uses Online Facilities, Use a CC from another Bank, Zip Code, Age, and Experience

Data Preprocessing

- Checked for Anomalous Values
  - Found negative Experience years and updated them with positive values
- Feature Engineering
  - Converted zip code to geographic region categories by taking the first two digits
- Duplicate value check
  - Found 1 duplicate row but had different IDs so we chose to keep it in
- Missing value treatment
  - No Missing Values were found in the data

Data Preprocessing

- Outlier check
    - Found a small percentage of outliers for Income, CC Avg, and Mortgage
    - No treatment needed
- Data preprocessing for modeling
    - Split our data in Train & Test sets
        - 70/30 Split
    - Converted categorical variables (Regional ZipCode & Education) into indicator variables

Model Building

We chose to do a **Classification Decision Tree** machine learning algorithm since this is supervised classification model where we can split the data into a training and test set to evaluate which method is best at identifying a client more willing to take out a loan.

Model Building

⚙️ How Decision Trees Work (Big Picture)

- A decision tree is a model that makes predictions by splitting data into smaller and smaller groups based on questions (conditions) about the features.
- Each "question" creates a branch, and the tree keeps splitting until it reaches a final decision (leaf node), like a class label or value.

Model Building

❓ How Does It Know Where to Split?

- At each step, the algorithm looks at all possible splits for each feature.
- For each possible split, it calculates how good that split is at separating the data.
- The goal is to make each branch as "pure" as possible — meaning each group has mostly one type of outcome (e.g., yes/no, 0/1, red/blue).

Model Building

💡 How Does It Measure the "Goodness" of a Split?

- It uses metrics like:
  - Gini Impurity (common for classification)
  - Entropy/Information Gain (another way to measure purity)
  - Variance Reduction (for regression trees)
- ➡️ Gini Impurity Example:
  - Measures how mixed the classes are in a split.
  - If a node contains only one class, Gini = 0 (pure).
  - Algorithm chooses the split that minimizes Gini impurity.

Model Building

📊 Process Example:

- Imagine a dataset of fruits with features like:
  - Color (Red, Green)
  - Size (Small, Large)
- And you want to predict if it's an Apple or not.
  - First, the tree looks at splitting on Color and sees how well it separates apples from non-apples.
  - Then, it looks at splitting on Size and sees how well that works.
  - Compares both splits.
  - Picks the split that gives the purest separation — say Color.
- Result:
  - Is Color == Red?
  - ├── Yes: Likely Apple
  - └── No: Check Size

Model Building

🚀 In Short:

- Decision trees split data by asking questions.
- They choose splits that best separate the outcomes.
- They use metrics like Gini or Entropy to decide the best split.
- The process repeats until reaching a stopping condition (like pure nodes, max depth).

Model Building

There are 3 types of Decision Trees we built that we looked to evaluate:

- **Default Decision Tree (Unpruned)**→ Good as a baseline but overfits
- **Pre-Pruned Decision Tree**→ Preferred when training time is critical
- **Post-Pruned Decision Tree**→ Best generalization but needs tuning

Model Building

1️⃣Default Decision Tree (Unpruned)

- ✅ Steps to Build:
    - Train a DecisionTreeClassifier without setting depth limits.
    - The tree grows fully until pure leaf nodes are reached.
- 📉 Downsides:
    - Overfits the training data (high variance).
    - Poor generalization to unseen data.
- 📈 Upsides:
    - Captures all patterns in the data (no bias).
    - Can serve as a baseline model.

Model Building

2️⃣ Pre-Pruned Decision Tree (Prevent Overgrowth)

- ✅ Steps to Build:
    - Limit the tree's depth (max_depth), minimum samples per leaf (min_samples_leaf), or impurity gain (min_impurity_decrease).
    - Stops tree growth early to prevent overfitting.
- 📉 Downsides:
    - Risk of underfitting if pruned too aggressively.
- 📈 Upsides:
    - Faster training and better generalization.
    - Avoids capturing noise in data.

Model Building

3️⃣Post-Pruned Decision Tree (After Training)

- ✅ Steps to Build:
  - Train a fully grown tree, then prune back branches using cost-complexity pruning (ccp_alpha).
  - Find the best alpha by cross-validation.
- 📉 Downsides:
  - More computationally expensive than pre-pruning.
- 📈 Upsides:
  - Fine-tunes the model after learning all patterns.
  - Balances bias-variance tradeoff well.

Model Performance Summary

Model Evaluation Criterion

1️⃣**Accuracy** - measures the overall correctness of predictions

❌ Not ideal for imbalanced datasets

❌ **Why NOT use it?** Since most customers don’t take out loans, a model could achieve 90%+ accuracy just by predicting "No" every time, but that wouldn't help identify loan customers

2️⃣**Recall** - focuses on capturing as many actual "yes" cases as possible

✔ Reduces false negatives (missed loan customers)

✅ **Why use it?** Since only a small number of customers take loans, recall ensures we identify as many of them as possible, even if it means some false positives

Model Performance Summary

Model Evaluation Criterion

3️⃣ **Precision** - measures how many predicted "yes" cases are actually correct

✔ Reduces false positives (incorrectly labeling non-loan customers as loan customers)

⚠️ **Why is it less useful here?** Since we already have few loan customers, favoring precision too much might mean we miss many potential loan customers, which is not ideal

4️⃣ **F1-Score** - Harmonic mean of Precision & Recall

✔ Balances false positives and false negatives

⚠️ **Why is it less useful here?** Since it doesn’t fully prioritize finding all loan customers, which is the main goal. Better when both false positives & false negatives matter equally, but here, missing a loan customer (false negative) is worse.

Model Performance Summary

- The Final Decision Tree was Pre-Pruned using the best recall score
- Best parameters found:
  - Max depth: 2
  - Max leaf nodes: 50
  - Min samples split: 10
  - Best test recall score: 1.0

Model Performance Summary

- The most important features used by the decision tree model for prediction
    - Income (87.65% Feature Importance)
    - Credit Card Average (6.69% Feature Importance)
    - Family (5.65% Feature Importance)

Model Performance Summary

Key performance metrics for **training data** of all the models

	Decision Tree (sklearn default)	Decision Tree (Pre-Pruning)	Decision Tree (Post-Pruning)
Accuracy	1.0	0.790286	1.0
Recall	1.0	1.000000	1.0
Precision	1.0	0.310798	1.0
F1	1.0	0.474212	1.0

Key performance metrics for **test data** of all the models

	Decision Tree (sklearn default)	Decision Tree (Pre-Pruning)	Decision Tree (Post-Pruning)
Accuracy	0.986000	0.779333	0.979333
Recall	0.932886	1.000000	0.852349
Precision	0.926667	0.310417	0.933824
F1	0.929766	0.473768	0.891228

Model Performance Improvement

- Performance Trend:
    - **Default Tree** → Highest accuracy & F-1 Score on test data.
    - **Pre-Pruned Tree** → Perfect Recall on test data.
    - **Post-Pruned Tree** → Highest Precision on test data.
- All had equally a perfect Recall on training data.
- Please mention the decision rules and check the feature importance

Model Performance Improvement

- **Default Tree** (No Pruning)
  - Used many features for importance
    - The top were income, family, graduate, advanced degree, CC avg, and age
  - Decision rules started with Income being <= $116,500
    - If true, CC Avg being <= 2.95
    - If false, Family being <= 2.5
  - Grew until all leaf nodes were pure with no limitations
  - Had a depth of 10

Model Performance Improvement

- **Pre-Pruned Tree**
  - Used 3 features for importance
    - They were income, CC avg, and family
  - Decision rules started with Income being <= $92,500
    - If true, CC Avg being <= 2.95
    - If false, Family being <= 2.5
  - Had a few decisions and a depth of 2

Model Performance Improvement

- **Post-Pruned Tree**
  - Used many features for importance
    - The top were income, family, graduate, advanced degree, CC avg, and age
  - Decision rules started with Income being <= $98,500
    - If true, CC Avg being <= 2.95
    - If false, Family being <= 2.5
  - Had many decisions and depth of 14, larger than the unpruned tree
  - No limitations since the ccp_alpha was 0

APPENDIX

Happy Learning !