Personal Loan Approval Prediction Using Decision Trees

Name: Ganesh Cheedarla **Student Id:** 23065307

Github Link: https://github.com/gnanesh1303/Personal-Loan-Approval-Prediction-Using-Decision-Trees

---

Introduction: Why Model Interpretability Matters in Loan Prediction

However, one of these decisions is whether to grant a personal loan, and this kind of decision can be very serious at a personal level. This alone is not enough for a machine learning model to prove its value. There should be something else which speaks for understanding how and why a decision is made.

Many statistical models - for example, Random Forrest, Neural Networks -, may give fairly accurate predictions. However, they are true black boxes; you get a result but, nothing explains it. This is problematic when issues like fairness, accountability, and trust come in especially around finance.

That's why we've decided to use Decision Trees in this project: not only can they're accurate but they're also explainable. Every prediction can literally be followed step by step to see how a final decision is made based on income, expenditures, and education level.

To develop a model that is smart and understandable: makes solid predictions but also can showcase its reason in a manner humans would be able to trust.

Analogy: The Banker and the Loan Decision

Imagine walking into a bank to apply for a personal loan. After reviewing your application, the banker says:
“Your loan has been denied.”

Naturally, you ask: **“Why?”**

A responsible banker might explain:

- Your income is below the required threshold
- You already have a high mortgage
- You don’t have a certificate of deposit (CD) account

Now imagine if the banker just shrugged and said:
“The system flagged you. I don’t know why.”

Would you trust that decision?

That’s exactly how many machine learning models behave - they give you an answer but no explanation. And in something as sensitive as financial approval, **explanations matter**.

That’s why this project uses **Decision Trees** - they act like the responsible banker. They don’t just say yes or no; they show the logic behind the decision. Whether it’s income, credit card spending, or education level, you can follow the reasoning step by step.

This level of **transparency builds trust**, helps reduce bias, and makes AI easier to understand especially in fields where fairness and accountability really count.

Where and Why to Use Decision Trees

Decision Trees are especially valuable when both accuracy and interpretability are needed. They are ideal in scenarios where decisions must be transparent, explainable, and aligned with compliance or ethical standards.

Decision Trees are most useful when you're working with:

- Financial or regulatory industries where decisions must be explained (e.g., banking, credit scoring)
- Human-centered applications where trust and transparency are essential (e.g., healthcare, education)
- Fairness and bias detection tasks where model behavior must be monitored
- Early-stage data exploration when a quick and visual understanding of the data is needed
- Model debugging, to identify which features the model depends on and why

Use Case	How Decision Trees Help
Loan Approval	Clearly explains why a customer was accepted or denied (e.g., low income, no CD account)
Credit Risk Assessment	Identifies high-risk profiles using transparent decision paths
Customer Segmentation	Groups customers based on interpretable feature combinations
Compliance Auditing	Provides justifiable, auditable logic for every decision
Feature Importance Review	Highlights the most influential features in a simple and visual way

---

Limitations and Considerations of Decision Trees

While Decision Trees are great for building models that are easy to understand and explain, they do come with some important limitations especially when you're using them in real-world scenarios like predicting loan approvals.

1. They can overfit easily
If you let a decision tree grow too deep, it might just memorize the training data instead of learning useful patterns. This means it performs well on the training set but poorly on new, unseen data. That’s why finding the right depth is so important. In our case, we found that a depth of around 4 worked best.

2. They don’t explain cause and effect
Just because a tree splits on income or education doesn't mean those things caused the outcome. The model is simply picking up on patterns in the data not making real-world judgments. So, while the explanations are clear, we still have to be careful not to confuse correlation with causation.

3. They can be unstable
Small changes in the data can sometimes lead to very different trees. This makes decision trees a bit fragile, especially when working with smaller or noisy datasets. It also means results can vary more than expected.

4. They can reflect bias in the data
If the training data contains historical biases for example, if certain customer groups were approved for loans more often in the past the model might learn and repeat those patterns. This can lead to unfair or unbalanced decisions, which is something we need to watch out for in any AI system.

5. They don't always capture complex patterns
While decision trees are easy to understand, they're not always the most powerful models. In more complex problems, they might miss subtle relationships between features that other models (like Random Forests or XGBoost) can pick up. The trade-off is between simplicity and performance.

What Makes Decision Trees Different?

Now let's analyze the performance and interpretability of Decision Trees against some of the other popular machine-learning methods. Most models are good at making predictions, but few can explain how they made their decisions; this is where Decision Trees differ.

Method	Description	Global or Local?	Strengths & Limitations
Linear Regression	Uses weighted sums of features to make predictions	Global	Easy to interpret, but assumes linearity
Random Forests	Combines many decision trees to improve accuracy	Global	More accurate, but hard to interpret as a whole
Gradient Boosting (e.g., XGBoost)	Builds trees in sequence to reduce errors	Global	Powerful and accurate, but lacks transparency
Permutation Importance	Measures how much model performance drops if a feature is shuffled	Global	Better for trust, but can be slow to compute
Decision Trees	Makes decisions by splitting data into logical conditions	Both	Easy to explain, fast to train, visually clear

---

Exploratory Data Analysis (EDA): Getting to Know the Data

Before jumping into model building, it's important to take a step back and understand the data we're working with. This is where Exploratory Data Analysis (EDA) comes in — helping us spot trends, detect issues, and uncover useful insights before making any predictions.

For this project, we used a dataset of 5,000 bank customers. Each record includes financial and behavioural details like income, credit card usage, education level, and whether the person accepted a personal loan offer.

The target variable is binary:

- 0 = Loan not accepted
- 1 = Loan accepted

Here's what we discovered during EDA:

- The data was clean, with no missing values.
- People with higher income, better education, or a CD account were more likely to accept the loan.
- The classes (accepted vs not accepted) were fairly balanced, which is good for training.
- Correlation heatmaps and visual plots helped highlight which features stood out most.

Overall, EDA gave us a clearer picture of what to expect — and made sure we weren't going in blind before training our Decision Tree model.

Data Preview

Before building the model, we started by simply **looking at the data** a habit that often reveals more than expected.

Using the head() function, we viewed the **first few rows** to understand the structure: which features we're working with, what the values look like, and whether anything seems off. This gave us a quick snapshot of the dataset before diving deeper.

Then we used describe() to summarize each column. This showed us things like:

- Average income levels and mortgage amounts
- How spread out or skewed the data is
- Whether any features might dominate due to their scale

These first steps helped ground us in the data and gave context to the features before starting the actual modeling work.

Figure 1: Initial rows and descriptive statistics

Top 5 Rows of the Dataset

Weather	Road_Type	Time_of_Day	Traffic_Density	Speed_Limit	Number_of_Vehicles	Driver_Alcohol	Accident_Severity	Road_Condition	Vehicle_Type	Driver_Age	Driver_Experience	Road_Light_Condition	Accident
Rainy	City Road	Morning	1.0	100.0	5.0	0.0	nan	Wet	Car	55.0	46.0	Artificial Light	0.0
Clear	Rural Road	Night	nan	120.0	3.0	0.0	Moderate	Wet	Truck	44.0	40.0	Artificial Light	0.0
Rainy	Highway	Evening	1.0	80.0	4.0	0.0	Low	Icy	Car	56.0	52.0	Artificial Light	0.0
Clear	City Road	Afternoon	2.0	60.0	3.0	0.0	Low	Under Construction	Bus	36.0	31.0	Daylight	0.0
Rainy	Highway	Morning	1.0	105.0	11.0	0.0	Low	Dry	Car	62.0	55.0	Artificial Light	1.0

Descriptive Statistics of Numeric Features

	Mean	Median	Mode	Standard Deviation
Traffic_Density	1.0	1.0	1.0	0.78
Speed_Limit	71.05	60.0	60.0	32.05
Number_of_Vehicles	3.29	3.0	3.0	2.02
Driver_Alcohol	0.16	0.0	0.0	0.37
Driver_Age	43.26	43.0	46.0	15.13
Driver_Experience	38.98	39.0	41.0	15.27
Accident	0.3	0.0	0.0	0.46

Missing Values Check

Before moving on to training the model, it's important to make sure that our dataset is complete. Missing values can disrupt the learning process or lead to biased predictions if not handled properly.

To check for this, we used the isnull().sum() function, which shows how many missing entries (if any) exist in each column.

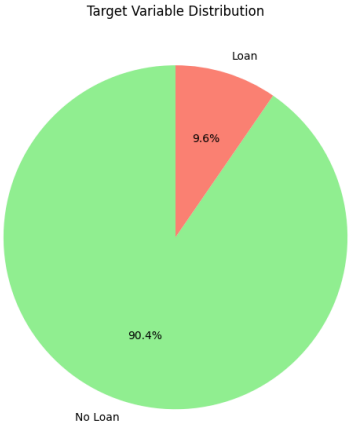
In this case, the dataset came out **clean** there were **no missing values** in any of the features. That means we can move forward with modeling confidently, without needing to fill in gaps or drop rows. It's always a good sign when we don't have to worry about imputation at this early stage

Missing Values Check

Feature	Missing Values
ID	0
Age	0
Experience	0
Income	0
ZIP.Code	0
Family	0
CCAvg	0
Education	0
Mortgage	0
Personal.Loan	0
Securities.Account	0
CD.Account	0
Online	0
CreditCard	0

Class Distribution

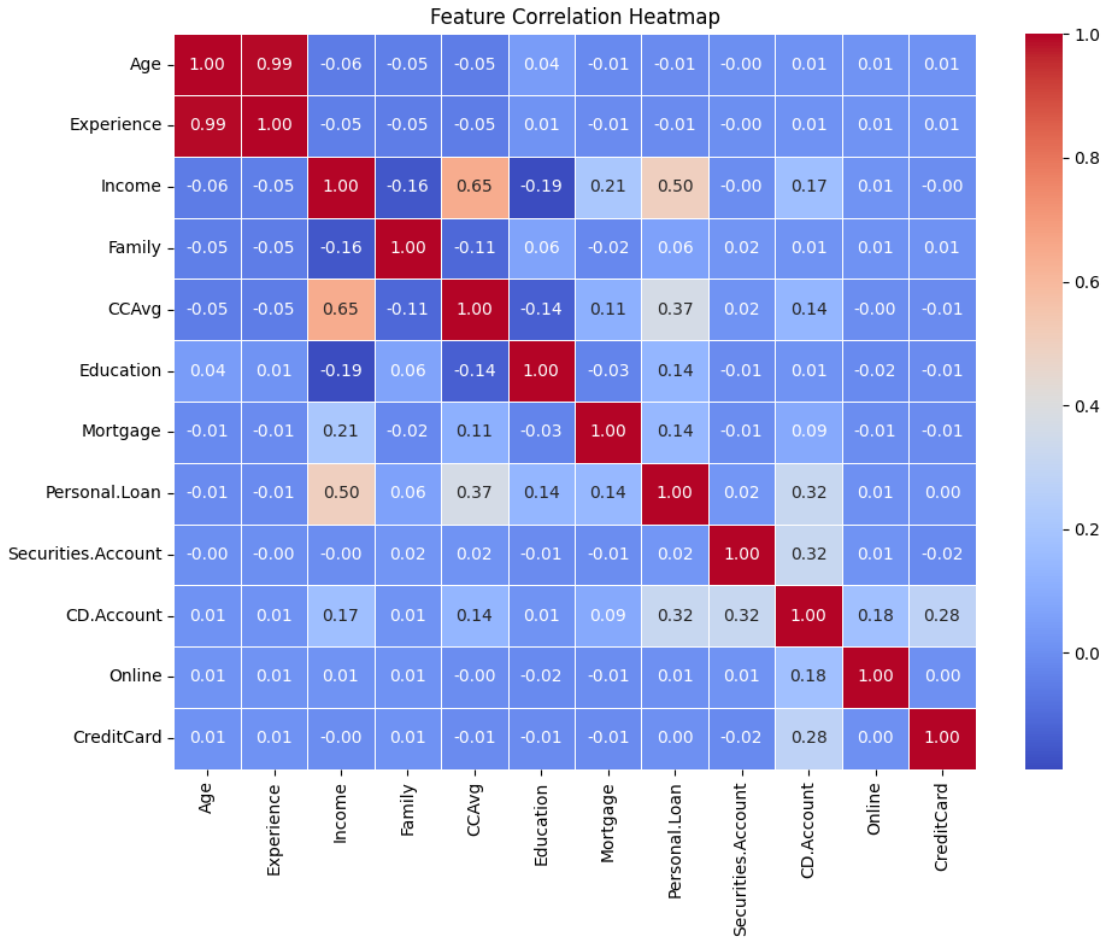
Before training, we first checked how balanced our dataset was in terms of the target variable: whether customers accepted a personal loan (1) or not (0).
Using a simple countplot, we found a **mild imbalance**, with slightly more customers in the "no loan" class.



While this isn't a severe imbalance, it's a reminder to pay close attention to **precision and recall** during evaluation especially when dealing with financial risk.

Feature Correlation Heatmap

Next, we looked at how the features relate to one another by using a **correlation matrix and heatmap**. This helped us identify any strong linear relationships between features.



This step is useful for:

- Spotting **redundant or closely linked features**
- Understanding potential **bias or dominance** in the model
- Supporting later interpretations using feature importance or decision paths

For example, features like Income, CCAvg, and CD Account showed high relevance, which later aligned with our model's results.

Data Preparation

Since Decision Trees **do not require feature scaling**, we skipped standardization. However, we still made sure the data was clean and complete. A quick check confirmed that there were **no missing values**.

Missing Values Check

Feature	Missing Values
ID	0
Age	0
Experience	0
Income	0
ZIP.Code	0
Family	0
CCAvg	0
Education	0
Mortgage	0
Personal.Loan	0
Securities.Account	0
CD.Account	0
Online	0
CreditCard	0

Then, we split the data into:

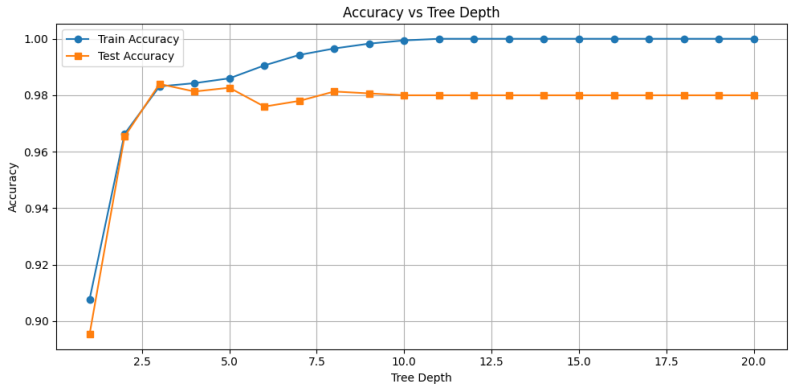
- **70% for training**
- **30% for testing**

We used stratified sampling to maintain the ratio of accepted and rejected loans in both sets.

Model Training and Visual Interpretations

Step 1: Train the Decision Tree

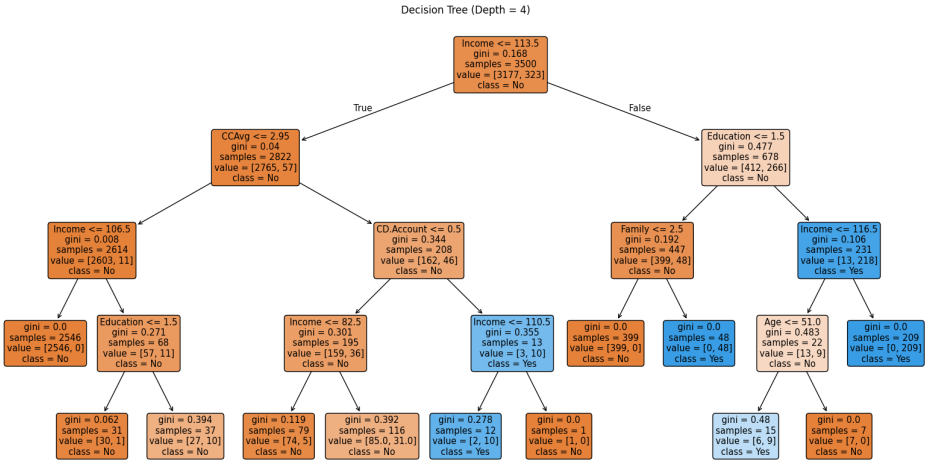
We trained a Decision Tree Classifier using the training data, and we experimented with different tree depths. We plotted accuracy on both training and testing data to avoid overfitting.



We found that **depth = 4** gave the best balance complex enough to capture patterns, but simple enough to generalize well.

Step 2: Visualize the Final Decision Tree

One of the biggest strengths of Decision Trees is that we can **see how decisions are made**. The full tree structure shows exactly which features were used and how thresholds were chosen at each step.

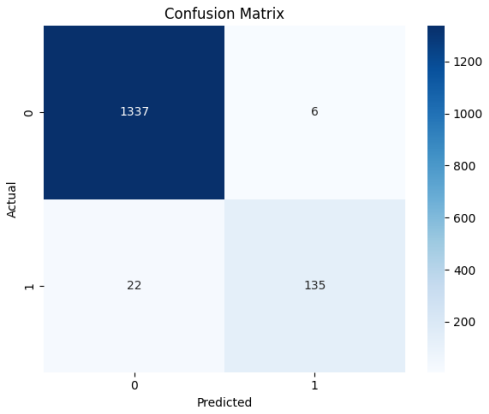


This visual makes the model explainable to non-technical stakeholders a major advantage when working with financial data.

Prediction and Performance Metrics

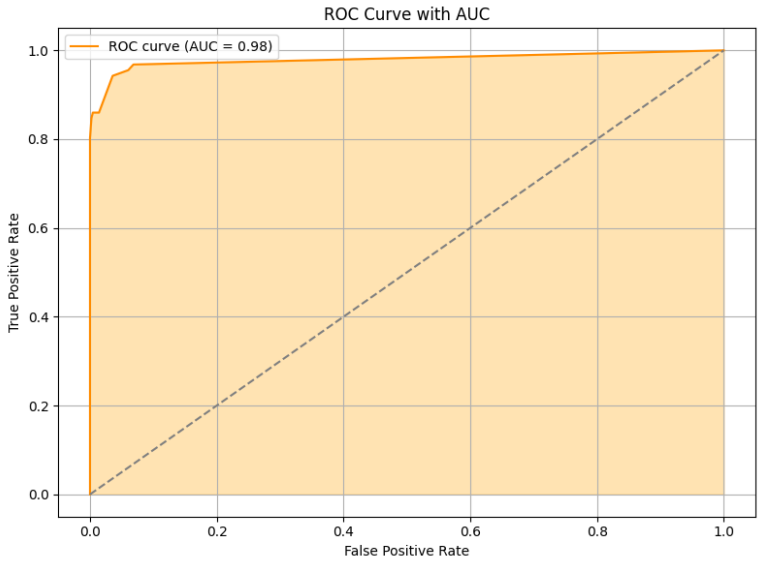
Step 3: Confusion Matrix

To evaluate performance, we started with a confusion matrix. It quickly showed that the model **correctly classified most customers**, with very few false approvals or rejections.



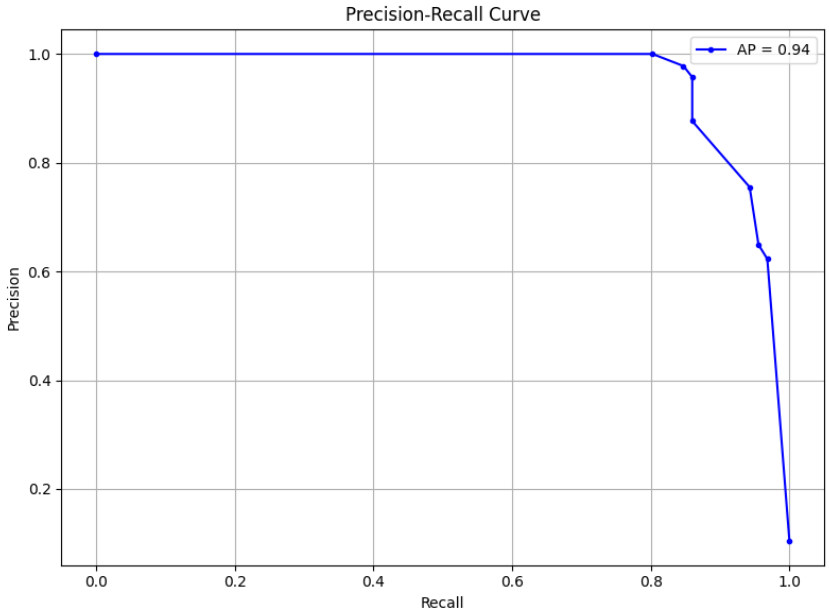
Step 4: ROC Curve

The ROC Curve plots how well the model balances true positives against false positives. Our AUC score was **0.98**, which indicates excellent predictive power.



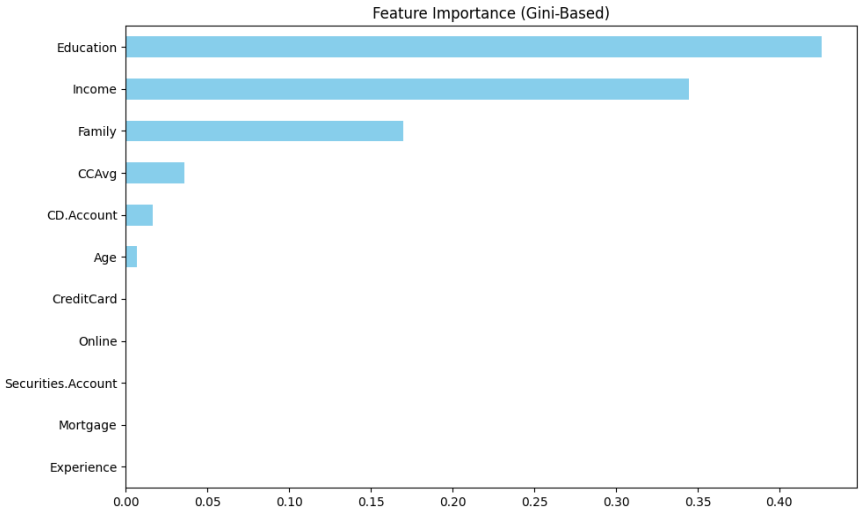
Step 5: Precision-Recall Curve

This curve is especially helpful for understanding how well the model balances **precision** (avoiding false positives) and **recall** (catching true positives). Our average precision score was **0.94** — again, very strong.



Final Notes

While we didn't use SHAP in this project (since Decision Trees are already interpretable), we were still able to explain individual and global feature influence using built-in tools like **feature importance** and **decision paths**.



Conclusion: Building a Model You Can Trust

In this project, we set out to predict whether a customer would accept a personal loan — but more importantly, we wanted to understand **why** the model made each decision.

That's why we used a **Decision Tree**. Unlike many black-box models, a tree gives you a clear and logical explanation. You can trace each prediction step-by-step — whether it's based on income, education, or spending habits.

Along the way, we:

- Explored the data to spot patterns and issues
- Built a clean, interpretable model
- Evaluated its performance using accuracy, precision, recall, and ROC curves
- Visualized the entire decision process

References

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13, 281–305. http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10), 78–87. https://doi.org/10.1145/2347736.2347755

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., ... & Lee, S.-I. (2020). From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence*, 2(1), 56–67.
https://doi.org/10.1038/s42256-019-0138-9

Molnar, C. (2022). *Interpretable Machine Learning* (2nd ed.). Leanpub.
https://christophm.github.io/interpretable-ml-book/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley.

UCI Machine Learning Repository. (n.d.). *Bank Personal Loan Dataset*.
(Adapted dataset used for modeling in this project.)