

Predicting Loan Approval

using Logistic Regression and Decision Trees

with Regularization and Hyperparameter Tuning

Objective:

The goal of this project is to build machine learning models that can predict whether a customer's loan application (or insurance claim) will be approved. The project focuses on comparing two popular algorithms – **Logistic Regression** and **Decision Tree Classifier** – while also applying **regularization techniques** and **hyperparameter optimization** (with GridSearchCV).

Dataset:

You can use a publicly available dataset from Kaggle, for example:

- **Loan Prediction Dataset:** Loan Prediction Problem Dataset on Kaggle
<https://www.kaggle.com/datasets/ninzaami/loan-predication/data>

| Column Name | Meaning | Explanation (Simple) | Example |
|-------------------|-----------------------|--|------------------------|
| Loan_ID | Loan Identifier | Unique ID for each loan application | LP001002 |
| Gender | Applicant's Gender | Whether the applicant is Male or Female | Male, Female |
| Married | Marital Status | Whether the applicant is married or not | Yes, No |
| Dependents | Number of Dependents | Number of people dependent on the applicant (kids, family) | 0, 1, 2, 3+ |
| Education | Education Level | Applicant's education background | Graduate, Not Graduate |
| Self_Employed | Employment Type | Whether the applicant is self-employed or not | Yes, No |
| ApplicantIncome | Applicant's Income | Monthly income of the main applicant | 5000 |
| CoapplicantIncome | Co-applicant's Income | Monthly income of co-applicant (e.g., spouse) | 2000 |
| LoanAmount | Loan Amount | Amount of loan applied (in thousands) | 128 = 128,000 |

| | | | |
|-------------------------|-------------------|---|--------------------------------|
| Loan_Amount_Term | Loan Term | Time period of the loan (in months) | 360 months = 30 years |
| Credit_History | Credit History | Record of applicant's past credit repayment (1 = good, 0 = bad) | 1, 0 |
| Property_Area | Property Location | Area type where the property is located | Urban, Semiurban, Rural |
| Loan_Status | Target Variable | Whether the loan was approved or not | Y = Approved, N = Not Approved |

Notes

- **Loan_ID** is just an identifier → drop it before training.
- **Categorical variables** (Gender, Married, Education, Self_Employed, Property_Area, Loan_Status) → need **encoding** (e.g., OneHotEncoder or LabelEncoder).
- **Dependents** looks numeric, but it's actually categorical because of "3+".
- **Credit_History** is very important (strong predictor).
- **ApplicantIncome + CoapplicantIncome** can be combined into a **TotalIncome** feature.
- **LoanAmount** often has missing values → need to handle them (mean/median imputation or log-transform).
- **Loan_Amount_Term** is usually 360 months (30 years), but some records are shorter/longer.

Real-world interpretation:

- Banks usually approve loans if **income is high, credit history is good, loan amount is reasonable compared to income, and property is in a developed area.**
- This dataset tries to simulate exactly that scenario.

Tasks to Perform:

Break: 11:10 to 11:30

Practical 11:30 to 12:45

1. Data Preprocessing

- Handle missing values.
- Encode categorical features (e.g., Gender, Education).
- Apply feature scaling (especially important for Logistic Regression).
- Split the dataset into training and test sets (70/30).

2. Model Training (without GridSearchCV)

◆ Logistic Regression

- Train Logistic Regression with different penalties: L1, L2.
- Evaluate models using Accuracy, Precision, Recall, and F1 Score.

◆ Decision Tree Classifier

- Train a baseline Decision Tree model using different criteria: **Gini** and **Entropy**.
- Adjust basic parameters: `max_depth`, `min_samples_split`, and `min_samples_leaf`.
- Evaluate performance using the same metrics (Accuracy, Precision, Recall, F1).
- Visualize the tree structure using `plot_tree()` to interpret feature importance and splits.

3. Model Training (with GridSearchCV)

◆ Logistic Regression (Tuning)

- Perform hyperparameter tuning for Logistic Regression: (`C`, `penalty`, `solver`, `l1_ratio`).
- Use GridSearchCV to ensure balanced splits.
- Compare best parameters and scores.

◆ Decision Tree (Tuning)

- Perform hyperparameter tuning for Decision Tree:
 - `Criterion`, `max_depth`, `min_samples_split`, `min_samples_leaf`
- Use GridSearchCV to find the best combination that minimizes overfitting.
- Report the **best parameters**, **best score**, and **feature importances_**.

4. Comparison & Analysis

- Show the effect of regularization and hyperparameter tuning.
- Discuss when Logistic Regression is preferable (interpretable, linear patterns)

- مین افضل مودل من كل التجارب الي جربتھم
- كام تجربة جربت
- وبكل تجربة كام النتيجة **test, train**
- ومین افضلھم بالنسبة الك

Deliverables on Github:

- A Jupyter Notebook with:
 - Data preprocessing steps.
 - Training and evaluation of both models (before and after GridSearchCV).
 - Performance metrics and visualizations (confusion matrix, bar plots of scores).
- A short report (1–2 pages) including:
 - Dataset description.
 - Methodology (Logistic Regression, Decision Tree, Regularization, GridSearchCV).
 - Results and analysis.
 - Conclusion: which model performed better and why.