# LOAN STATUS PREDICTION

To predict loan status with machine learning, we need to train a machine learning model for classifying loan approval or denial. For this, we need a dataset containing information about loan status, so that we can understand what type of transactions lead to fraud. For this task, I collected a dataset from Kaggle, which contains historical information about loan approval and denial status.

# **LOAN STATUS PREDICTION USING PYTHON:**

**Step 1: Setting Up Your Environment**

# Installing Python and Jupyter

1. Install Python: Make sure you have Python installed.

2. Install Jupyter Notebook: we can also work with Jupyter Notebook by running the “pip install notebook” in the command prompt and running by “Jupyter notebook”.

# **Step 2: Installing Necessary Libraries**

# Open the notebook and redirect to the file “ihlp\_project.py”. And import the libraries in a cell.

Software Requirements:

Python3, pandas library , numpy library, scikit-learn library, matplotlib library.

# **Running each cell from the beginning:**

* You can experiment by editing the code in any cell and rerunning it by pressing `Shift + Enter.
* Select the `+ button located in the toolbar to include additional cells for notes or code.
* Recall to frequently save your notebook by either pressing `Ctrl + S` or by clicking the floppy disc icon in the toolbar.

# **Data Preparation:**

Download the loan\_data.csv dataset and place it in your working directory.

# **Code Walkthrough:**

Import Libraries:

The code starts by importing the necessary libraries:

pandas (as pd): for data manipulation numpy (as np): for numerical operations scikit-learn related functions:

accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score, roc\_curve for evaluation metrics

matplotlib.pyplot (as plt) for data visualization.

**Load Dataset:**

The path to the CSV file is defined in data\_path.

The data is loaded using pd.read\_csv(data\_path), storing it in the data

variable.

The first 10 rows are printed using data.head(10) to verify data loading.

The Loan\_ID column is dropped as it's not relevant for prediction.

# **Explore the Dataset:**

The code retrieves the data shape using data.shape and stores it in rows (number of data points) and columns (number of features).

Summary statistics are generated using data.describe() to understand data distribution.

# **Data Cleaning and Preprocessing:**

Check for missing values using data.isnull().sum().

Replace missing values with 0 using data.fillna(0, inplace=True). Encode categorical features using LabelEncoder from scikit-learn.

The Loan\_Status column is converted to numerical labels using label\_encoder.fit\_transform(). A custom function encoder is defined to handle other categorical features:

Converts 'Male' in Gender to 0, 'Female' to 1.

Similar conversions are done for Married, Education, and Self\_Employed. One-hot encoding is applied using pd.get\_dummies().

# **Split Data into Training and Testing Sets:**

Import train\_test\_split from sklearn.model\_selection. Separate features (X) and target variable (y).

Split the data into training and testing sets using train\_test\_split.

The test size is set to 20% (test\_size=0.2) for evaluation, and the random state is set to 42 (random\_state=42) for reproducibility.

Missing values are filled with 0 in X\_train and X\_test.

# **Logistic Regression Model:**

Import LogisticRegression from sklearn.linear\_model. Initialize the model using model = LogisticRegression().

Train the model on the training data using model.fit(X\_train, y\_train).

# **Make Predictions:**

Use the trained model to predict loan status for the testing data with

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

# **Model Evaluation:**

Loss Evaluation (Mean Squared Error - MSE):

Import mean\_squared\_error from sklearn.metrics.

Calculate MSE between actual and predicted values using mean\_squared\_error(y\_test, y\_pred). R-Squared:

Import r2\_score from sklearn.metrics.

Calculate R-Squared using r2\_score(y\_test, y\_pred). R-squared indicates the proportion of variance explained by the model.

# **Model Performance:**

Import various classification metrics from sklearn.metrics:

Accuracy\_score, precision\_score, recall\_score, f1-score , roc\_auc score.

Calculate each metric for the model's performance on the testing data. Print the values of Accuracy, Precision, Recall, F1-score, and ROC

**Stats visualization:**

A list of metrics stores values of Accuracy, precision, recall, f1-score, and roc-auc.

With the plt.bar(metrics, values) we can see the performance metrics of a model.