**NFT ASSIGNMENT-2**

**Objective:** Loan Approval Prediction using Machine Learning

**Problem Statement:**

Dream Housing Finance company deals in all kinds of home loans. They have presence across all urban, semi urban and rural areas. Customer first applies for home loan and after that company validates the customer eligibility for loan.

Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have provided a dataset to identify the customers segments that are eligible for loan amount so that they can specifically target these customers.

**Input Dataset**: <https://drive.google.com/file/d/1LIvIdqdHDFEGnfzIgEh4L6GFirzsE3US/view>

**Output Targeted :**The output is given in 0’s(not approved) or 1’s (approved).  
  
**Data Collection:**Data is collected from website:

<https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset>

| **Variable** | **Description** |
| --- | --- |
| Loan\_ID | Unique Loan ID |
| Gender | Male/ Female |
| Married | Applicant married (Y/N) |
| Dependents | Number of dependents |
| Education | Applicant Education (Graduate/ Under Graduate) |
| Self\_Employed | Self employed (Y/N) |
| ApplicantIncome | Applicant income |
| CoapplicantIncome | Coapplicant income |
| LoanAmount | Loan amount in thousands |
| Loan\_Amount\_Term | Term of loan in months |
| Credit\_History | credit history meets guidelines |
| Property\_Area | Urban/ Semi Urban/ Rural |
| Loan\_Status | Loan approved (Y/N) |

**2. Data Preprocessing:**

**Handling Missing Values**:

Filled missing values in "LoanAmount," "Loan\_Amount\_Term," and "Credit\_History" with their respective means.

df.isnull().sum()

Loan\_ID 0

Gender 13

Married 3

Dependents 15

Education 0

Self\_Employed 32

ApplicantIncome 0

CoapplicantIncome 0

LoanAmount 22

Loan\_Amount\_Term 14

Credit\_History 50

Property\_Area 0

Loan\_Status 0

dtype: int64

df['LoanAmount']=df['LoanAmount'].fillna(df['LoanAmount'].mean())

df['Loan\_Amount\_Term']=df['Loan\_Amount\_Term'].fillna(df['Loan\_Amount\_Term'].mean())

df['Credit\_History']=df['Credit\_History'].fillna(df['Credit\_History'].mean())

df.isnull().sum()

Loan\_ID 0

Gender 13

Married 3

Dependents 15

Education 0

Self\_Employed 32

ApplicantIncome 0

CoapplicantIncome 0

LoanAmount 0

Loan\_Amount\_Term 0

Credit\_History 0

Property\_Area 0

Loan\_Status 0

dtype: int64

Imputed missing values in "Gender," "Married," "Dependents," and "Self\_Employed" with the mode.

df['Gender']=df["Gender"].fillna(df['Gender'].mode()[0])

df['Married']=df["Married"].fillna(df['Married'].mode()[0])

df['Dependents']=df["Dependents"].fillna(df['Dependents'].mode()[0])

df['Self\_Employed']=df["Self\_Employed"].fillna(df['Self\_Employed'].mode()[0])

df.isnull().sum()

Loan\_ID 0

Gender 0

Married 0

Dependents 0

Education 0

Self\_Employed 0

ApplicantIncome 0

CoapplicantIncome 0

LoanAmount 0

Loan\_Amount\_Term 0

Credit\_History 0

Property\_Area 0

Loan\_Status 0

dtype: int64

**Data Transformation:**

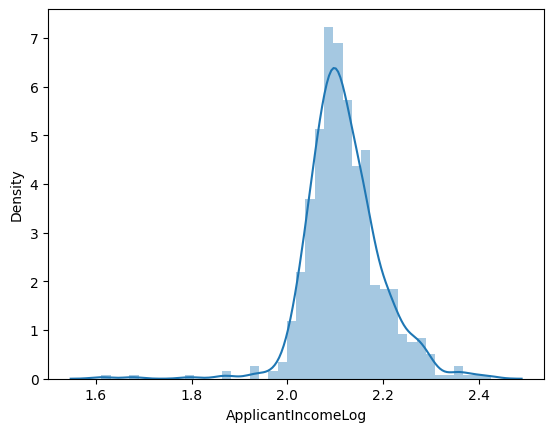
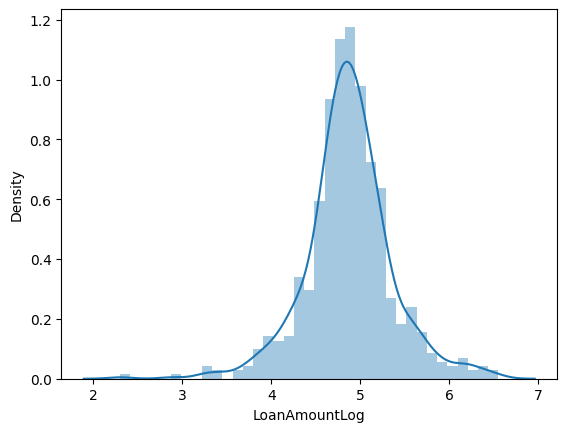
Applied log transformation to "ApplicantIncome," "CoapplicantIncome," "LoanAmount," "Loan\_Amount\_Term," and created corresponding log-transformed columns.

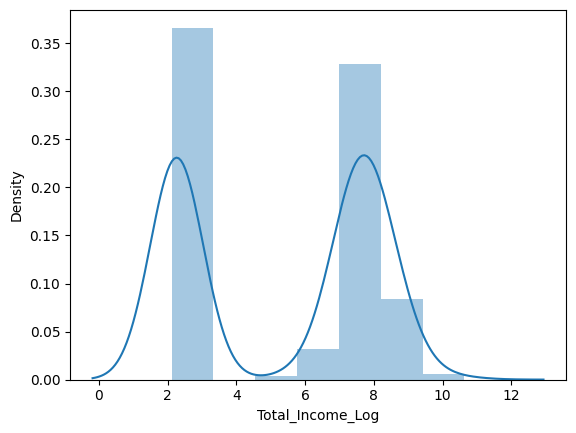
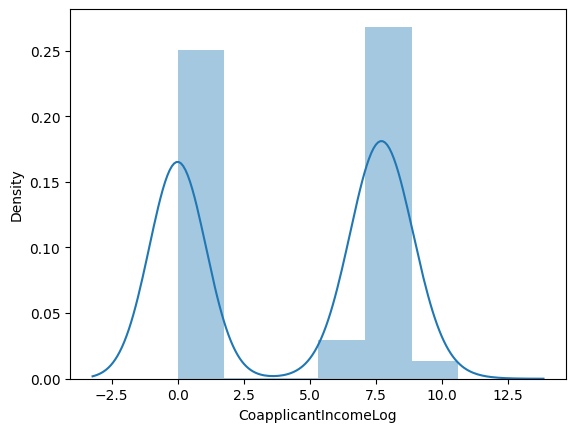
df['ApplicantIncome'] = np.log(df['ApplicantIncome'])

df['CoapplicantIncome'] = np.log(df['CoapplicantIncome'] + 1)  # Adding 1 to avoid log(0)

df['LoanAmount'] = np.log(df['LoanAmount'])

df['Loan\_Amount\_Term'] = np.log(df['Loan\_Amount\_Term'] + 1)  # Adding 1 to avoid log(0)



**3. Data Labeling:**

Encoded categorical variables like Gender, Married, Dependents, Education, Self\_Employed, Property\_Area, Loan\_Status using LabelEncoder.

encoder = LabelEncoder()

categorical\_columns = ['Gender', 'Married', 'Dependents', 'Education', 'Self\_Employed', 'Property\_Area', 'Loan\_Status']

for col in categorical\_columns:

    df[col] = encoder.fit\_transform(df[col])

**4. Data Split:**

Utilized StratifiedShuffleSplit to split the dataset into training and testing sets, maintaining class distribution.

# Data Split

X = df.drop(columns=['Loan\_Status', 'Loan\_ID'], axis=1)

y = df['Loan\_Status']

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

# Standardize the features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

**5.Convolutional Neural Network:**

model = Sequential([

    Conv1D(64, 7, activation='relu', padding='same', input\_shape=(X\_train.shape[1], 1)),

    MaxPooling1D(2),

    Conv1D(128, 5, activation='relu', padding='same'),

    MaxPooling1D(2),

    Conv1D(256, 3, activation='relu', padding='same'),

    MaxPooling1D(2),

    Flatten(),

    Dense(512, activation='relu'),

    Dropout(0.5),

    Dense(256, activation='relu'),

    Dropout(0.5),

    Dense(1, activation='sigmoid')

])

Here, a Sequential model is created, which is a linear stack of layers. The layers defined in this model are:

* Conv1D(64, 7, activation='relu', padding='same', input\_shape=(X\_train.shape[1], 1)): This is a 1D convolutional layer with 64 filters, each of size 7, using the ReLU activation function. It expects input data with shape (X\_train.shape[1], 1).
* MaxPooling1D(2): This is a max-pooling layer with pool size 2. It reduces the spatial dimensions of the output from the previous layer.
* The pattern of Convolutional and MaxPooling layers repeats, gradually increasing the number of filters.
* Flatten(): This layer flattens the output from the previous layer, converting it to a 1D array.
* Dense(512, activation='relu'): A fully connected layer with 512 units and ReLU activation.
* Dropout(0.5): Dropout is a regularization technique to reduce overfitting. It randomly sets a fraction of input units to 0 during training.
* Similar Dense and Dropout layers are repeated.
* Dense(1, activation='sigmoid'): The final layer is a single-unit dense layer with a sigmoid activation function, which is appropriate for binary classification.

**3. Data set being used for the problem statement**<https://drive.google.com/file/d/1LIvIdqdHDFEGnfzIgEh4L6GFirzsE3US/view>

**DONE BY:**

**BT21CSE106-Anandapu Srihitha**

**BT21CSE100-Katnapalli Rishitha**