LOAN RISK ASSESSMENT USING NEURAL NETWORK

Project-1 Submitted in partial fulfilment for the award of

B.Sc. Degree in Computer Science

Madurai Kamaraj University, Madurai.

NAME	**:**	**P.ANBU SELVAN**
ROLL.NO	**:**	**22AUCS018**
CLASS	**:**	**III-B.Sc. COMPUTER SCIENCE**
SUBMITTED ON	**:**	**17/02/2025**
PROJECT GUIDE	**:**	**Dr.T.KATHIRVALAVAKUMAR**

Research Center in Computer Science

V.H.N.SENTHIKUMARA NADAR COLLEGE(Autonomous)
Virudhunagar.

LOAN RISK ASSESSMENT USING NEURAL NETWORK

Project-1 Submitted in partial fulfilment for the award of

B.Sc. Degree in Computer Science

Madurai Kamaraj University, Madurai.

NAME	**:**	**P.ANBU SELVAN**
ROLL.NO	**:**	**22AUCS018**
CLASS	**:**	**III-B.Sc. COMPUTER SCIENCE**
SUBMITTED ON	**:**	**17/02/2025**
PROJECT GUIDE	**:**	**Dr.T.KATHIRVALAVAKUMAR**

Project Guide Signature **HOD Signature**

Research Center in Computer Science

V.H.N.SENTHIKUMARA NADAR COLLEGE(Autonomous)
Virudhunagar.

LOAN RISK ASSESSMENT USING NEURAL NETWORK

OBJECTIVES

Objective of this project is to build a neural network model for binary classification.The task of binary classification is classifying loan approval status based on various features.The model will predict whether an individual will default on the loan.

PROBLEM DESCRIPTION

Develop a model that predicts individuals who will repay the loan amount, based on their previous data.

The project involves:

- Data preprocessing.
- Train a neural network.
- Evaluating the performance of the model based on accuracy.
- Allowing user input for real-time prediction.

PROCEDURE

DATA PREPROCESSING

- **Encoding Variables**: String Variables (e.g., Home, Intent, Default) are converted into numerical values.
- **Normalization**: Features like Age, Income, and Amount are normalized to scale of the data between 0 and 1.

MODEL ARCHITECTURE

- **Neural Network Architecture**: The model consists of three layers:
    1. **Input Layer**: features with a bias term added.
    2. **Hidden Layers**: Two hidden layers with sigmoid activations.
    3. **Output Layer**: A single output node that uses a sigmoid activation to predict a value between 0 and 1.
- **Activation Function**: Sigmoid activation function is used for both hidden layers and the output layer.

  Applied activation function (sigmoid) to the hidden layer:

$$\phi(v) = \frac{1}{1+exp(-net)}$$

- **Backpropagation**: The error is propagated backward from the output layer to the input layer to update the weights.
- **Loss Function**: Mean Squared Error (MSE) is used to calculate the loss during training.

MODEL TRAINING

- **Training and Testing Split**: The dataset is split into training (80%) and testing (20%) subsets.
- **Learning Rate**: A learning rate of 0.01 is used during the training process.
- **Epochs**: The model is trained for 500 epochs to minimize the loss.

LIBRARIES USED

Pandas,NumPy,Random,Math.

ALGORITHM STEPS

Data Collection and Preprocessing

- **Load Dataset**
  - Read data from a CSV file.
- **Encode Categorical Variables**
  - Convert string values (e.g., Home, Intent, Default) into numerical values.
- **Handle Missing Values**
  - Fill any missing values in the dataset with the mean of the respective column.
- **Normalization**
  - Normalize numerical features (e.g., Age, Income, Amount, etc.) to bring them within the scale of 0 to 1.
- **Shuffle and Split Data**
  - Shuffle the dataset to ensure random distribution.
  - Split the data into training (80%) and testing (20%) sets.
- **Initialize Weights**
  - Randomly initialize weights for connections between input, hidden, and output layers.
- **Define Sigmoid Function**
  - Implement the sigmoid activation function and its derivative for use in forward and backward propagation.
- **Forward Propagation**
  - Compute the output of the neural network by propagating inputs through the layers using the sigmoid activation function.

- **Backward Propagation**
    - Calculate errors by comparing predicted output to actual targets.
    - Adjust weights using the backpropagation algorithm to minimize the error.
- **Training**
    - Iterate over the training dataset for a defined number of cycles (epochs) to update weights and minimize the loss.
    - Track the loss for each epoch to monitor the training progress.
- **Model Prediction**
    - Use the trained neural network to make predictions on the test data.
    - Classify outputs as binary values (0 or 1) based on a threshold.
- **Calculate Accuracy**
    - Evaluate the accuracy of the model by comparing the predicted classes with the actual test labels.
    - User Input and Real-Time Prediction
- **Get User Input**
    - Prompt the user to enter various features of a new loan applicant.
    - Encode and normalize the user inputs.
- **Predict Outcome**
    - Use the trained model to predict the loan status for the user-inputted data.
    - Output the predicted loan status.

EXPERIMENT RESULTS

Dataset Overview

Repository: kaggle

- **Dataset : Loan risk prediction dataset**
- **Total Records:** 8,145
- **Target Variable:** Status (0 or 1)
- **Features Used for Prediction**

Table 1: Features details

Feature	Description
Age	The age of the individual applying for the loan.
Income	The annual income of the individual.
Home Ownership	Whether the individual owns, rents, or has a mortgage.
Loan Amount	The requested loan amount.
Employment Length	Number of years the individual has been employed.
Intent	The purpose of the loan (e.g., medical, personal, home improvement).
Rate	The interest rate associated with the loan.
Percent Income	The proportion of income allocated to loan payments.
Credit Length	The number of years the individual has had credit history.
Default	Whether the individual has previously defaulted on a loan.

Table 2: Comparison of model Performance

Case1: Number of hidden layer:1 Learning rate:0.1 Number of Cycles (Epochs):500 Total Neurons:10+10+1 Accuracy:**69.7%**	**Case2:** Number of hidden layer:1 Learning rate:0.01 Number of Cycles (Epochs):500 Total Neurons:10+10+1 Accuracy:**72.5%**
Case3: Number of hidden layer:1 Learning rate:0.01 Number of Cycles (Epochs):1000 Total Neurons:10+10+1 Accuracy:**73.9%**	**Case4:** Number of hidden layer:2 Learning rate:0.01 Number of Cycles (Epochs):500 Total Neurons:10+20+1 Accuracy:**74.2%**
Case5: Number of hidden layer:2 Learning rate:0.1 Number of Cycles (Epochs):1000 Total Neurons:10+20+1 Accuracy:**70.7%**	**Case6:** Number of hidden layer:2 Learning rate:0.001 Number of Cycles (Epochs):500 Total Neurons:10+20+1 Accuracy:**65.4%**
Case7: Number of hidden layer:2 Learning rate:0.01 Number of Cycles (Epochs):500 Total Neurons:10+20+1 Accuracy:**77.8%**	**Case8:** Number of hidden layer:2 Learning rate:0.01 Number of Cycles (Epochs):500 Total Neurons:10+30+1 Accuracy:**84.94%**

Observations

- Increasing the number of neurons and hidden layers generally improved accuracy.
- Lower learning rates (0.01) performed better in most cases than higher learning rates (0.1 or 0.001).
- The best performance was observed in Case 8 with 84.94% accuracy using 2 hidden layers and 10+30+1 neurons.
- Higher cycles (e.g., 1000) did not always guarantee better performance.

CONCLUSION

This project successfully implemented a neural network-based model to predict loan defaulter using financial and demographic features. Through multiple experiments, we analyzed the impact of different hyperparameters, including number of hidden layers, learning rate, number of hidden neurons, and training cycles on model performance.