Assignment 1

Name: Pola Gnana Shekar

roll No: 21CS10052

In [1]:

```
# import all the necessary libraries here
import pandas as pd

df = pd.read_csv('../../dataset/cross-validation.csv')
print(df.shape)
```

```
(614, 13)
```

In [4]:

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load the dataset
data = pd.read_csv("../../dataset/cross-validation.csv")

# Divide the data into features (X) and target (y)
X = data.drop("Loan_Status", axis=1)
y = data["Loan_Status"]

# Split the data into 80% train and 20% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

# Print the sizes of the split datasets
print("Train set size:", X_train.shape[0])
print("Test set size:", X_test.shape[0])
```

```
Train set size: 491
Test set size: 123
```

In [6]:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
import warnings
warnings.filterwarnings("ignore", category=UserWarning)

# Load the dataset
data = pd.read_csv("../../dataset/cross-validation.csv")

# Drop the Loan_ID column as it's not needed for modeling
data = data.drop("Loan_ID", axis=1)

# Convert Dependents column to numeric values
data['Dependents'] = data['Dependents'].replace('3+', 3).astype(float)

# Encode categorical variables using one-hot encoding
data = pd.get_dummies(data, columns=['Gender', 'Married', 'Education', 'Self_Employed',

# Handle missing values (for simplicity, let's fill missing values with column means)
columns_with_missing_values = ['Dependents', 'LoanAmount', 'Loan_Amount_Term', 'Credit_H
for column in columns_with_missing_values:
    data[column].fillna(data[column].mean(), inplace=True)

# Split the data into features (X) and target (y)
X = data.drop("Loan_Status", axis=1)
y = (data["Loan_Status"] == 'Y').astype(int)  # Convert 'Y' to 1 and 'N' to 0


# Split the data into 80% train and 20% test
train_size = int(0.8 * len(data))
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

# Normalize the data
X_train = (X_train - X_train.mean()) / X_train.std()
X_test = (X_test - X_train.mean()) / X_train.std()  # Use training mean and std for norm

# Train the Logistic Regression model using scikit-learn's LogisticRegression
def logistic_regression_saga(X, y, max_iter):
    model = LogisticRegression(penalty='none', solver='saga', max_iter=max_iter, fit_int
    model.fit(X, y)
    weights = model.coef_[0]
    return weights

# Train the Logistic Regression model using the saga solver
learning_rate = 0.1
num_epochs = 1000
weights = logistic_regression_saga(X_train, y_train, max_iter=num_epochs)

# Cross Validation logic using scikit-learn's LogisticRegression
k = 5
fold_size = len(X_train) // k
accuracies = []
precisions = []
recalls = []

for i in range(k):
    start = i * fold_size
    end = (i + 1) * fold_size if i < k - 1 else len(X_train)
```

```
    X_val_fold = X_train[start:end]
    y_val_fold = y_train[start:end]

    X_train_fold = np.concatenate((X_train[:start], X_train[end:]))
    y_train_fold = np.concatenate((y_train[:start], y_train[end:]))

    weights_fold = logistic_regression_saga(X_train_fold, y_train_fold, max_iter=num_epo

    model_fold = LogisticRegression(penalty='none', solver='saga', max_iter=num_epochs)
    model_fold.fit(X_train_fold, y_train_fold)

    y_pred_fold = model_fold.predict(X_val_fold)

    accuracy_fold = np.mean(y_pred_fold == y_val_fold)
    precision_fold = np.sum(y_pred_fold * y_val_fold) / np.sum(y_pred_fold)
    recall_fold = np.sum(y_pred_fold * y_val_fold) / np.sum(y_val_fold)

    accuracies.append(accuracy_fold)
    precisions.append(precision_fold)
    recalls.append(recall_fold)

mean_accuracy = np.mean(accuracies)
mean_precision = np.mean(precisions)
mean_recall = np.mean(recalls)

print("Mean Accuracy:", mean_accuracy)
print("Mean Precision:", mean_precision)
print("Mean Recall:", mean_recall)
```

```
Mean Accuracy: 0.7962481962481962
Mean Precision: 0.7875964475473237
Mean Recall: 0.9651490045441993
```