Loan prediction

'pandas' is used for data manipulation and analysis. 'train_test_split' is used to split the dataset into training and testing sets. *'StandardScaler' is used for standardization of numerical features.* 'OneHotEncoder' is used for one-hot encoding categorical features. *'ColumnTransformer' is used to apply different preprocessing to different columns.* 'Pipeline' is used to streamline a lot of the routine processes in machine learning workflows. *'SimpleImputer' is used to handle missing values in the dataset.* 'LogisticRegression' is the logistic regression model used for classification. *'accuracy_score', 'classification_report', and 'confusion_matrix' are used for model evaluation.

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [2]:
```
df = pd.read_csv('loan-predictionUC.csv')
```

The line is using the read_csv function from the pandas library to read a CSV file named 'loan_predictionUC.csv' and store it in a DataFrame called df.

In [3]:
```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\mythili\appdata\local\programs\python\python311\lib\sit
e-packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\users\mythili\appdata\local\programs\python\python311\l
ib\site-packages (from scikit-learn) (1.24.0)
Requirement already satisfied: scipy>=1.5.0 in c:\users\mythili\appdata\local\programs\python\python311\lib\sit
e-packages (from scikit-learn) (1.9.3)
Requirement already satisfied: joblib>=1.1.1 in c:\users\mythili\appdata\local\programs\python\python311\lib\si
te-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\mythili\appdata\local\programs\python\python311
\lib\site-packages (from scikit-learn) (3.2.0)
```

installation of scikit-learn library

In [4]:
```
import sklearn as sklearn
```

Importing the scikit-learn library to our program

In [5]:
```
df = df.drop(['Loan_ID'], axis=1)
```

This line of code drops the 'Loan_id' column along the columns axis (axis=1) and assigns the modified DataFrame back to the variable df.

In [6]:
```
X = df.drop('Loan_Status', axis=1)
y = df['Loan_Status']
```

create a new DataFrame X that contains all the columns of the original DataFrame df except for the 'Loan_Status' column. This is commonly done when you want to separate the features (independent variables) from the target variable.

The y = df['Loan_Status'] line is used to create a Series y containing only the 'Loan_status' column, which typically represents the target variable or the variable you want to predict.

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

These sets are used for training and evaluating machine learning models. The training set is used to train the model, and the testing set is used to assess the model's performance on unseen data.

In [8]:
```
numeric_features = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term']
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])
```

numeric_features: This is a list of column names representing the numeric features in your dataset. In this case, it includes 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount', and 'Loan_Amount_Term'.

numeric_transformer: This is a scikit-learn Pipeline that defines a sequence of transformations to be applied to the numeric features. The transformations include:

Imputation (SimpleImputer): This step fills in missing values using the median of each column. The SimpleImputer class is used with the strategy='median' parameter.

Scaling (StandardScaler): This step standardizes the features by removing the mean and scaling to unit variance. The StandardScaler scales the features to have a mean of 0 and a standard deviation of 1.

This pipeline is useful when you want to apply the same sequence of transformations to multiple sets of data (e.g., training and testing sets) consistently.

In [9]:
```
categorical_features = ['Gender', 'Married', 'Education', 'Self_Employed', 'Property_Area', 'Credit_History', 'I
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
```

categorical_features: This is a list of column names representing the categorical features in your dataset. In this case, it includes 'Gender', 'Married', 'Education', 'Self-employed', 'Property_area', 'Credit_history', and 'Dependents'.

categorical_transformer: This is a scikit-learn Pipeline that defines a sequence of transformations to be applied to the categorical features. The transformations include:

Imputation (SimpleImputer): This step fills in missing values using the most frequent value (mode) of each column. The SimpleImputer class is used with the strategy='most_frequent' parameter.

One-Hot Encoding (OneHotEncoder): This step converts categorical variables into binary vectors, commonly known as one-hot encoding. Each category becomes a binary column, and the presence or absence of the category is indicated by 1 or 0, respectively. The OneHotEncoder class is used with the handle_unknown='ignore' parameter, which allows the encoder to ignore unseen categories during transformation.

This pipeline is designed to handle missing values and convert categorical features into a format suitable for machine learning algorithms.

In [10]:
```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])
```

umeric_transformer: This is the pipeline for preprocessing numeric features that you defined earlier.

categorical_transformer: This is the pipeline for preprocessing categorical features that you defined earlier.

numeric_features: This is a list of column names representing the numeric features in your dataset.

categorical_features: This is a list of column names representing the categorical features in your dataset.

The ColumnTransformer is a powerful tool in scikit-learn that allows you to apply different transformations to different subsets of columns in your dataset. In this case, it applies the numeric_transformer to the numeric features and the categorical_transformer to the categorical features.

In [11]:
```
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])
```

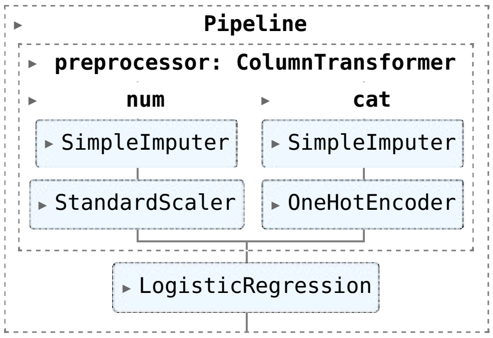
preprocessor: This is the ColumnTransformer you defined earlier, which contains the preprocessing steps for both numeric and categorical features. It handles the transformation of input data before it is fed into the model. The preprocessor is assigned the name 'preprocessor' within the pipeline.

classifier: This is the machine learning model you want to train. In this case, it's a logistic regression classifier (LogisticRegression()). The classifier is assigned the name 'classifier' within the pipeline.

Pipeline: The Pipeline class is used to sequentially apply a list of transforms and a final estimator. The transforms are applied in the order they appear in the steps list, and the final estimator is trained on the transformed data.

In [12]:
```
model.fit(X_train, y_train)
```

Out[12]:



The code responsible for training our machine learning model using the training data.

In [13]:
```
y_pred = model.predict(X_test)
```

For making predictions using the trained machine learning model on a set of test data

In [14]:
```
print(f'Accuracy: {accuracy_score(y_test, y_pred)}')
print(f'Classification Report:\n{classification_report(y_test, y_pred)}')
print(f'Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}')
```

```
Accuracy: 0.7886178861788617
Classification Report:
              precision    recall  f1-score   support

           N       0.95      0.42      0.58        43
           Y       0.76      0.99      0.86        80

    accuracy                           0.79       123
   macro avg       0.85      0.70      0.72       123
weighted avg       0.83      0.79      0.76       123

Confusion Matrix:
[[18 25]
 [ 1 79]]
```

accuracy_score(y_test, y_pred): This function from scikit-learn computes the accuracy of the predicted labels (y_pred) compared to the true labels (y_test). It is calculated as the ratio of correctly predicted instances to the total number of instances in the test set.The result is printed as a string indicating the accuracy.

classification_report(y_test, y_pred): This function generates a text report showing the main classification metrics. It includes precision, recall, F1-score, and support for each class (in this case, each class of the target variable 'Loan_status').The result is printed as a formatted string.

confusion_matrix(y_test, y_pred): This function creates a confusion matrix, which is a table that shows the number of true positive, true negative, false positive, and false negative predictions. The result is printed as a formatted string.

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js