Experiment 3: Data Preparation Process in Python for Financial/Banking/Insurance/Credit Dataset

1. Objectives

The objective of this experiment is to teach the fundamental steps involved in preparing data for machine learning models and visualization.

We aim to: ● Objective 1: Understand the importance of data cleaning (removing duplicates, missing data handling, etc.). ● Objective 2: Apply data normalization to bring features on the same scale. ● Objective 3: Perform feature selection to identify relevant features for model building. ● Objective 4: Implement imputation techniques for missing values. ● Objective 5: Learn how to use label encoding and one-hot encoding for categorical variables. ● Objective 6: Create a train-test split for machine learning tasks.

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [100…
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

In [3]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [4]:
```
from google.colab import files

uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving loan-data.csv to loan-data.csv
```

In [ ]:

In [114…
```
from google.colab import files

uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving credit_risk_dataset.csv to credit_risk_dataset.csv
```

In [6]:
```
df = pd.read_csv('loan-data.csv')
```

In [115…
```
df = pd.read_csv('credit_risk_dataset.csv')
```

In [10]:
```
# 1. Structure of the dataset
print("Shape of dataset:", df.shape)
```

```
Shape of dataset: (614, 13)
```

In [116…
```
print("Columns:", df.columns)
```

```
Columns: Index(['person_age', 'person_income', 'person_home_ownership',
       'person_emp_length', 'loan_intent', 'loan_grade', 'loan_amnt',
       'loan_int_rate', 'loan_status', 'loan_percent_income',
       'cb_person_default_on_file', 'cb_person_cred_hist_length'],
      dtype='object')
```

In [117…
```
print("\nData Types:\n", df.dtypes)
```

```
Data Types:
 person_age                      int64
person_income                   int64
person_home_ownership          object
person_emp_length             float64
loan_intent                    object
loan_grade                     object
loan_amnt                       int64
loan_int_rate                 float64
loan_status                     int64
loan_percent_income           float64
cb_person_default_on_file      object
cb_person_cred_hist_length      int64
dtype: object
```

In [118…
```
# 2. Statistical Summary
print("\nStatistical Summary (Numerical Columns):\n", df.describe())
print("\nStatistical Summary (Categorical Columns):\n", df.describe(include='object
```

```
Statistical Summary (Numerical Columns):
         person_age  person_income  person_emp_length     loan_amnt  \
count  32581.000000   3.258100e+04       31686.000000  32581.000000
mean      27.734600   6.607485e+04           4.789686   9589.371106
std        6.348078   6.198312e+04           4.142630   6322.086646
min       20.000000   4.000000e+03           0.000000    500.000000
25%       23.000000   3.850000e+04           2.000000   5000.000000
50%       26.000000   5.500000e+04           4.000000   8000.000000
75%       30.000000   7.920000e+04           7.000000  12200.000000
max      144.000000   6.000000e+06         123.000000  35000.000000

       loan_int_rate   loan_status  loan_percent_income  \
count   29465.000000  32581.000000         32581.000000
mean       11.011695      0.218164             0.170203
std         3.240459      0.413006             0.106782
min         5.420000      0.000000             0.000000
25%         7.900000      0.000000             0.090000
50%        10.990000      0.000000             0.150000
75%        13.470000      0.000000             0.230000
max        23.220000      1.000000             0.830000

       cb_person_cred_hist_length
count                32581.000000
mean                     5.804211
std                      4.055001
min                      2.000000
25%                      3.000000
50%                      4.000000
75%                      8.000000
max                     30.000000

Statistical Summary (Categorical Columns):
       person_home_ownership loan_intent loan_grade cb_person_default_on_file
count                  32581       32581      32581                     32581
unique                     4           6          7                         2
top                     RENT   EDUCATION          A                         N
freq                   16446        6453      10777                     26836
```

In [12]:
```
df.drop_duplicates(inplace=True)
print(df)
```

```
      Loan_ID  Gender Married Dependents     Education Self_Employed  \
0    LP001002    Male      No          0      Graduate            No
1    LP001003    Male     Yes          1      Graduate            No
2    LP001005    Male     Yes          0      Graduate           Yes
3    LP001006    Male     Yes          0  Not Graduate            No
4    LP001008    Male      No          0      Graduate            No
..        ...     ...     ...        ...           ...           ...
609  LP002978  Female      No          0      Graduate            No
610  LP002979    Male     Yes         3+      Graduate            No
611  LP002983    Male     Yes          1      Graduate            No
612  LP002984    Male     Yes          2      Graduate            No
613  LP002990  Female      No          0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0               5849                0.0         NaN             360.0
1               4583             1508.0       128.0             360.0
2               3000                0.0        66.0             360.0
3               2583             2358.0       120.0             360.0
4               6000                0.0       141.0             360.0
..               ...                ...         ...               ...
609             2900                0.0        71.0             360.0
610             4106                0.0        40.0             180.0
611             8072              240.0       253.0             360.0
612             7583                0.0       187.0             360.0
613             4583                0.0       133.0             360.0

     Credit_History Property_Area Loan_Status
0               1.0         Urban           Y
1               1.0         Rural           N
2               1.0         Urban           Y
3               1.0         Urban           Y
4               1.0         Urban           Y
..              ...           ...         ...
609             1.0         Rural           Y
610             1.0         Rural           Y
611             1.0         Urban           Y
612             1.0         Urban           Y
613             0.0     Semiurban           N

[614 rows x 13 columns]
```

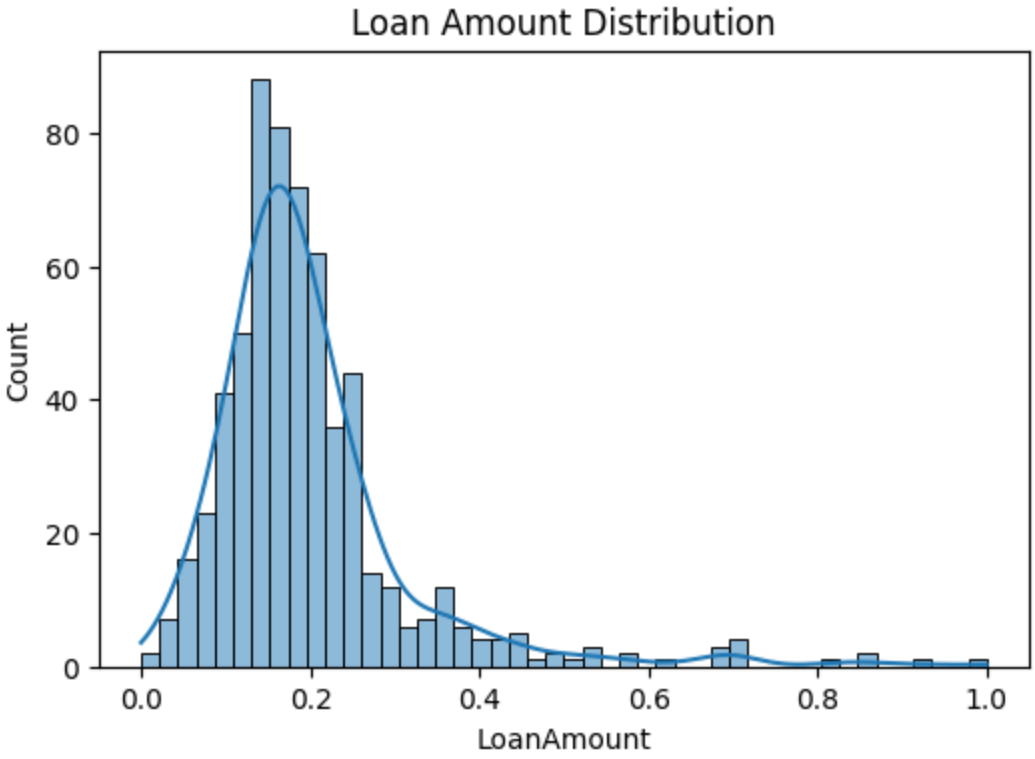
In [16]:
```
# Imputation: Fill missing numerical values with mean
df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)
print(df)
```

```
      Loan_ID  Gender Married Dependents     Education Self_Employed  \
0    LP001002    Male      No          0      Graduate            No
1    LP001003    Male     Yes          1      Graduate            No
2    LP001005    Male     Yes          0      Graduate           Yes
3    LP001006    Male     Yes          0  Not Graduate            No
4    LP001008    Male      No          0      Graduate            No
..        ...     ...     ...        ...           ...           ...
609  LP002978  Female      No          0      Graduate            No
610  LP002979    Male     Yes         3+      Graduate            No
611  LP002983    Male     Yes          1      Graduate            No
612  LP002984    Male     Yes          2      Graduate            No
613  LP002990  Female      No          0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0               5849                0.0  146.412162             360.0
1               4583             1508.0  128.000000             360.0
2               3000                0.0   66.000000             360.0
3               2583             2358.0  120.000000             360.0
4               6000                0.0  141.000000             360.0
..               ...                ...         ...               ...
609             2900                0.0   71.000000             360.0
610             4106                0.0   40.000000             180.0
611             8072              240.0  253.000000             360.0
612             7583                0.0  187.000000             360.0
613             4583                0.0  133.000000             360.0

     Credit_History Property_Area Loan_Status
0               1.0         Urban           Y
1               1.0         Rural           N
2               1.0         Urban           Y
3               1.0         Urban           Y
4               1.0         Urban           Y
..              ...           ...         ...
609             1.0         Rural           Y
610             1.0         Rural           Y
611             1.0         Urban           Y
612             1.0         Urban           Y
613             0.0     Semiurban           N

[614 rows x 13 columns]
```

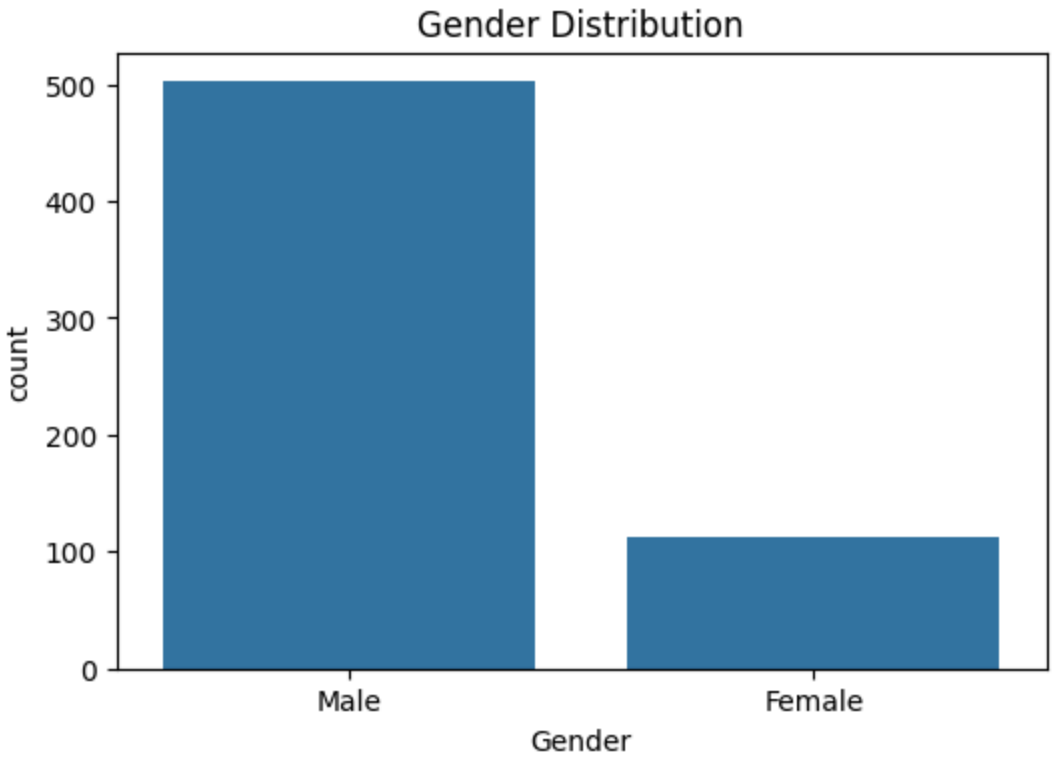
In [18]:
```python
# Imputation: Fill missing categorical values with mode
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
print(df)
```

```
      Loan_ID  Gender Married Dependents     Education Self_Employed  \
0    LP001002    Male      No          0      Graduate            No
1    LP001003    Male     Yes          1      Graduate            No
2    LP001005    Male     Yes          0      Graduate           Yes
3    LP001006    Male     Yes          0  Not Graduate            No
4    LP001008    Male      No          0      Graduate            No
..        ...     ...     ...        ...           ...           ...
609  LP002978  Female      No          0      Graduate            No
610  LP002979    Male     Yes         3+      Graduate            No
611  LP002983    Male     Yes          1      Graduate            No
612  LP002984    Male     Yes          2      Graduate            No
613  LP002990  Female      No          0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0               5849                0.0  146.412162             360.0
1               4583             1508.0  128.000000             360.0
2               3000                0.0   66.000000             360.0
3               2583             2358.0  120.000000             360.0
4               6000                0.0  141.000000             360.0
..               ...                ...         ...               ...
609             2900                0.0   71.000000             360.0
610             4106                0.0   40.000000             180.0
611             8072              240.0  253.000000             360.0
612             7583                0.0  187.000000             360.0
613             4583                0.0  133.000000             360.0

     Credit_History Property_Area Loan_Status
0               1.0         Urban           Y
1               1.0         Rural           N
2               1.0         Urban           Y
3               1.0         Urban           Y
4               1.0         Urban           Y
..              ...           ...         ...
609             1.0         Rural           Y
610             1.0         Rural           Y
611             1.0         Urban           Y
612             1.0         Urban           Y
613             0.0     Semiurban           N

[614 rows x 13 columns]
```

In [26]:
```
plt.figure(figsize=(6,4))
sns.histplot(df['LoanAmount'], kde=True)
plt.title('Loan Amount Distribution')
plt.show()
```

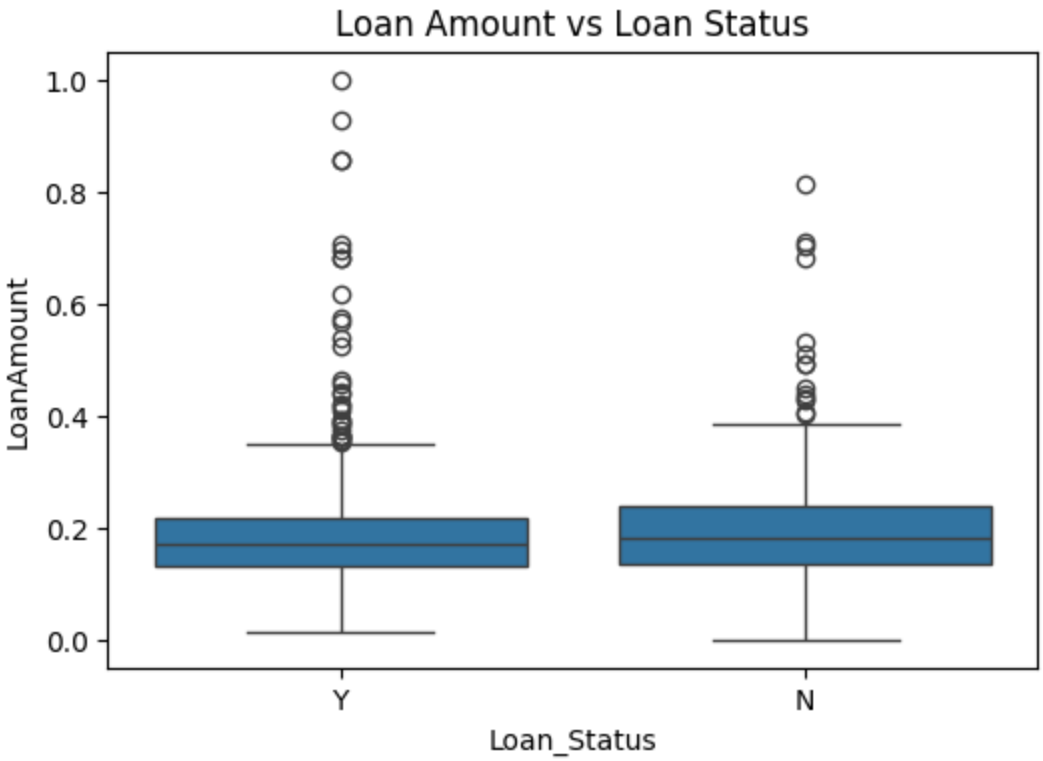


In [27]:
```
# Count plot for categorical variable (e.g., Gender)
plt.figure(figsize=(6,4))
sns.countplot(data=df, x='Gender')
plt.title('Gender Distribution')
plt.show()
```

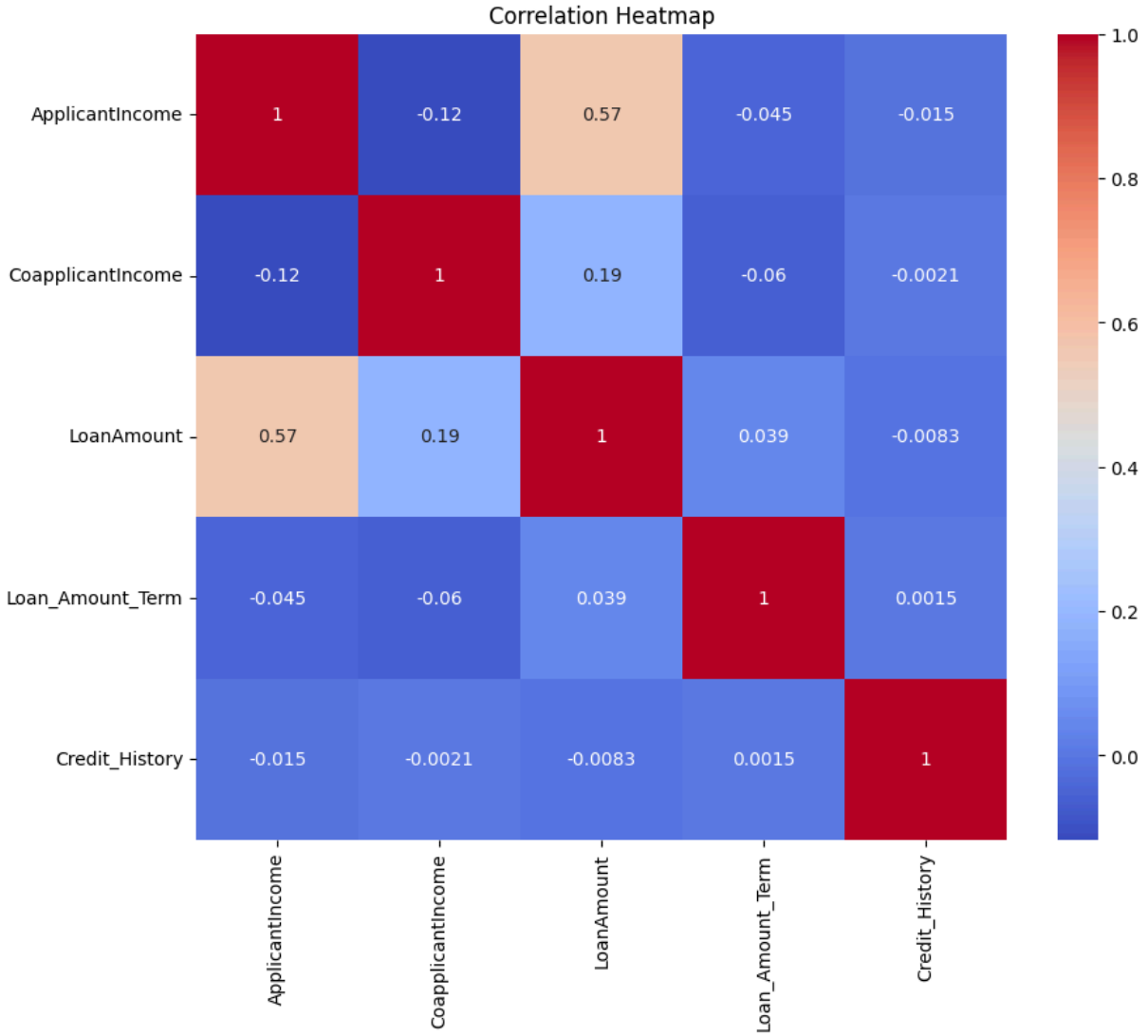


In [29]:
```
# 6. Bivariate Analysis (loan amount vs. loan status)
plt.figure(figsize=(6,4))
```

```
sns.boxplot(data=df, x='Loan_Status', y='LoanAmount')
plt.title('Loan Amount vs Loan Status')
plt.show()
```



In [31]:
```
# 7. Correlation Heatmap (numerical variables)
plt.figure(figsize=(10,8))
# Select only numerical features for correlation calculation
numerical_df = df.select_dtypes(include=['number'])
sns.heatmap(numerical_df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



In [34]:
```
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Step 2: Normalize using MinMaxScaler
scaler = MinMaxScaler()

# Create a new DataFrame to store normalized values
df_normalized = df.copy()
df_normalized[numerical_columns] = scaler.fit_transform(df[numerical_columns])

# Check the normalized DataFrame
print(df_normalized.head())
```

```
    Loan_ID Gender Married Dependents     Education Self_Employed  \
0  LP001002   Male      No          0      Graduate            No
1  LP001003   Male     Yes          1      Graduate            No
2  LP001005   Male     Yes          0      Graduate           Yes
3  LP001006   Male     Yes          0  Not Graduate            No
4  LP001008   Male      No          0      Graduate            No

   ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0         0.070489           0.000000    0.198860           0.74359
1         0.054830           0.036192    0.172214           0.74359
2         0.035250           0.000000    0.082489           0.74359
3         0.030093           0.056592    0.160637           0.74359
4         0.072356           0.000000    0.191027           0.74359

   Credit_History Property_Area Loan_Status
0             1.0         Urban           Y
1             1.0         Rural           N
2             1.0         Urban           Y
3             1.0         Urban           Y
4             1.0         Urban           Y
```

In [36]:
```
# Normalizing 'loan_amount' and 'income' columns
scaler = MinMaxScaler()
df[['LoanAmount', 'ApplicantIncome']] = scaler.fit_transform(df[['LoanAmount', 'App
print(df)
```

```
      Loan_ID  Gender Married Dependents     Education Self_Employed  \
0    LP001002    Male      No          0      Graduate            No
1    LP001003    Male     Yes          1      Graduate            No
2    LP001005    Male     Yes          0      Graduate           Yes
3    LP001006    Male     Yes          0  Not Graduate            No
4    LP001008    Male      No          0      Graduate            No
..        ...     ...     ...        ...           ...           ...
609  LP002978  Female      No          0      Graduate            No
610  LP002979    Male     Yes         3+      Graduate            No
611  LP002983    Male     Yes          1      Graduate            No
612  LP002984    Male     Yes          2      Graduate            No
613  LP002990  Female      No          0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
0           0.070489                0.0    0.198860          0.743590
1           0.054830             1508.0    0.172214          0.743590
2           0.035250                0.0    0.082489          0.743590
3           0.030093             2358.0    0.160637          0.743590
4           0.072356                0.0    0.191027          0.743590
..               ...                ...         ...               ...
609         0.034014                0.0    0.089725          0.743590
610         0.048930                0.0    0.044863          0.358974
611         0.097984              240.0    0.353111          0.743590
612         0.091936                0.0    0.257598          0.743590
613         0.054830                0.0    0.179450          0.743590

     Credit_History Property_Area Loan_Status
0               1.0         Urban           Y
1               1.0         Rural           N
2               1.0         Urban           Y
3               1.0         Urban           Y
4               1.0         Urban           Y
..              ...           ...         ...
609             1.0         Rural           Y
610             1.0         Rural           Y
611             1.0         Urban           Y
612             1.0         Urban           Y
613             0.0     Semiurban           N

[614 rows x 13 columns]
```

In [48]:
```
print(df.columns)
```

```
Index(['Loan_ID', 'Married', 'Dependents', 'Self_Employed', 'ApplicantIncome',
       'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History',
       'Property_Area', 'Loan_Status', 'Gender_Male',
       'Education_Not Graduate'],
      dtype='object')
```

In [93]:
```
# Display selected features to verify
print("Selected features before encoding:")
print(df_selected.head())
```

```
Selected features before encoding:
   LoanAmount  ApplicantIncome  Credit_History  Gender_Male Married
0    0.198860         0.070489             1.0         True      No
1    0.172214         0.054830             1.0         True     Yes
2    0.082489         0.035250             1.0         True     Yes
3    0.160637         0.030093             1.0         True     Yes
4    0.191027         0.072356             1.0         True      No
```

In [82]:
```
# Feature selection (assuming we want 'loan_amount', 'income','credit_score')
selected_features = ['LoanAmount', 'ApplicantIncome', 'Credit_History', 'Gender_Mal
df_selected = df[selected_features]
```

In [92]:
```
# Display original DataFrame columns
print("Original DataFrame columns:")
print(df.columns)

# One-Hot Encoding for 'Gender' and 'Married' columns
df_encoded = pd.get_dummies(df, columns=['Property_Area', 'Married'], drop_first=Tr

# Display the DataFrame after One-Hot Encoding
print("\nDataFrame after One-Hot Encoding for 'Gender' and 'Married':")
print(df_encoded.head())
```

```
Original DataFrame columns:
Index(['Loan_ID', 'Married', 'Dependents', 'ApplicantIncome',
       'CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term', 'Credit_History',
       'Property_Area', 'Gender_Male', 'Education_Not Graduate',
       'Loan_Status_Y', 'Self_Employed_Yes'],
      dtype='object')

DataFrame after One-Hot Encoding for 'Gender' and 'Married':
    Loan_ID Dependents  ApplicantIncome  CoapplicantIncome  LoanAmount  \
0  LP001002          0         0.070489                0.0    0.198860
1  LP001003          1         0.054830             1508.0    0.172214
2  LP001005          0         0.035250                0.0    0.082489
3  LP001006          0         0.030093             2358.0    0.160637
4  LP001008          0         0.072356                0.0    0.191027

   Loan_Amount_Term  Credit_History  Gender_Male  Education_Not Graduate  \
0           0.74359             1.0         True                   False
1           0.74359             1.0         True                   False
2           0.74359             1.0         True                   False
3           0.74359             1.0         True                    True
4           0.74359             1.0         True                   False

   Loan_Status_Y  Self_Employed_Yes  Property_Area_Semiurban  \
0           True              False                    False
1          False              False                    False
2           True               True                    False
3           True              False                    False
4           True              False                    False

   Property_Area_Urban  Married_Yes
0                 True        False
1                False         True
2                 True         True
3                 True         True
4                 True        False
```

In [95]:
```
# One-Hot Encoding for nominal categories (e.g., 'Gender' and 'Education')
df_encoded = pd.get_dummies(df_selected, columns=['Gender_Male', 'Married'], drop_f
```

In [98]:
```
# Assuming 'Property_Area' is in the original DataFrame (df)
le = LabelEncoder()
df['Property_Area_Encoded'] = le.fit_transform(df['Property_Area'])

# Now, include 'Property_Area_Encoded' in your feature selection
selected_features = ['LoanAmount', 'ApplicantIncome', 'Credit_History', 'Gender_Mal
df_selected = df[selected_features]

# Proceed with One-Hot Encoding
df_encoded = pd.get_dummies(df_selected, columns=['Gender_Male', 'Married'], drop_f

# df_encoded should now contain all your desired features, including the encoded 'P
```

In [99]:
```
print("\nDataFrame after Label Encoding and One-Hot Encoding:")
print(df_encoded.head())
```

```
DataFrame after Label Encoding and One-Hot Encoding:
   LoanAmount  ApplicantIncome  Credit_History  Property_Area_Encoded  \
0    0.198860         0.070489             1.0                      2
1    0.172214         0.054830             1.0                      0
2    0.082489         0.035250             1.0                      2
3    0.160637         0.030093             1.0                      2
4    0.191027         0.072356             1.0                      2

   Gender_Male_True  Married_Yes
0              True        False
1              True         True
2              True         True
3              True         True
4              True        False
```

In [107…
```
df = pd.read_csv("loan-data.csv", names=['Loan-Status', 'Loan_ID', 'Loan_History'],
```

In [108…
```
# Define features (X) and target (y)
X = df.drop('Loan-Status', axis=1)
y = df['Loan-Status']
```

In [109…
```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [112…
```
# Display the shapes of the resulting datasets
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
X_train shape: (491, 2)
X_test shape: (123, 2)
y_train shape: (491,)
y_test shape: (123,)
```

In [ ]: