In [70]:

```python
import pandas as pd
import numpy as np

df = pd.read_csv('loan_prediction_datasets.csv')
df
```

Out[70]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | Loa |
|---|---|---|---|---|---|---|---|---|---|
| **0** | LP001015 | Male | Yes | 0 | Graduate | No | 5720 | 0 | |
| **1** | LP001022 | Male | Yes | 1 | Graduate | No | 3076 | 1500 | |
| **2** | LP001031 | Male | Yes | 2 | Graduate | No | 5000 | 1800 | |
| **3** | LP001035 | Male | Yes | 2 | Graduate | No | 2340 | 2546 | |
| **4** | LP001051 | Male | No | 0 | Not Graduate | No | 3276 | 0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **362** | LP002971 | Male | Yes | 3+ | Not Graduate | Yes | 4009 | 1777 | |
| **363** | LP002975 | Male | Yes | 0 | Graduate | No | 4158 | 709 | |
| **364** | LP002980 | Male | No | 0 | Graduate | No | 3250 | 1993 | |
| **365** | LP002986 | Male | Yes | 0 | Graduate | No | 5000 | 2393 | |
| **366** | LP002989 | Male | No | 0 | Graduate | Yes | 9200 | 0 | |

367 rows × 12 columns

In [71]:

```python
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())
print(df[['LoanAmount','Loan_Amount_Term','Credit_History']])
```

```
     LoanAmount  Loan_Amount_Term  Credit_History
0         110.0             360.0        1.000000
1         126.0             360.0        1.000000
2         208.0             360.0        1.000000
3         100.0             360.0        0.825444
4          78.0             360.0        1.000000
..          ...               ...             ...
362       113.0             360.0        1.000000
363       115.0             360.0        1.000000
364       126.0             360.0        0.825444
365       158.0             360.0        1.000000
366        98.0             180.0        1.000000

[367 rows x 3 columns]
```

In [72]:
```python
df_copy = df.copy()

df_copy['Gender'] = df_copy['Gender'].map({'Male':1,'Female': 0})

df_copy['Married'] = df_copy['Married'].map({ 'Yes': 1,'No' : 0 })

df_copy['Dependents'] = df_copy['Dependents'].map({'1':1,'0': 0, '2':2 , '3+' : 3})

df_copy['Education'] = df_copy['Education'].map({'Graduate':1, 'Not Graduate' : 0})

df_copy['Self_Employed'] = df_copy['Self_Employed'].map({'Yes':1, 'No' : 0})

df_copy['Property_Area'] = df_copy['Property_Area'].map({'Urban':2, 'Semiurban' : 1 ,'Rural'

print(df_copy[['Gender', 'Married','Dependents','Education','Self_Employed','Property_Area']]
```

```
     Gender  Married  Dependents  Education  Self_Employed  Property_Area
0       1.0        1         0.0          1            0.0              2
1       1.0        1         1.0          1            0.0              2
2       1.0        1         2.0          1            0.0              2
3       1.0        1         2.0          1            0.0              2
4       1.0        0         0.0          0            0.0              2
..      ...      ...         ...        ...            ...            ...
362     1.0        1         3.0          0            1.0              2
363     1.0        1         0.0          1            0.0              2
364     1.0        0         0.0          1            0.0              1
365     1.0        1         0.0          1            0.0              0
366     1.0        0         0.0          1            1.0              0

[367 rows x 6 columns]
```

In [73]:
```python
df['TotalIncome'] = df['ApplicantIncome'] + df['CoapplicantIncome']
print(df['TotalIncome'])
```

```
0      5720
1      4576
2      6800
3      4886
4      3276
       ... 
362    5786
363    4867
364    5243
365    7393
366    9200
Name: TotalIncome, Length: 367, dtype: int64
```

In [74]:
```python
df['DTI'] = df['LoanAmount'] / (df['TotalIncome'] + 1)
print(df['DTI'])
```

```
0      0.019227
1      0.027529
2      0.030584
3      0.020462
4      0.023802
         ...   
362    0.019527
363    0.023624
364    0.024027
365    0.021369
366    0.010651
Name: DTI, Length: 367, dtype: float64
```

```
In [75]: df['AI'] = df['LoanAmount'] / df['TotalIncome']
         df
```

Out[75]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | CoapplicantIncome | Loa |
|---|---|---|---|---|---|---|---|---|---|
| **0** | LP001015 | Male | Yes | 0 | Graduate | No | 5720 | 0 | |
| **1** | LP001022 | Male | Yes | 1 | Graduate | No | 3076 | 1500 | |
| **2** | LP001031 | Male | Yes | 2 | Graduate | No | 5000 | 1800 | |
| **3** | LP001035 | Male | Yes | 2 | Graduate | No | 2340 | 2546 | |
| **4** | LP001051 | Male | No | 0 | Not Graduate | No | 3276 | 0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **362** | LP002971 | Male | Yes | 3+ | Not Graduate | Yes | 4009 | 1777 | |
| **363** | LP002975 | Male | Yes | 0 | Graduate | No | 4158 | 709 | |
| **364** | LP002980 | Male | No | 0 | Graduate | No | 3250 | 1993 | |
| **365** | LP002986 | Male | Yes | 0 | Graduate | No | 5000 | 2393 | |
| **366** | LP002989 | Male | No | 0 | Graduate | Yes | 9200 | 0 | |

367 rows × 15 columns

```
In [76]: from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LinearRegression
         from sklearn.metrics import mean_squared_error, r2_score

         df["Credit_History"] = df["Credit_History"].fillna(df["Credit_History"].mean())
         X = df[['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_History
         y = df['AI']

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

         model = LinearRegression()

         model.fit(X_train, y_train)

         y_pred = model.predict(X_test)


         print("Mean Squared Error : ", mean_squared_error(y_test, y_pred))
         print("R2 Score :", r2_score(y_test, y_pred))
```

```
Mean Squared Error :  1.9730437205553017e-12
R2 Score : 0.999999984794658
```

In [80]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,f1_score,accuracy_score

# y_pred_porb = model.predict_prob(X_test)
x = df[['Credit_History','DTI','TotalIncome']]
y = df['AI']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

model = LogisticRegression(max_iter = 1000)

model.fit(x_train, y_train)

yy_pred = model.predict(x_test)
print("Confusion Matrix: ", confusion_matrix(y_test, yy_pred))

print("F1 Score: ",f1_score(y_true, y_pred, average='macro'))

print("Accuracy_ score: ",accuracy_score(y_true, y_pred))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [80], in <cell line: 11>()
      7 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_stat
e=42)
      9 model = LogisticRegression(max_iter = 1000)
---> 11 model.fit(x_train, y_train)
     13 yy_pred = model.predict(x_test)
     14 print("Confusion Matrix: ", confusion_matrix(y_test, yy_pred))

File C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:1516, in L
ogisticRegression.fit(self, X, y, sample_weight)
   1506     _dtype = [np.float64, np.float32]
   1508 X, y = self._validate_data(
   1509     X,
   1510     y,
   (...)
   1514     accept_large_sparse=solver not in ["liblinear", "sag", "saga"],
   1515 )
-> 1516 check_classification_targets(y)
   1517 self.classes_ = np.unique(y)
   1519 multi_class = _check_multi_class(self.multi_class, solver, len(self.classes_))

File C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\multiclass.py:197, in check_cl
assification_targets(y)
    189 y_type = type_of_target(y)
    190 if y_type not in [
    191     "binary",
    192     "multiclass",
   (...)
    195     "multilabel-sequences",
    196 ]:
--> 197     raise ValueError("Unknown label type: %r" % y_type)

ValueError: Unknown label type: 'continuous'
```

In [ ]:
```
```