In [31]:

```
# Import libraries
import pandas as pd
import numpy as np
import warnings
## Data Visualization
import seaborn as sns
import matplotlib.pyplot as plt

# Configure libraries
warnings.filterwarnings('ignore')
plt.rcParams['figure.figsize'] = (10, 10)
plt.style.use('seaborn')
```

In [32]:

```
# Load dataset
df_bank = pd.read_csv('Loan Prediction Dataset.csv')

# print(df_bank.info())
print('Shape of dataframe:', df_bank.shape)
df_bank.head()
```

```
Shape of dataframe: (614, 12)
```

Out[32]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	Male	No	0	Graduate	No	5849	0.0
1	Male	Yes	1	Graduate	No	4583	1508.0
2	Male	Yes	0	Graduate	Yes	3000	0.0
3	Male	Yes	0	Not Graduate	No	2583	2358.0
4	Male	No	0	Graduate	No	6000	0.0

In [33]:

```
# class distribution
df_bank['Married'].value_counts()
```

Out[33]:

```
Yes    398
No     213
Name: Married, dtype: int64
```

In [34]:
```
# handling missing values
df_bank.isnull().sum()
```

Out[34]:
```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [35]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
# Handle non-numeric columns
non_numeric_cols = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Empl
label_encoder = LabelEncoder()
for col in non_numeric_cols:
    df_bank[col] = label_encoder.fit_transform(df_bank[col])

# Now, scale the numeric columns
scaler = StandardScaler()
numeric_cols = ['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount']
df_bank[numeric_cols] = scaler.fit_transform(df_bank[numeric_cols])

df_bank.head()
```

Out[35]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	1	0	0	0	0	0.072991	-0.554487
1	1	1	1	0	0	-0.134412	-0.038732
2	1	1	0	0	1	-0.393747	-0.554487
3	1	1	0	1	0	-0.462062	0.251980
4	1	0	0	0	0	0.097728	-0.554487

In [36]:

```
df_bank_ready = df_bank.copy()

# Split dataset into training and testing

# Select Features
feature = df_bank_ready.drop('Loan_Status', axis=1)

# Select Target
target = df_bank_ready['Loan_Status']

# Set Training and Testing Data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(feature , target, shuffle

# Show the Training and Testing Data
print('Shape of training feature:', X_train.shape)
print('Shape of testing feature:', X_test.shape)
print('Shape of training label:', y_train.shape)
print('Shape of training label:', y_test.shape)
```

```
Shape of training feature: (491, 11)
Shape of testing feature: (123, 11)
Shape of training label: (491,)
Shape of training label: (123,)
```

In [37]:

```
# Modelling

def evaluate_model(model, x_test, y_test):
    from sklearn import metrics

    # Predict Test Data
    y_pred = model.predict(x_test)

    # Calculate accuracy, precision, recall, f1-score, and kappa score
    acc = metrics.accuracy_score(y_test, y_pred)
    prec = metrics.precision_score(y_test, y_pred)
    rec = metrics.recall_score(y_test, y_pred)
    f1 = metrics.f1_score(y_test, y_pred)
    kappa = metrics.cohen_kappa_score(y_test, y_pred)

    # Calculate area under curve (AUC)
    y_pred_proba = model.predict_proba(x_test)[::,1]
    fpr, tpr, _ = metrics.roc_curve(y_test, y_pred_proba)
    auc = metrics.roc_auc_score(y_test, y_pred_proba)

    # Display confussion matrix
    cm = metrics.confusion_matrix(y_test, y_pred)

    return {'acc': acc, 'prec': prec, 'rec': rec, 'f1': f1, 'kappa': kappa,
            'fpr': fpr, 'tpr': tpr, 'auc': auc, 'cm': cm}
```

In [40]:
```
# Check for missing values in X_train and X_test
missing_train = X_train.isna().sum()
missing_test = X_test.isna().sum()

print("Missing values in X_train:\n", missing_train)
print("Missing values in X_test:\n", missing_test)
```

```
Missing values in X_train:
 Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           17
Loan_Amount_Term     10
Credit_History       44
Property_Area         0
dtype: int64
Missing values in X_test:
 Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           5
Loan_Amount_Term     4
Credit_History       6
Property_Area        0
dtype: int64
```

In [42]:

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean')  # Replace 'mean' with your chosen st
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

print("Missing values in X_train:\n", missing_train)
print("Missing values in X_test:\n", missing_test)
```

```
Missing values in X_train:
 Gender                0
Married               0
Dependents            0
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           17
Loan_Amount_Term     10
Credit_History       44
Property_Area         0
dtype: int64
Missing values in X_test:
 Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           5
Loan_Amount_Term     4
Credit_History       6
Property_Area        0
dtype: int64
```

In [43]:

```
from sklearn import tree

# Building Decision Tree model
dtc = tree.DecisionTreeClassifier(random_state=0)
dtc.fit(X_train, y_train)

# Evaluate Model
dtc_eval = evaluate_model(dtc, X_test, y_test)

# Print result
print('Accuracy:', dtc_eval['acc'])
print('Precision:', dtc_eval['prec'])
print('Recall:', dtc_eval['rec'])
print('F1 Score:', dtc_eval['f1'])
print('Cohens Kappa Score:', dtc_eval['kappa'])
print('Area Under Curve:', dtc_eval['auc'])
print('Confusion Matrix:\n', dtc_eval['cm'])
```

```
Accuracy: 0.7073170731707317
Precision: 0.7857142857142857
Recall: 0.7857142857142857
F1 Score: 0.7857142857142857
Cohens Kappa Score: 0.32417582417582413
Area Under Curve: 0.6620879120879121
Confusion Matrix:
 [[21 18]
 [18 66]]
```