**DISEASE PREDICTION USING MACHINE LEARNING**

**Final Project Report**

1. Introduction
   1. Project overviews
   2. Objectives
2. Project Initialization and Planning Phase
   1. Define Problem Statement
   2. Project Proposal (Proposed Solution)
   3. Initial Project Planning
3. Data Collection and Preprocessing Phase
   1. Data Collection Plan and Raw Data Sources Identified
   2. Data Quality Report
   3. Data Exploration and Preprocessing
4. Model Development Phase
   1. Feature Selection Report
   2. Model Selection Report
   3. Initial Model Training Code, Model Validation and Evaluation Report
5. Model Optimization and Tuning Phase
   1. Hyper-parameter Tuning Documentation
   2. Performance Metrics Comparison Report
   3. Final Model Selection Justification
6. Results
   1. Output Screenshots
7. Advantages & Disadvantages
8. Conclusion
9. Future Scope
10. Appendix
    1. Source Code
    2. GitHub & Project Demo Link

**1.INTRODUCTION**

**1.1 Project overview**

The project aims to analyze loan prediction using machine learning, representing a transformative approach in financial services aimed at early detection, prevention, and management of loan defaults through advanced computational techniques. This overview delves into the application of machine learning models in loan prediction, highlighting their capability to analyze diverse datasets—including credit history, income, and other financial factors—to forecast loan eligibility, default risk, and loan amount sanction. Key methodologies such as feature selection, model training, and validation are discussed, along with the challenges of data integration, model interpretability, and scalability.

**1.2 Objectives**

In today’s fast-paced world, financial institutions require robust methods to assess the risk associated with loan applications. Manual evaluations are often time-consuming and subjective. Using machine learning models, we can predict loan eligibility and amount sanctions efficiently and accurately. This model can be used by financial institutions to streamline their loan approval process or by individuals to understand their eligibility and potential loan amount before applying. The project aims to predict loan eligibility and the sanction amount for up to 42 different financial scenarios based on a set of input parameters.

**2. Project Initialization and Planning Phase**

**2.1 Define Problem Statement**

The early prediction of loan eligibility and the appropriate sanction amount can significantly improve decision-making in financial institutions, reduce default rates, and enhance customer satisfaction. With the advent of big data, machine learning, and advanced analytics, there is a growing potential to predict loan outcomes based on historical data and financial indicators. This predictive capability can aid in proactive decision-making, risk management, and personalized financial services. Accurately predicting loan eligibility and amount sanction is crucial for early intervention and improved financial outcomes.

**2.2 Project Proposal (Proposed solution)**

This project proposal outlines a solution to address the problem of early loan prediction through machine learning. With a clear objective to develop a predictive model for assessing loan eligibility and amount sanction based on applicant's financial data, the proposal defines the scope of the project, including data collection, model development, and deployment. The proposed solution details the approach to be used, key features of the model, and specifies the resource requirements including hardware, software, and personnel. By creating an accurate and user-friendly tool, the project aims to enable efficient loan processing and improve early loan approval.

**2.3 Initial Project Planning**

 Initial Project Planning involves outlining key objectives, defining scope, and identifying financial patterns.

 It encompasses setting timelines, allocating resources, and determining the overall project strategy.

**3. Data Collection and Preprocessing Phase**

**3.1 Data Collection Plan and Raw Data Sources Identified**

 The dataset for "Loan Prediction Using Machine Learning" is sourced from Kaggle and other financial data repositories.

 Extract data from existing financial systems, conduct financial surveys, or gather data from financial applications.

 API integration for financial systems, online survey tools, data extraction scripts.

 Gathered a dataset from Kaggle containing applicant information such as age, gender, income, credit history, and loan details for loan prediction. The dataset includes features relevant for building and training the prediction model, enabling accurate risk assessments and analysis.

**3.2 Data Quality Report**

The Data Quality Report will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

**3.3 Data Exploration and preprocessing**

Dataset variables will be statistically analyzed to identify patterns and outliers, with Python employed for preprocessing tasks like normalization and feature engineering. Data cleaning will address missing values and outliers, ensuring quality for subsequent analysis and modeling, and forming a strong foundation for insights and predictions.

**4. Model Development Phase**

**4.1 Feature Selection Report**

Feature selection for loan prediction in machine learning involves identifying and choosing relevant features from data to improve model accuracy and interpretability. Techniques such as filter methods (e.g., correlation), wrapper methods (e.g., recursive feature elimination), and embedded methods (e.g., Lasso regression) are commonly used to select optimal features. This process helps mitigate overfitting, reduce computational complexity, and enhance the predictive power of models.

**4.2 Model Selection Report**

Model selection for loan prediction in machine learning entails evaluating various algorithms (e.g., SVM, Random Forest, Neural Networks) based on performance metrics like accuracy, sensitivity, and specificity. The goal is to identify the model that best balances predictive power, interpretability, and computational efficiency for the specific loan prediction task.

**4.3 Initial Model Training Code, Model Validation and valuation Report**

Initial model training involves splitting data into training and validation sets, fitting models (e.g., SVM, Random Forest) on training data, and tuning hyper-parameters via techniques like cross-validation. Evaluation includes assessing metrics such as accuracy, precision, recall, and ROC AUC to gauge model performance and generalizability for loan prediction tasks.

**5.Model Optimization and Tuning Phase**

**Final Model Selection Justification**

The final model choice is justified based on its superior performance metrics (e.g., highest accuracy, AUC), robustness to cross-validation, and interpretability of feature importance. This model demonstrates optimal balance between predictive power and computational efficiency for accurate loan prediction.

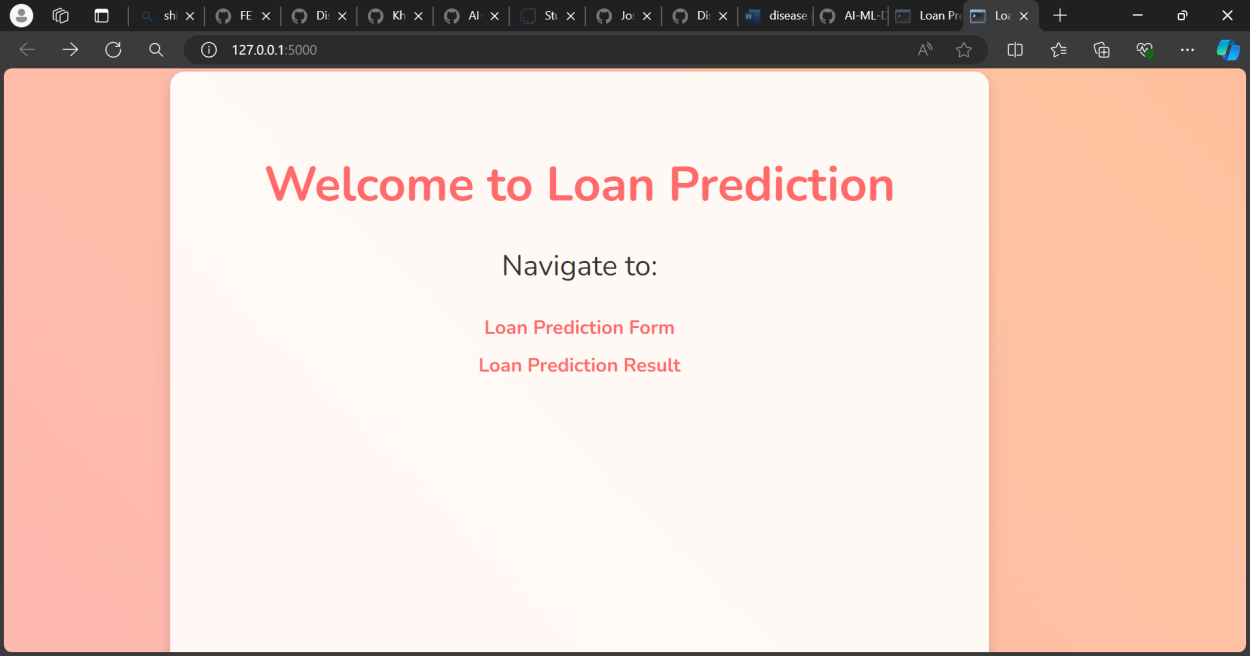
**6. RESULTS**

**6.1 Output Screenshots**

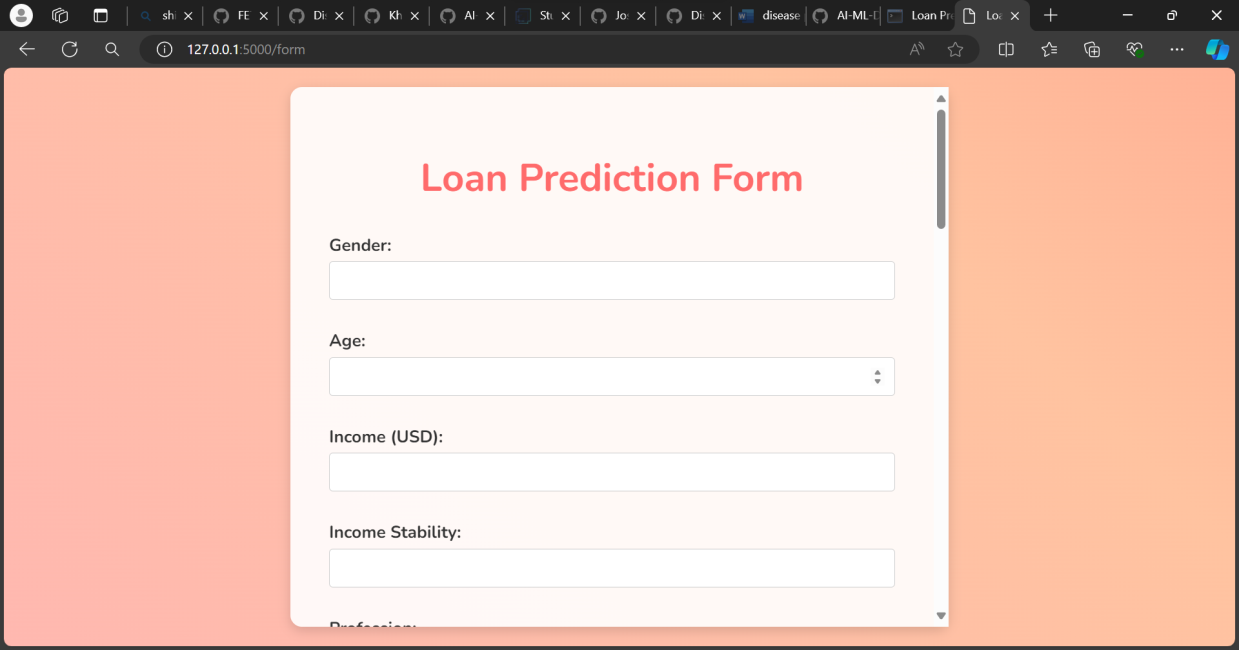
**Index.**

**Index.HTML**

**HOME PAGE**

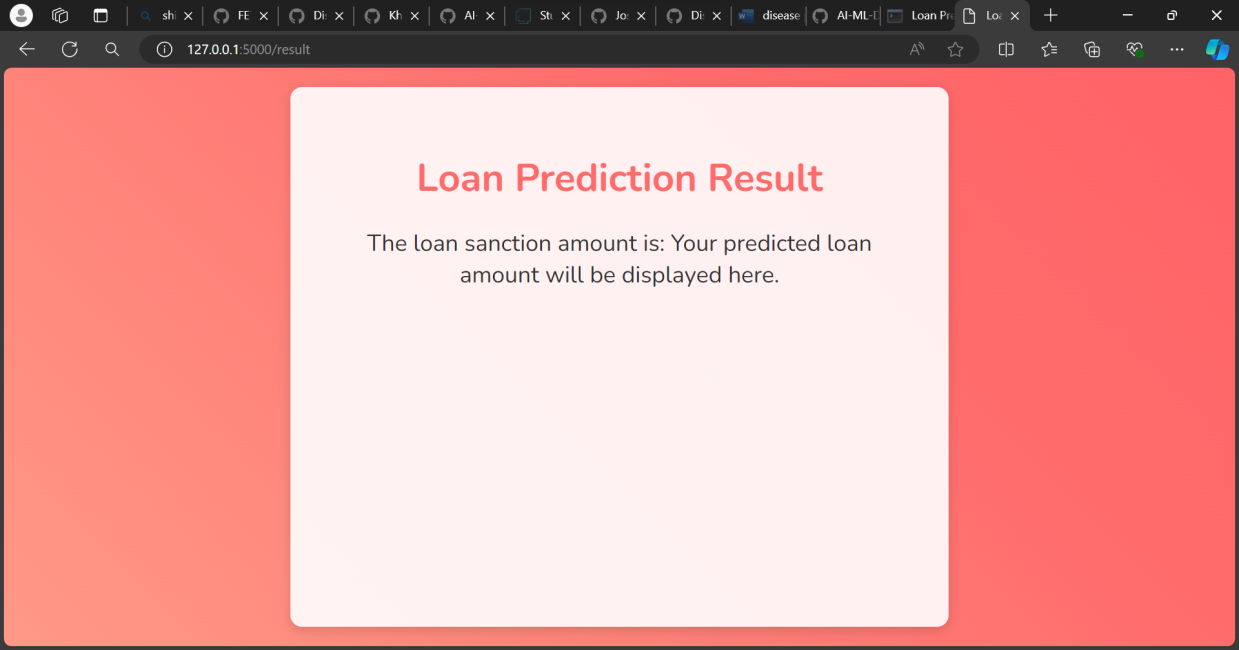


**Form.html**

****

**OUTPUT PAGE**

RESULT.HTML



**7. Advantages and disadvantages**

## 7. ADVANTAGES AND DISADVANTAGES

### Advantages:

1. **Early Detection**: Machine learning models can analyze large amounts of financial data to detect patterns and anomalies that may indicate the early stages of loan default risk. This allows for timely intervention and better decision-making, potentially reducing financial losses.
2. **Personalized Risk Assessment**: ML algorithms can assess individual risk factors based on personal financial data, credit history, lifestyle choices, and other factors, providing a more personalized assessment of loan eligibility and risk.
3. **Improved Accuracy**: ML models can process complex datasets and identify subtle correlations that may not be apparent through traditional statistical methods, leading to more accurate predictions of loan eligibility and the sanction amount.
4. **Operational Efficiency**: By automating the analysis of large datasets, ML optimizes financial resources, reduces processing times, and enhances the efficiency of loan approval processes.
5. **Scalability**: ML models can easily scale to handle increasing volumes of data, making them suitable for large financial institutions with vast amounts of loan application data.

### Disadvantages:

1. **Complexity and Interpretability**: ML models can be complex, making it hard to interpret results. Financial institutions may find it challenging to explain decisions to stakeholders and regulators.
2. **Data Quality**: ML relies heavily on high-quality, unbiased data for accurate predictions. Poor data quality can lead to inaccurate predictions and potential financial risks.
3. **Overfitting**: Models may fit training data too closely, leading to poor generalization when applied to new, unseen data, potentially reducing the reliability of predictions.
4. **Causality**: Difficulty in distinguishing correlation from causation in predictions. ML models may identify patterns that are not causally related to loan defaults, leading to misleading conclusions.
5. **Bias and Fairness**: Risk of inheriting biases from training data, impacting fairness and leading to discriminatory practices in loan approval processes.

## 8. CONCLUSION

In conclusion, the implementation of machine learning for loan prediction and loan amount sanction represents a pivotal advancement in financial analytics. By harnessing the power of predictive models and data-driven insights, this project not only enhances the accuracy of loan eligibility assessments but also revolutionizes early risk detection strategies. Through meticulous data preprocessing, feature engineering, model training, and validation, we have established a robust framework capable of forecasting loan risks with unprecedented precision. Moreover, the integration of user-friendly interfaces empowers financial professionals and applicants alike to make informed decisions, leading to timely loan approvals and improved financial outcomes.

## 9. FUTURE SCOPE

1. **Personalized Financial Services**: Use ML algorithms to personalize loan offers and financial services based on individual applicant data, including credit profiles, income patterns, and spending habits.
2. **Early Detection and Risk Management**: Develop machine learning models to detect potential loan default risks at early stages by analyzing various financial indicators, credit data, and transaction histories.
3. **Predictive Analytics**: Improve loan prediction models to forecast the likelihood of loan default based on demographic data, economic conditions, and market trends.
4. **Population Financial Management**: Apply machine learning to analyze large-scale financial data to identify trends, patterns, and potential market shifts, aiding in proactive financial management and policy-making.
5. **Continuous Learning and Improvement**: Establish frameworks for continuous learning and improvement of machine learning models using real-world financial data and feedback from financial professionals.

**10.Appendix**

**10.1. Source Code**

**Code Snippets**

# Load dataset

train\_dataset = pd.read\_csv('train.csv')

test\_dataset = pd.read\_csv('test.csv')

train\_dataset.describe()

# Dataset shape

# For train data

train\_rows = train\_dataset.shape[0]

train\_cols = train\_dataset.shape[1]

# For test data

# test\_rows = test\_dataset.shape[0]

# test\_cols = test\_dataset.shape[1]

print(f'There are {train\_rows} rows and {train\_cols} columns in train dataset')

print(f'There are {test\_rows} rows and {test\_cols} columns in test dataset')

# Data types

print('TRAIN DATASET :\n')

display(train\_dataset.dtypes)

print()

print('TEST DATASET :\n')

display(test\_dataset.dtypes)

# Checking for missing values

print('Missing values in Train Set : \n')

display(train\_dataset.isna().sum().sort\_values(ascending=False))

print('Missing values in Test set : \n')

display(test\_dataset.isna().sum().sort\_values(ascending=False))

# Missing values percent

train\_total\_cells, test\_total\_cells = (train\_rows \* train\_cols) , (test\_rows \* test\_cols)

train\_missing\_cells = train\_dataset.isna().sum().sum()

test\_missing\_cells = test\_dataset.isna().sum().sum()

train\_missing\_perc = np.round((train\_missing\_cells/ train\_total\_cells)\*100, 2)

test\_missing\_perc = np.round((test\_missing\_cells/ test\_total\_cells)\*100, 2)

print(f'Total missing percent in Train data is {train\_missing\_perc} %')

print(f'Total missing percent in Test data is {test\_missing\_perc} %')

# Checking for cardinality

print('Number of unique values in Train data : \n')

display(train\_dataset.nunique().sort\_values(ascending=False))

print('Number of unique values in Test data : \n')

display(test\_dataset.nunique().sort\_values(ascending=False))

# Checking for mis-spelled words

columns = train\_dataset.select\_dtypes('object').columns

for col in columns:

    if col == 'Customer ID':

        continue   # Excluding Customer ID feature

    print(f'For Train data : {train\_dataset[col].unique()}')

for col in columns:

    if col=='Customer ID':

        continue

    print(f'For Test data :{test\_dataset[col].unique()}')

# General Overview

train\_dataset['Loan Sanction Amount (USD)'].describe()

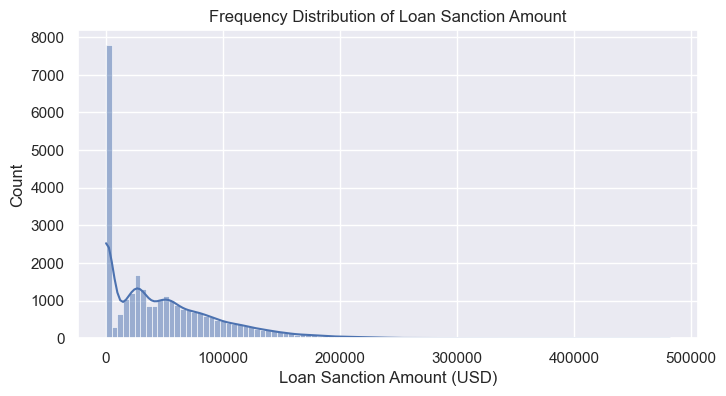
# Histogram Plot

plt.figure(figsize=(8,4))

sns.histplot(train\_dataset['Loan Sanction Amount (USD)'], kde=True)

plt.title('Frequency Distribution of Loan Sanction Amount')

plt.show()



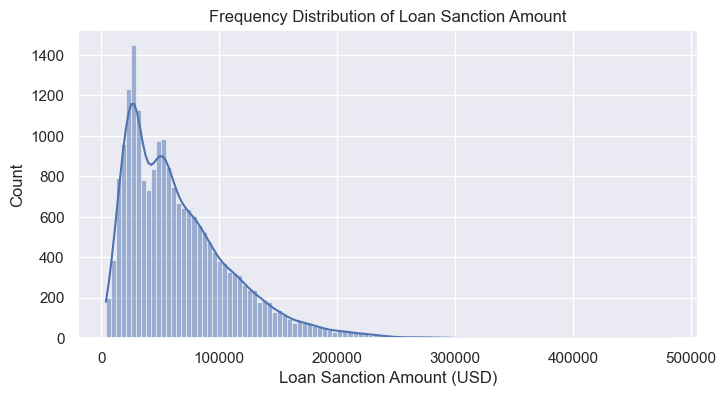
# Histogram Plot without zero

plt.figure(figsize=(8,4))

sns.histplot(target\_without\_zero['Loan Sanction Amount (USD)'], kde=True)

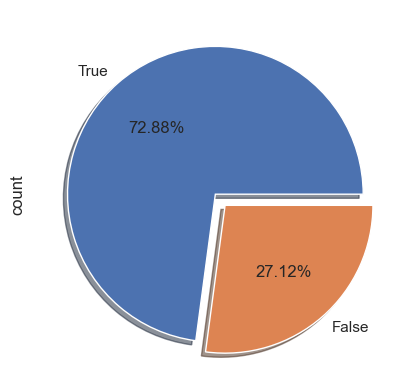
plt.title('Frequency Distribution of Loan Sanction Amount')

plt.show()



# Distribution of Loan sanctioned

train\_dataset['Loan sanctioned'].value\_counts().plot(kind='pie', autopct='%.2f%%', explode=[0, 0.1], shadow=True)



# Analysis for Regression

plt.figure(figsize=(13,10))

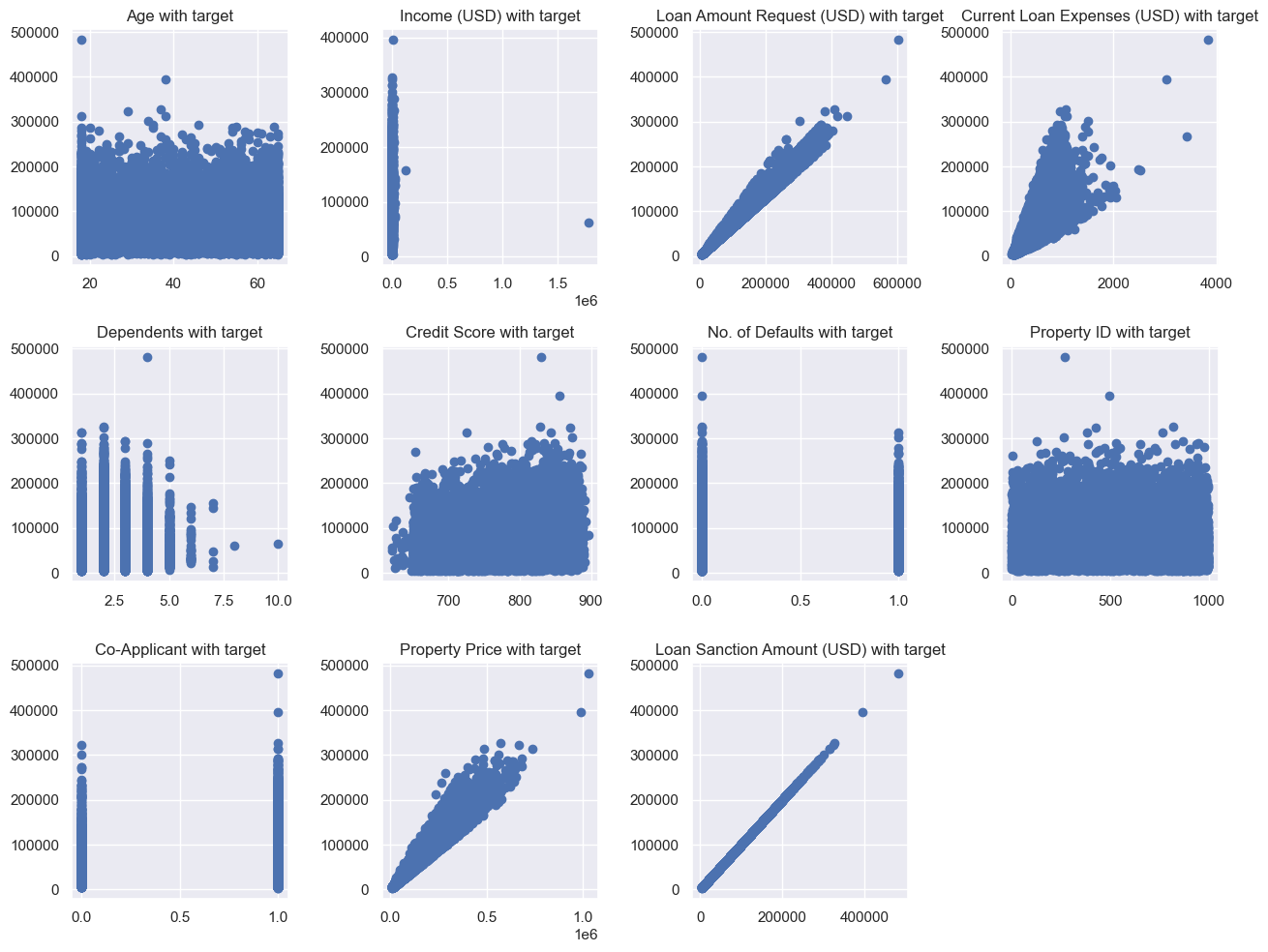
for index, col in enumerate(con\_cols):

    plt.subplot(3,4, index+1)

    plt.scatter(x=target\_without\_zero[col], y=target\_without\_zero['Loan Sanction Amount (USD)'])

    plt.title(f'{col} with target')

plt.tight\_layout()



# Analysis for Regression after removing some points

plt.figure(figsize=(13,10))

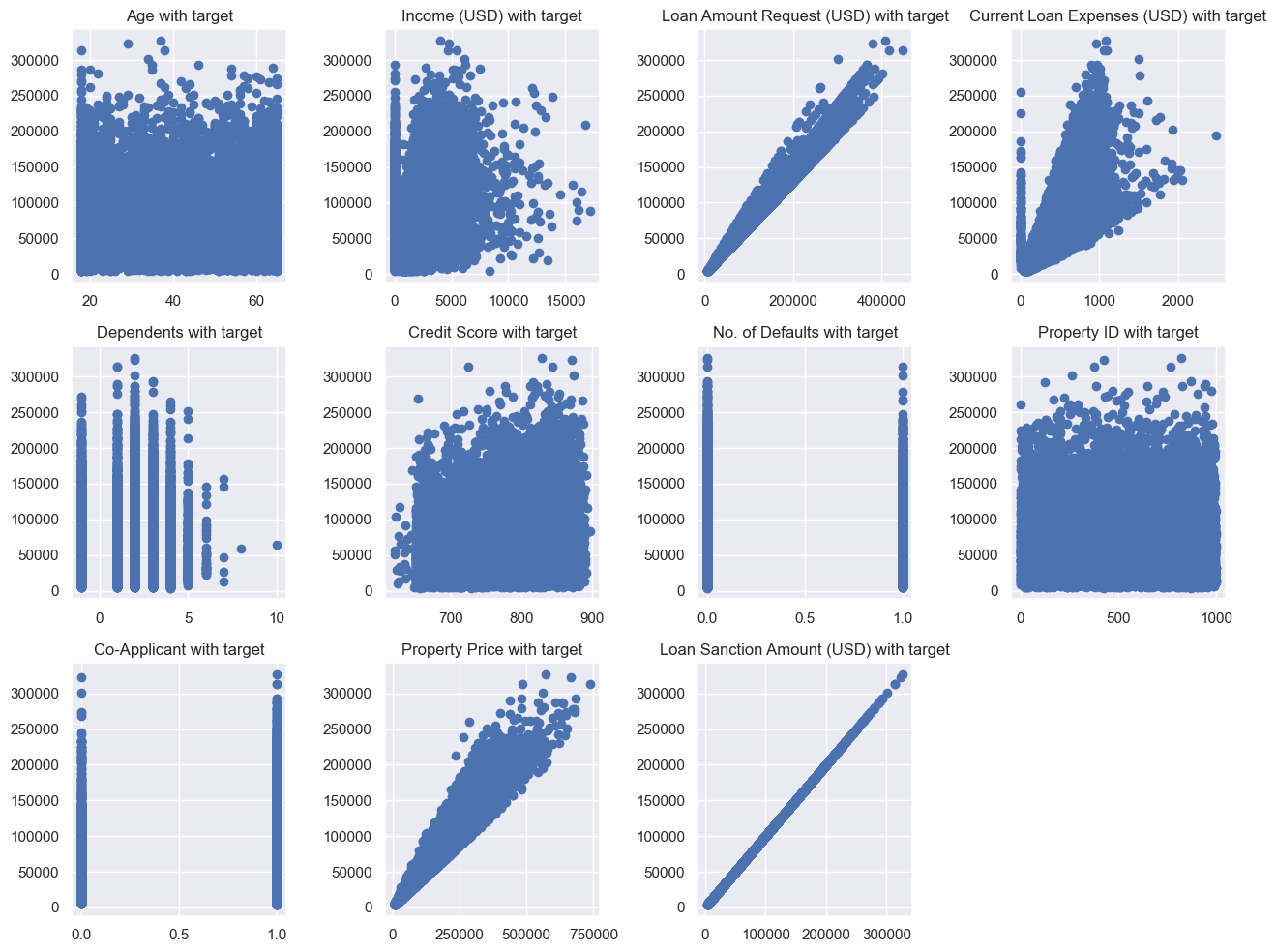
for index, col in enumerate(con\_cols):

    plt.subplot(3,4, index+1)

    plt.scatter(x=target\_without\_zero[col], y=target\_without\_zero['Loan Sanction Amount (USD)'])

    plt.title(f'{col} with target')

plt.tight\_layout()



# Distribution plot

plt.figure(figsize=(12,8))

for index, col in enumerate(con\_cols):

    if col =='Loan Sanction Amount (USD)':

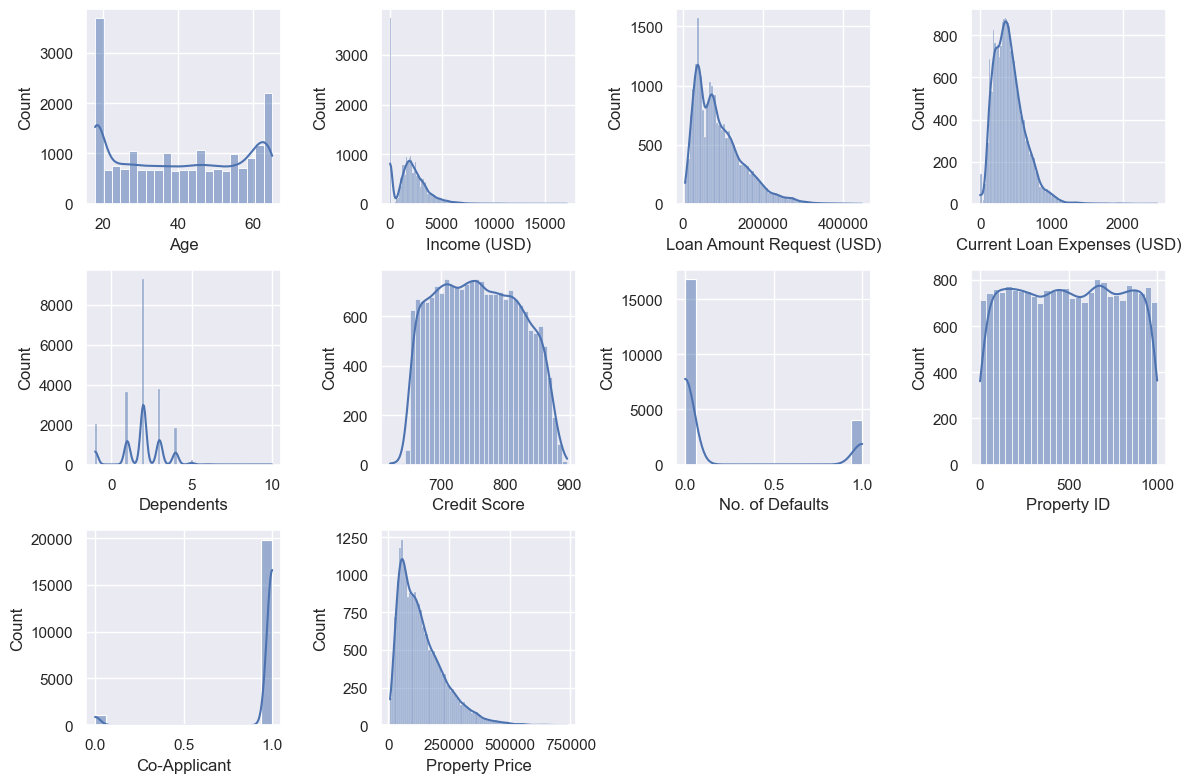
        continue

    plt.subplot(3,4, index+1)

    sns.histplot(x=col, kde=True, data=target\_without\_zero)

print('Distribution of continuous features')

plt.tight\_layout()



# Relation of categorical columns with target feature

plt.figure(figsize=(14,16))

for index, col in enumerate(cat\_cols):

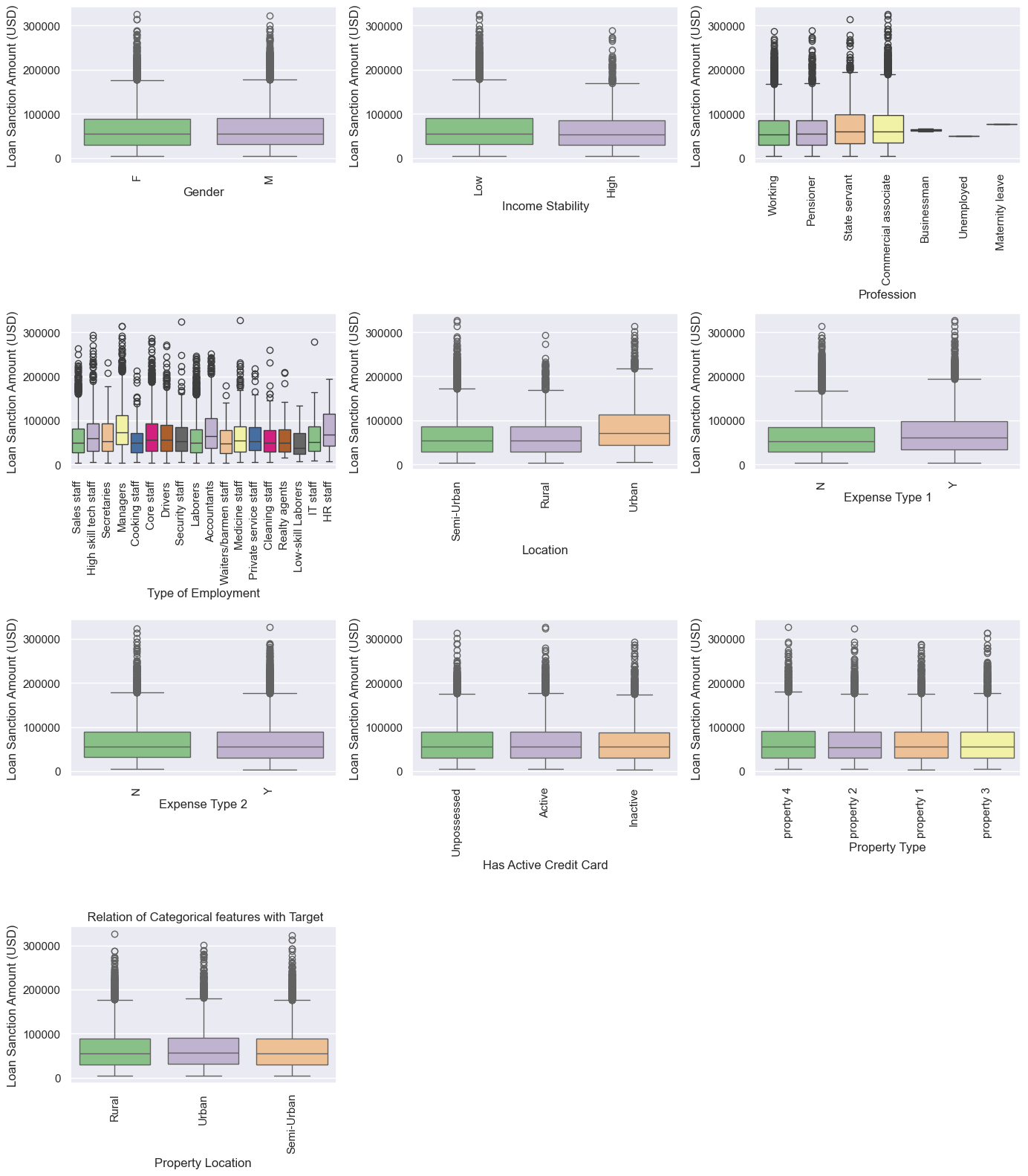
    plt.subplot(4,3, index+1)

    sns.boxplot(x=col, y='Loan Sanction Amount (USD)', data=target\_without\_zero, palette='Accent')

    plt.xticks(rotation=90)

plt.title(' Relation of Categorical features with Target')

plt.tight\_layout()



# Profession feature looks different.

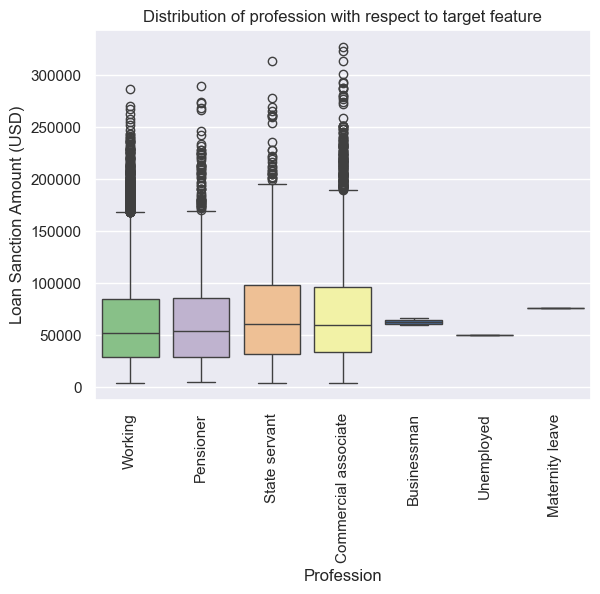
# Box-plot for Profession feature

sns.boxplot(x='Profession', y='Loan Sanction Amount (USD)', data=target\_without\_zero, palette='Accent')

plt.title('Distribution of profession with respect to target feature')

plt.xticks(rotation=90)

plt.show()



# Let's try violin plot

plt.figure(figsize=(14,16))

for index, col in enumerate(cat\_cols):

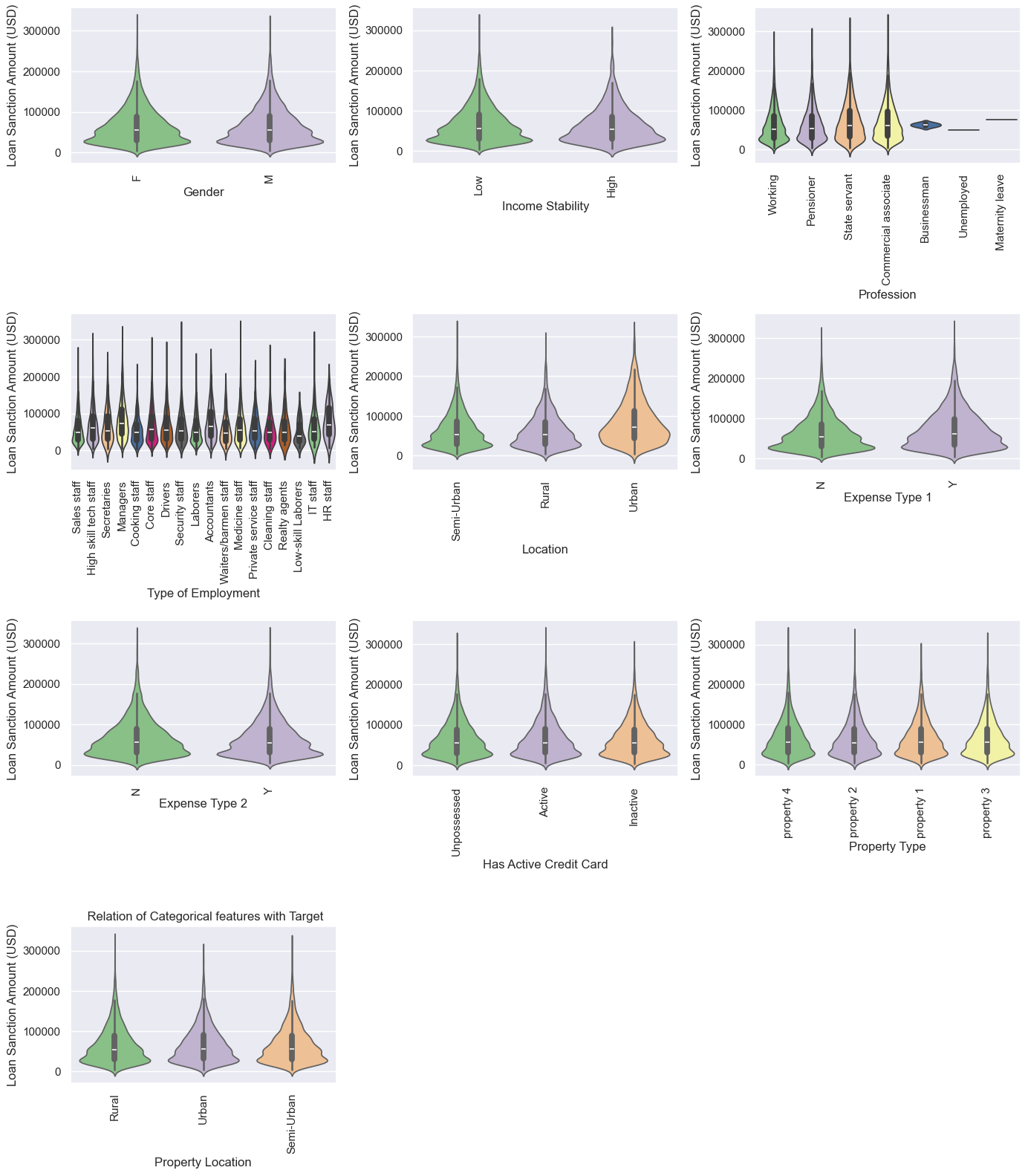
    plt.subplot(4,3, index+1)

    sns.violinplot(x=col, y='Loan Sanction Amount (USD)', data=target\_without\_zero, palette='Accent')

    plt.xticks(rotation=90)

plt.title(' Relation of Categorical features with Target')

plt.tight\_layout()



# Relatiob between categorical and loan sanctioned feature

plt.figure(figsize=(12,14))

for index, col in enumerate(cat\_cols):

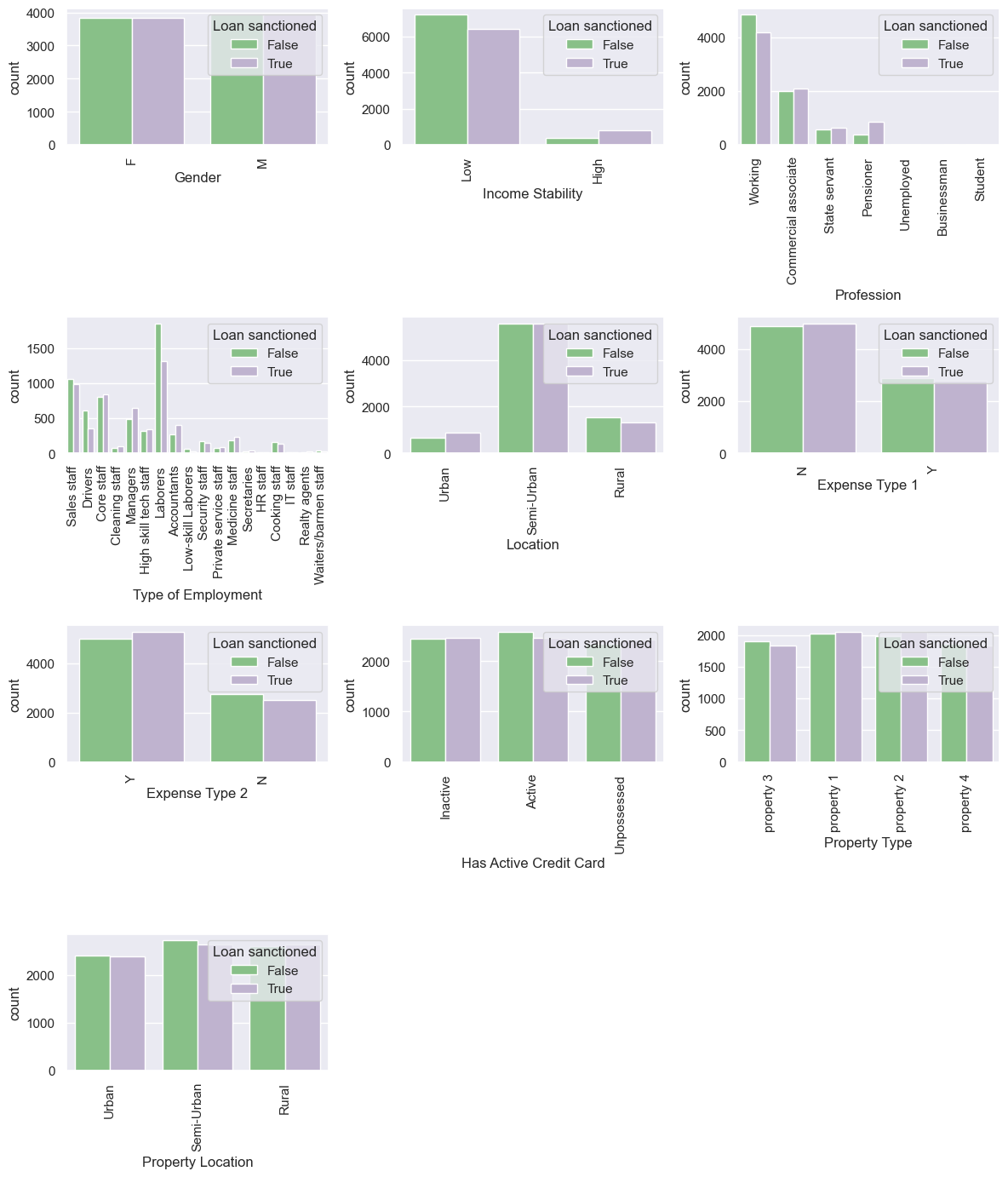
    plt.subplot(4,3, index+1)

    sns.countplot(x=col, hue='Loan sanctioned', data=train\_clf\_sample, palette='Accent')

    plt.xticks(rotation=90)

print('Distribution of features with loan sanctioned')

plt.tight\_layout()



# Relation between continuous features with loan sanctioned

plt.figure(figsize=(13,10))

for index, col in enumerate(con\_cols):

    if col == 'Loan Sanction Amount (USD)':

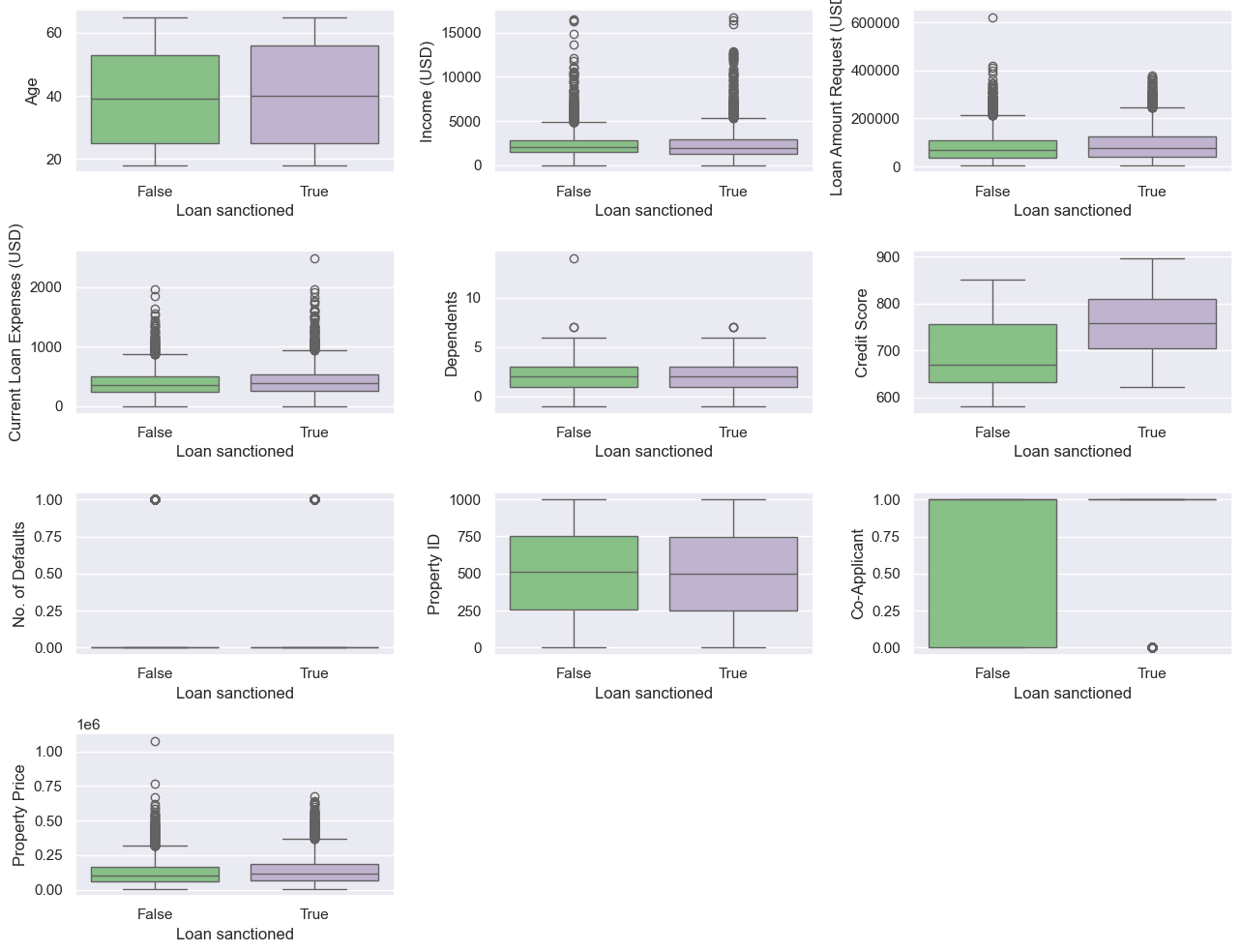
        continue

    plt.subplot(4,3, index+1)

    sns.boxplot(x='Loan sanctioned', y=col, data=train\_clf\_sample, palette='Accent')

print('Distribution of Continuous features with loan sanctioned')

plt.tight\_layout()



# Using violin plot for more clearer view

plt.figure(figsize=(13,10))

for index, col in enumerate(con\_cols):

    if col == 'Loan Sanction Amount (USD)':

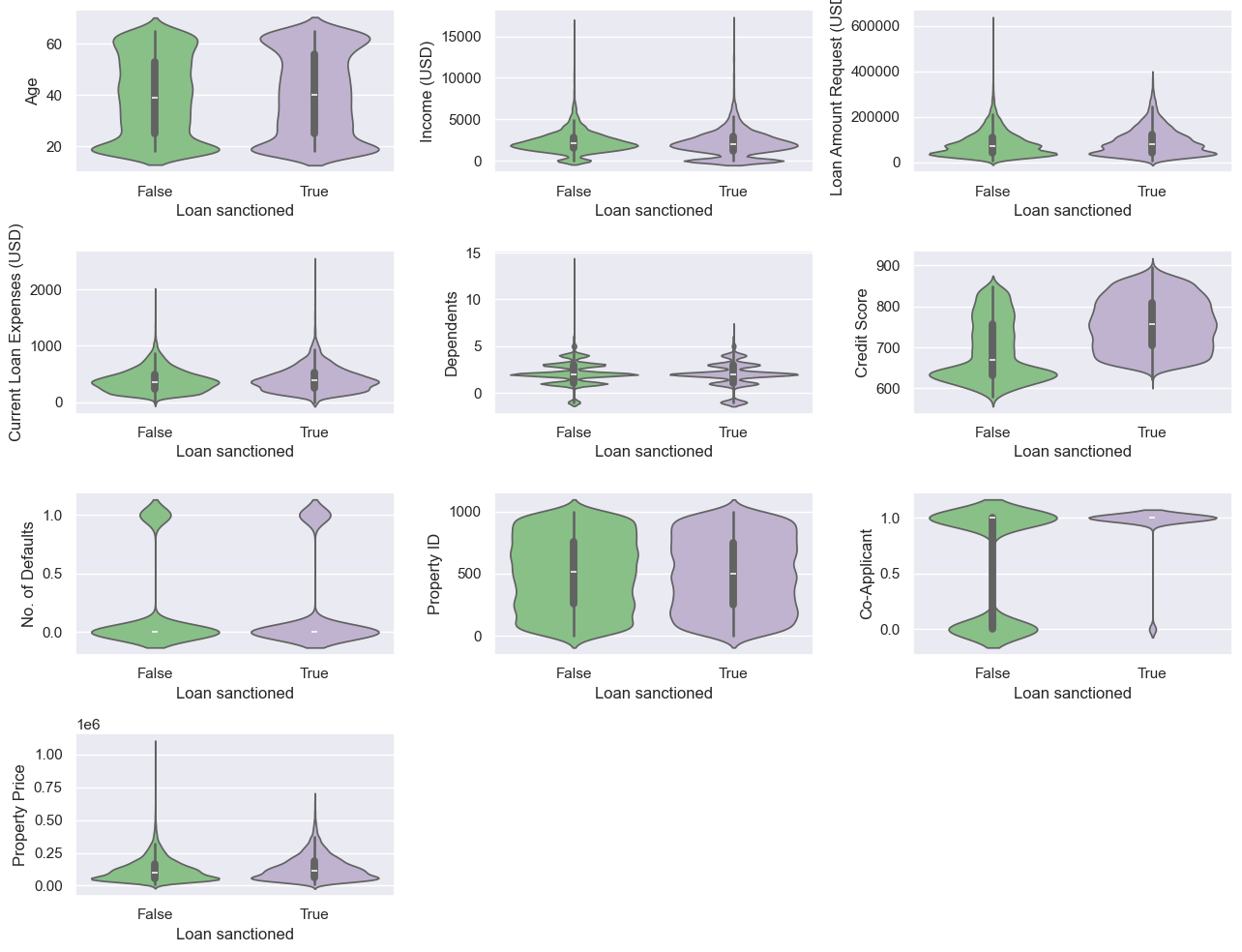
        continue

    plt.subplot(4,3, index+1)

    sns.violinplot(x='Loan sanctioned', y=col, data=train\_clf\_sample, palette='Accent')

print('Distribution of Continuous features with loan sanctioned')

plt.tight\_layout()



reg\_exp = train\_set[train\_set['Loan Sanction Amount (USD)'] > 0]

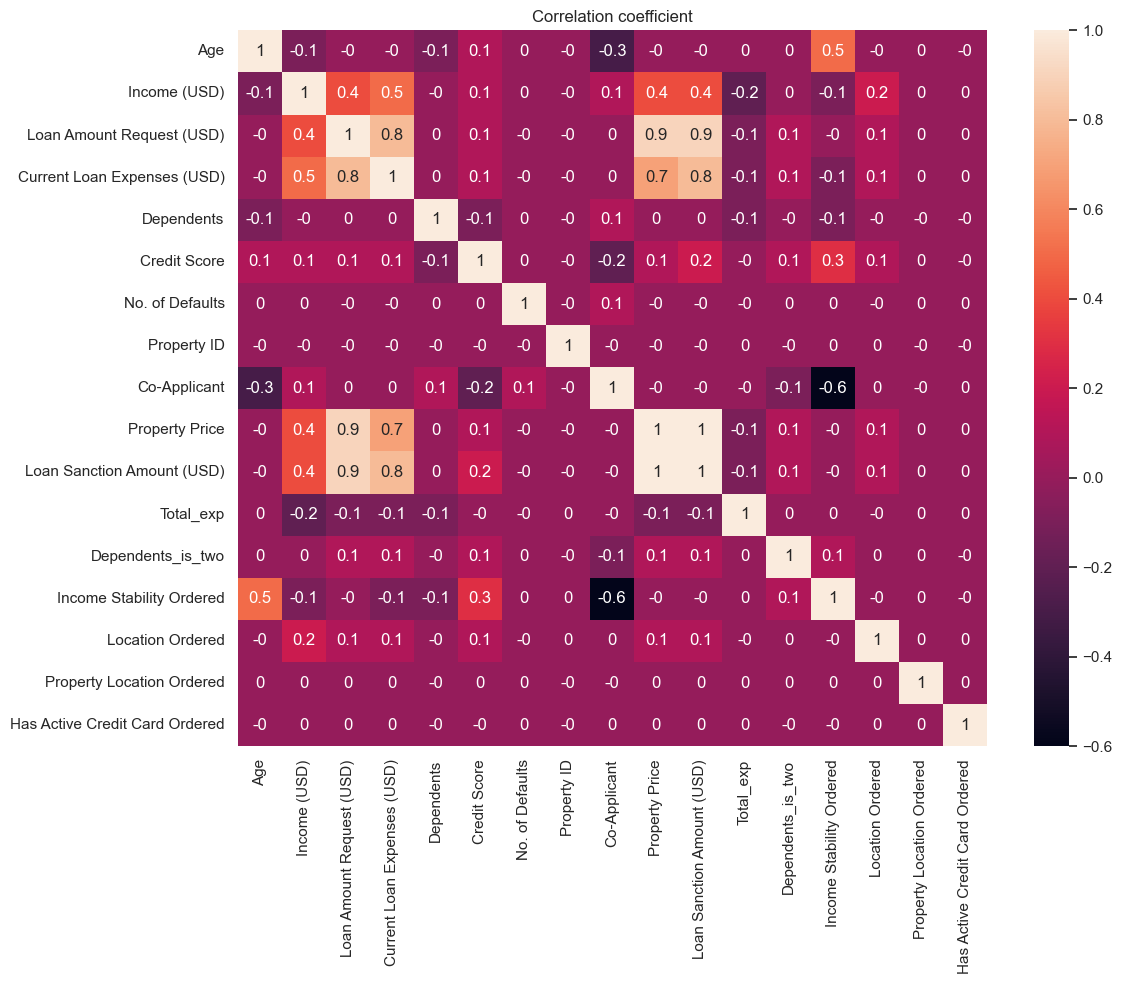
# corr = np.round(reg\_exp.corr(), 1)

plt.figure(figsize=(12,10))

sns.heatmap(corr, annot=True)

plt.title('Correlation coefficient')

plt.tight\_layout()



# Load your data

data = pd.read\_csv('train.csv')

# Remove rows with NaN in the target variable

data = data.dropna(subset=['Loan Sanction Amount (USD)'])

# Split the data into features and target

X = data.drop('Loan Sanction Amount (USD)', axis=1)

y = data['Loan Sanction Amount (USD)']

# Preprocessing

numeric\_features = ['Age', 'Income (USD)', 'Loan Amount Request (USD)', 'Current Loan Expenses (USD)',

                    'Dependents', 'Credit Score', 'Property Age', 'Property Price']

numeric\_transformer = Pipeline(steps=[

    ('imputer', SimpleImputer(strategy='median')),

    ('scaler', StandardScaler())])

categorical\_features = ['Gender', 'Income Stability', 'Profession', 'Type of Employment', 'Location',

                        'Expense Type 1', 'Expense Type 2', 'Has Active Credit Card', 'Property Location']

categorical\_transformer = Pipeline(steps=[

    ('imputer', SimpleImputer(strategy='most\_frequent')),

    ('onehot', OneHotEncoder(handle\_unknown='ignore'))])

preprocessor = ColumnTransformer(

    transformers=[

        ('num', numeric\_transformer, numeric\_features),

        ('cat', categorical\_transformer, categorical\_features)])

# Append regressor to preprocessing pipeline

pipeline = Pipeline(steps=[('preprocessor', preprocessor),

                           ('regressor', DecisionTreeRegressor())])

# Fit the pipeline

pipeline.fit(X, y)

# Save the fitted pipeline

joblib.dump(pipeline, 'pipeline.pkl')

**form.HTML:**

 <!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Loan Prediction Form</title>

    <link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

</head>

<body>

    <div class="container">

        <h1>Loan Prediction Form</h1>

        <form action="/predict" method="post">

            <label for="Gender">Gender:</label>

            <input type="text" id="Gender" name="Gender">

            <label for="Age">Age:</label>

            <input type="number" id="Age" name="Age">

            <label for="Income (USD)">Income (USD):</label>

            <input type="number" id="Income (USD)" name="Income (USD)" step="0.01">

            <label for="Income Stability">Income Stability:</label>

            <input type="text" id="Income Stability" name="Income Stability">

            <label for="Profession">Profession:</label>

            <input type="text" id="Profession" name="Profession">

            <label for="Type of Employment">Type of Employment:</label>

            <input type="text" id="Type of Employment" name="Type of Employment">

            <label for="Location">Location:</label>

            <input type="text" id="Location" name="Location">

            <label for="Loan Amount Request (USD)">Loan Amount Request (USD):</label>

            <input type="number" id="Loan Amount Request (USD)" name="Loan Amount Request (USD)" step="0.01">

            <label for="Current Loan Expenses (USD)">Current Loan Expenses (USD):</label>

            <input type="number" id="Current Loan Expenses (USD)" name="Current Loan Expenses (USD)" step="0.01">

            <label for="Expense Type 1">Expense Type 1:</label>

            <input type="text" id="Expense Type 1" name="Expense Type 1">

            <label for="Expense Type 2">Expense Type 2:</label>

            <input type="text" id="Expense Type 2" name="Expense Type 2">

            <label for="Dependents">Dependents:</label>

            <input type="number" id="Dependents" name="Dependents">

            <label for="Credit Score">Credit Score:</label>

            <input type="number" id="Credit Score" name="Credit Score" step="0.01">

            <label for="No. of Defaults">No. of Defaults:</label>

            <input type="number" id="No. of Defaults" name="No. of Defaults">

            <label for="Has Active Credit Card">Has Active Credit Card:</label>

            <input type="text" id="Has Active Credit Card" name="Has Active Credit Card">

            <label for="Property ID">Property ID:</label>

            <input type="number" id="Property ID" name="Property ID">

            <label for="Property Age">Property Age:</label>

            <input type="number" id="Property Age" name="Property Age" step="0.01">

            <label for="Property Type">Property Type:</label>

            <input type="number" id="Property Type" name="Property Type">

            <label for="Property Location">Property Location:</label>

            <input type="text" id="Property Location" name="Property Location">

            <label for="Co-Applicant">Co-Applicant:</label>

            <input type="number" id="Co-Applicant" name="Co-Applicant">

            <label for="Property Price">Property Price:</label>

            <input type="number" id="Property Price" name="Property Price" step="0.01">

            <input type="submit" value="Submit">

        </form>

    </div>

</body>

</html>

**Result.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Loan Prediction Result</title>

    <link rel="stylesheet" href="{{ url\_for('static', filename='styles.css') }}">

</head>

<body>

    <div class="container">

        <h1>Loan Prediction Result</h1>

        <p>The loan sanction amount is: {{ prediction }}</p>

    </div>

</body>

</html>

**App.py:**

from flask import Flask, request, render\_template

import joblib

import pandas as pd

import numpy as np

app = Flask(\_\_name\_\_)

# Load the pre-trained pipeline which includes the preprocessor and the regressor

pipeline = joblib.load('pipeline.pkl')

@app.route('/')

def home():

    return render\_template('form.html')

@app.route('/predict', methods=['POST'])

def predict():

    # Get form data

    form\_data = request.form.to\_dict()

    # Convert form data to DataFrame

    input\_data = pd.DataFrame([form\_data])

    # Cast numeric columns to float

    numeric\_columns = ['Age', 'Income (USD)', 'Loan Amount Request (USD)', 'Current Loan Expenses (USD)',

                       'Dependents', 'Credit Score', 'Property Age', 'Property Price']

    for column in numeric\_columns:

        input\_data[column] = pd.to\_numeric(input\_data[column], errors='coerce')

    # Print the form data and preprocessed data for debugging

    print("Form Data:", form\_data)

    print("Input Data for Prediction:", input\_data)

    # Use the pipeline to preprocess and predict

    try:

        loan\_eligibility = pipeline.predict(input\_data)[0]

        print("Prediction:", loan\_eligibility)

        return render\_template('result.html', prediction=loan\_eligibility)

    except ValueError as e:

        return str(e)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

**10.2 GitHub and project Demo link:**

Github link: **https://github.com/HEVANTH8/**

Project Demo link: [Student Dashboard (smartinternz.com)](https://smartinternz.com/Student/guided_project_workspace/723405)