Exploratory Data Analysis (EDA)

LOAN DATASET

Anushka Jain

12322239

AIDE - Engineer

LOVELY PROFESSIONAL UNIVERSITY

FUTURENSE TECHNOLOGY

Exploratory Data Analysis Project using Python

LOAN DATASET

Introduction

This report presents the findings from the Exploratory Data Analysis (EDA) performed on the loan dataset. The analysis aims to uncover patterns, relationships, and insights that can help in understanding the data better. The steps involved in the analysis include data loading and cleaning, univariate analysis, bivariate analysis, and multivariate analysis.

Dataset Overview

The dataset used for this analysis is 'loan `2.csv'`. It contains various features related to loans, such as loan amount, interest rate, borrower details, and more. In this dataset, we used Pandas, Numpy, matplotlib, seaborn, and opendatasets libraries.

Here's the step-by-step outline of the project :

- Download the dataset.
- Data Preparation and Cleaning.
- Exploratory Analysis and Visualizations.
- Summary and Conclusion.

Installing Important Libraries

by using “% pip install library_name” command we can download the libraries.

```
# install imp. libraries
%pip install pandas
%pip install matplotlib
%pip install seaborn
%pip install sklearn
%pip install numpy
```

• Pandas library :

pandas is an open-source library built on top of numpy providing high performance, easy to use data structures and data analysis tools for python. It allows for fast analysis and data cleaning and preparation.

• Numpy library:

NumPy is a library for Python that adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

• Matplotlib library :

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

• Seaborn library :

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Importing Important Libraries

```
# importing libraries
import pandas as pd    Import "pandas" could not be resolved from source
import numpy as np    Import "numpy" could not be resolved
import matplotlib.pyplot as plt    Import "matplotlib.pyplot" could not be resolved from source
import seaborn as sns    Import "seaborn" could not be resolved from source
from scipy.stats import *    Import "scipy.stats" could not be resolved
from sklearn.decomposition import PCA    Import "sklearn.decomposition" could not be resolved from source
import scipy.stats    Import "scipy.stats" could not be resolved
from scipy.stats import chi2_contingency, pearsonr    Import "scipy.stats" could not be resolved
```

Data Loading and Initial Exploration

The dataset was loaded into a pandas DataFrame. Initial exploration included checking for null values, basic statistical summaries, and an overview of the data types for each column.

Handling Missing Values

Missing values were identified and handled appropriately. Various techniques were used, such as filling missing values with the mean for numerical columns.

Handling Outliers

Outliers were detected and handled using the Interquartile Range (IQR) method. This step ensures that the data is clean and ready for further analysis.

```
import pandas as pd    Import "pandas" could not be resolved from source

# Load the data
file_path = 'Loan 2.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the data
print(data.head())
```

```
  Customer ID              Name Gender  Age  Income (USD) Income Stability  \
0     C-36995   Frederica Shealy      F   56       1933.05              Low
1     C-33999  America Calderone      M   32       4952.91              Low
2      C-3770      Rosetta Verne      F   65        988.19             High
3     C-26480         Zoe Chitty      F   65           NaN             High
4     C-23459       Afton Venema      F   31       2614.77              Low

  Profession     Type of Employment    Location  Loan Amount Request (USD)  \
0    Working            Sales staff  Semi-Urban                   72809.58
1    Working                    NaN  Semi-Urban                   46837.47
2  Pensioner                    NaN  Semi-Urban                   45593.04
3  Pensioner                    NaN       Rural                   80057.92
4    Working  High skill tech staff  Semi-Urban                  113858.89

   ...  Credit Score No. of Defaults Has Active Credit Card  Property ID  \
0  ...        809.44               0                    NaN          746
1  ...        780.40               0           Unpossessed          608
2  ...        833.15               0           Unpossessed          546
3  ...        832.70               1           Unpossessed          890
4  ...        745.55               1                Active          715

   Property Age  Property Type Property Location  Co-Applicant  \
```

```
# Get basic information about the data
print(data.info())

# Get statistical summary of the data
print(data.describe())

# Check for null values
print(data.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 24 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Customer ID                  30000 non-null  object
 1   Name                         30000 non-null  object
 2   Gender                       29947 non-null  object
 3   Age                          30000 non-null  int64
 4   Income (USD)                 25424 non-null  float64
 5   Income Stability             28317 non-null  object
 6   Profession                   30000 non-null  object
 7   Type of Employment           22730 non-null  object
 8   Location                     30000 non-null  object
 9   Loan Amount Request (USD)    30000 non-null  float64
 10  Current Loan Expenses (USD)  29828 non-null  float64
 11  Expense Type 1               30000 non-null  object
 12  Expense Type 2               30000 non-null  object
 13  Dependents                   27507 non-null  float64
 14  Credit Score                 28297 non-null  float64
 15  No. of Defaults              30000 non-null  int64
 16  Has Active Credit Card       28434 non-null  object
 17  Property ID                  30000 non-null  int64
 18  Property Age                 25150 non-null  float64
 19  Property Type                30000 non-null  int64
...
Co-Applicant                      0
Property Price                    0
Loan Sanction Amount (USD)      340
dtype: int64
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```python
# Fill null values for numerical columns with median
for column in data.select_dtypes(include=['float64', 'int64']).columns:    "dtypes": Unknown word.
    data[column].fillna(data[column].median(), inplace=True)    "fillna": Unknown word.

# Fill null values for categorical columns with mode
for column in data.select_dtypes(include=['object']).columns:    "dtypes": Unknown word.
    data[column].fillna(data[column].mode()[0], inplace=True)    "fillna": Unknown word.

# Verify null values are handled
print(data.isnull().sum())
```

```
Customer ID                     0
Name                            0
Gender                          0
Age                             0
Income (USD)                    0
Income Stability                0
Profession                      0
Type of Employment              0
Location                        0
Loan Amount Request (USD)       0
Current Loan Expenses (USD)     0
Expense Type 1                  0
Expense Type 2                  0
Dependents                      0
Credit Score                    0
No. of Defaults                 0
Has Active Credit Card          0
Property ID                     0
Property Age                    0
Property Type                   0
Property Location               0
Co-Applicant                    0
Property Price                  0
Loan Sanction Amount (USD)      0
dtype: int64
```

```python
# Function to remove outliers using IQR
def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)    "quantile": Unknown word.
    Q3 = df[column].quantile(0.75)    "quantile": Unknown word.
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Apply the function to each numerical column
for column in data.select_dtypes(include=['float64', 'int64']).columns:    "dtypes": Unknown word.
    data = remove_outliers_iqr(data, column)

# Display the data after removing outliers
print(data.describe())
```

```
                Age  Income (USD)  Loan Amount Request (USD)  \
count  16968.000000  16968.000000                16968.000000
mean      39.127416   2162.684262                76320.368761
std       15.688570    697.519946                44747.602671
min       18.000000    377.700000                 6108.050000
25%       24.000000   1670.475000                39116.837500
50%       39.000000   2222.435000                68607.700000
75%       53.000000   2515.915000               105365.145000
max       65.000000   4378.180000               228220.460000

       Current Loan Expenses (USD)    Dependents  Credit Score  \
count                 16968.000000  16968.000000  16968.000000
mean                    365.290167      2.206742    736.771743
std                     165.428449      0.853187     67.289045
min                      33.760000      1.000000    580.850000
25%                     236.337500      2.000000    684.887500
50%                     352.540000      2.000000    739.820000
75%                     467.462500      3.000000    788.222500
max                     860.380000      4.000000    890.020000

       No. of Defaults   Property ID  Property Age  Property Type  \
count          16968.0  16968.000000  16968.000000   16968.000000
mean               0.0    502.008545   2162.960506       2.452735
std                0.0    287.087116    692.549032       1.119334
min                0.0      1.000000    377.700000       1.000000
...
25%                1.0     56848.710000                15411.222500
50%                1.0     98803.565000                35941.010000
75%                1.0    155474.310000                67754.017500
max                1.0    317198.700000               148035.580000
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```python
# Fill null values for numerical columns with median
for column in data.select_dtypes(include=['float64', 'int64']).columns:    "dtypes": Unknown word.
    data[column].fillna(data[column].median(), inplace=True)    "fillna": Unknown word.

# Fill null values for categorical columns with mode
for column in data.select_dtypes(include=['object']).columns:    "dtypes": Unknown word.
    data[column].fillna(data[column].mode()[0], inplace=True)    "fillna": Unknown word.

# Verify null values are handled
null_values_after = data.isnull().sum()
null_values_after
```

```
Customer ID                       0
Name                              0
Gender                            0
Age                               0
Income (USD)                      0
Income Stability                  0
Profession                        0
Type of Employment                0
Location                          0
Loan Amount Request (USD)         0
Current Loan Expenses (USD)       0
Expense Type 1                    0
Expense Type 2                    0
Dependents                        0
Credit Score                      0
No. of Defaults                   0
Has Active Credit Card            0
Property ID                       0
Property Age                      0
Property Type                     0
Property Location                 0
Co-Applicant                      0
Property Price                    0
Loan Sanction Amount (USD)        0
dtype: int64
```

```python
# Function to remove outliers using IQR
def remove_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)    "quantile": Unknown word.
    Q3 = df[column].quantile(0.75)    "quantile": Unknown word.
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]

# Apply the function to each numerical column
for column in data.select_dtypes(include=['float64', 'int64']).columns:    "dtypes": Unknown word.
    data = remove_outliers_iqr(data, column)

# Display the data after removing outliers
data_after_outliers = data.describe()
data_after_outliers
```

	Age	Income (USD)	Loan Amount Request (USD)	Current Loan Expenses (USD)	Dependents	Credit Score	No. of Defaults	Property ID	Property Age	Property Type	Co-Applicant	Property Price	Loan Sanction Amount (USD)
count	15890.000000	15890.000000	15890.000000	15890.000000	15890.000000	15890.000000	15890.0	15890.000000	15890.000000	15890.000000	15890.0	15890.000000	15890.000000
mean	39.138011	2098.318277	72465.174687	351.725171	2.204468	735.915064	0.0	501.888483	2100.010986	2.453870	1.0	106426.180276	41604.798665
std	15.690879	638.253844	41416.205829	155.085601	0.856376	67.248515	0.0	287.161871	634.762799	1.119143	0.0	63767.584119	34232.305441
min	18.000000	472.040000	6108.050000	33.760000	1.000000	580.850000	0.0	1.000000	472.040000	1.000000	1.0	-999.000000	-999.000000
25%	24.000000	1642.635000	38449.522500	229.947500	2.000000	683.925000	0.0	252.000000	1647.622500	1.000000	1.0	55208.450000	14886.685000
50%	39.000000	2220.215000	66282.645000	342.470000	2.000000	739.820000	0.0	502.000000	2223.250000	2.000000	1.0	94256.520000	35209.395000
75%	53.000000	2443.127500	99716.700000	449.947500	3.000000	787.212500	0.0	750.750000	2437.792500	3.000000	1.0	146937.782500	63991.097500
max	65.000000	3626.620000	195086.780000	787.630000	4.000000	890.020000	0.0	999.000000	3626.620000	4.000000	1.0	287329.980000	137638.120000

```
# 'data' is our processed DataFrame
output_file_path = 'processed_loan_data.csv'

# Save the DataFrame to a CSV file
data.to_csv(output_file_path, index=False)

print(f"Processed data has been saved to {output_file_path}")
```

```
Processed data has been saved to processed_loan_data.csv
```

```
# Load the data
file_path = 'processed_loan_data.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the data
print("First few rows of the dataset:")
print(data.head())
```

Univariate Analysis

It is the simplest form of data analysis where we examine each variable individually. The purpose of this analysis is to understand the distribution

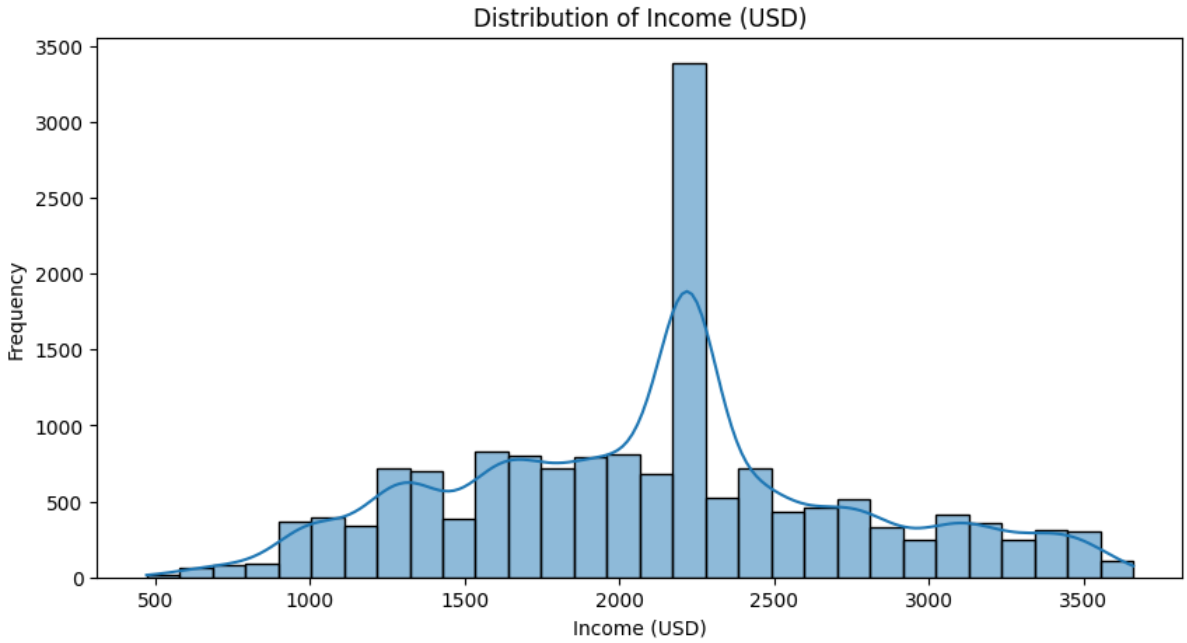
and characteristics of each feature in the dataset. It includes statistical summaries and visualizations such as histograms, box plots, and bar charts.

Features of Univariate Data Analysis:

- Central Tendency: Measures like mean, median, and mode.
- Dispersion: Measures like range, variance, and standard deviation.
- Shape: Skewness and kurtosis of the distribution.
- Visualisation: Histograms for numerical data, and bar plots for categorical data.

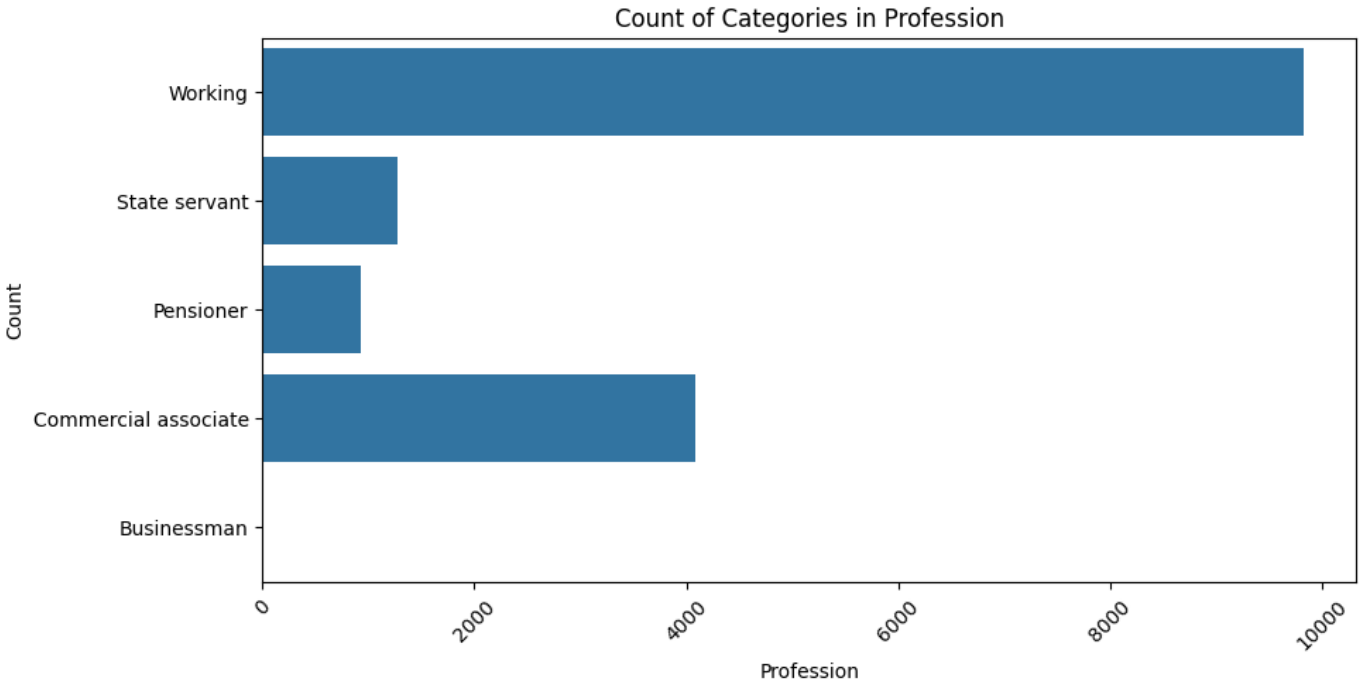
```
# Univariate Analysis for Numerical Features

numerical_columns = data.select_dtypes(include=['float64', 'int64']).columns

print("\nSummary Statistics for Numerical Features:")
for column in numerical_columns:
    print(f"\nStatistics for {column}:")
    print(f"Mean: {data[column].mean()}")
    print(f"Median: {data[column].median()}")

    # Measures of Dispersion
    print(f"Range: {data[column].max() - data[column].min()}")
    print(f"Variance: {data[column].var()}")
    print(f"Standard Deviation: {data[column].std()}")

    # Measures of Shape
    print(f"Skewness: {data[column].skew()}")
    print(f"Kurtosis: {data[column].kurtosis()}")

    # Histogram
    plt.figure(figsize=(10, 5))    "figsize": Unknown word.
    sns.histplot(data[column], kde=True, bins=30)    "histplot": Unknown word.
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)    "xlabel": Unknown word.
    plt.ylabel('Frequency')    "ylabel": Unknown word.
    plt.show()
```

```
Statistics for Income (USD):
Mean: 2106.1819553887203
Median: 2222.435
Range: 3188.03
Variance: 412714.03950779355
Standard Deviation: 642.4282368543537
Skewness: 0.16199362526661434
Kurtosis: -0.3350356751557846
```




```python
# Univariate Analysis for Categorical Features

categorical_columns = data.select_dtypes(include=['object']).columns

print("\nSummary Statistics for Categorical Features:")
for column in categorical_columns:
    print(f"\nStatistics for {column}:")
    print(data[column].value_counts())
    print(f"Mode: {data[column].mode()[0]}")

    # Bar Plot
    plt.figure(figsize=(10, 5))    "figsize": Unknown word.
    sns.countplot(data[column])    "countplot": Unknown word.
    plt.title(f'Count of Categories in {column}')
    plt.xlabel(column)    "xlabel": Unknown word.
    plt.ylabel('Count')    "ylabel": Unknown word.
    plt.xticks(rotation=45)    "xticks": Unknown word.
    plt.show()
```

```
Statistics for Profession:
Profession
Working                 9835
Commercial associate    4079
State servant           1269
Pensioner                933
Businessman                1
Name: count, dtype: int64
Mode: Working
```



Bivariate Analysis

It is a statistical method used to determine the relationship between two variables. It helps to understand how one variable is affected by changes in another.

Bivariate analysis can be applied to:

- Numerical vs. Numerical: Correlation and scatter plots.
- Numerical vs. Categorical: Box plots, violin plots, and t-tests.
- Categorical vs. Categorical: Contingency tables and chi-square tests.

Features of Bivariate Analysis:

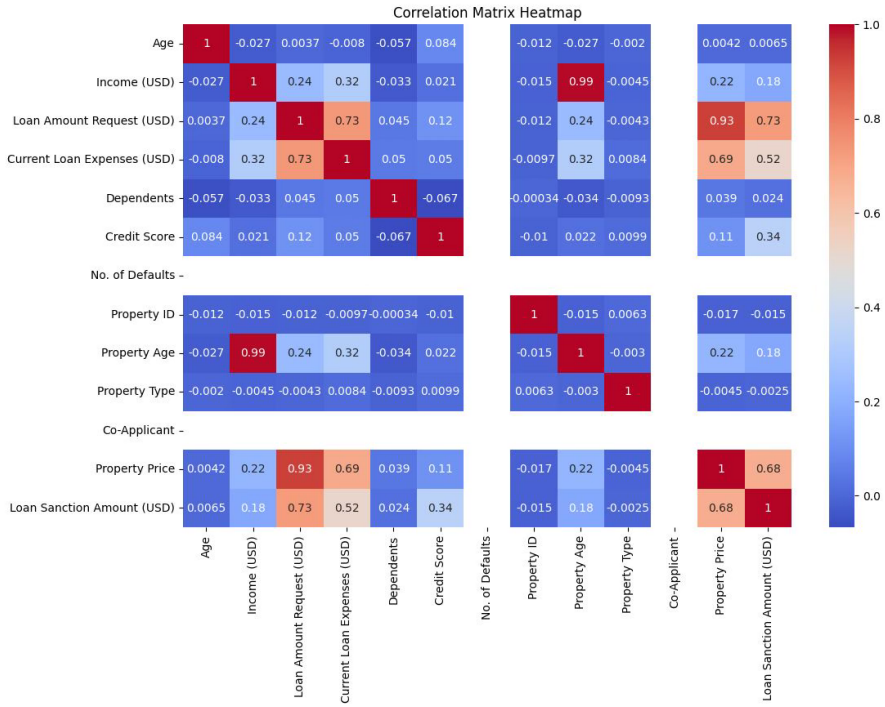
- Correlation: Measures the strength and direction of the linear relationship between two numerical variables.
- Scatter Plots: Visual representation of the relationship between two numerical variables.
- Box Plots: Shows the distribution of a numerical variable across the categories of a categorical variable.
- Violin Plots: Similar to box plots but also show the kernel density of the data.
- Contingency Tables: Display the frequency distribution of categorical variables.
- Chi-Square Tests: Test the association between two categorical variables.

```
# Numerical vs. Numerical: Correlation and Scatter Plot
numerical_columns = data.select_dtypes(include=['float64', 'int64']).columns

print("\nCorrelation between Numerical Features:")
correlation_matrix = data[numerical_columns].corr()
print(correlation_matrix)
```

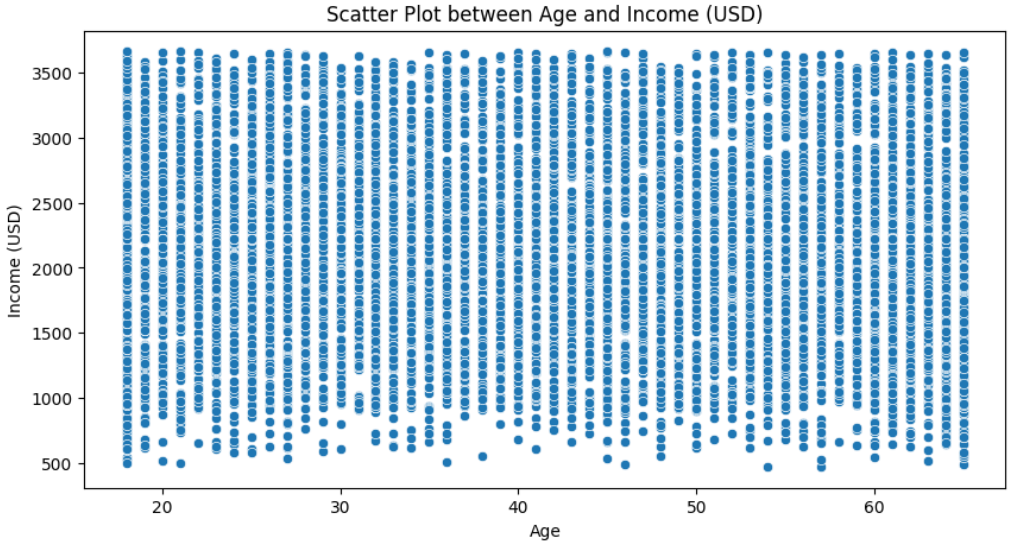
```
Correlation between Numerical Features:
                                Age  Income (USD)  \
Age                        1.000000     -0.026527
Income (USD)              -0.026527      1.000000
Loan Amount Request (USD)  0.003691      0.240544
Current Loan Expenses (USD) -0.007977      0.319590
Dependents                -0.056981     -0.033185
Credit Score               0.084496      0.021051
No. of Defaults                 NaN           NaN
Property ID               -0.012239     -0.014644
Property Age              -0.026711      0.993999
Property Type             -0.001978     -0.004515
Co-Applicant                    NaN           NaN
Property Price             0.004209      0.224053
Loan Sanction Amount (USD)  0.006529      0.176932

                           Loan Amount Request (USD)  \
Age                                         0.003691
Income (USD)                                0.240544
Loan Amount Request (USD)                   1.000000
Current Loan Expenses (USD)                 0.734165
Dependents                                  0.044845
Credit Score                                0.115891
No. of Defaults                                  NaN
...
Property Type                         -0.004485                  -0.002520
Co-Applicant                                NaN                        NaN
Property Price                        1.000000                   0.675525
Loan Sanction Amount (USD)            0.675525                   1.000000
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

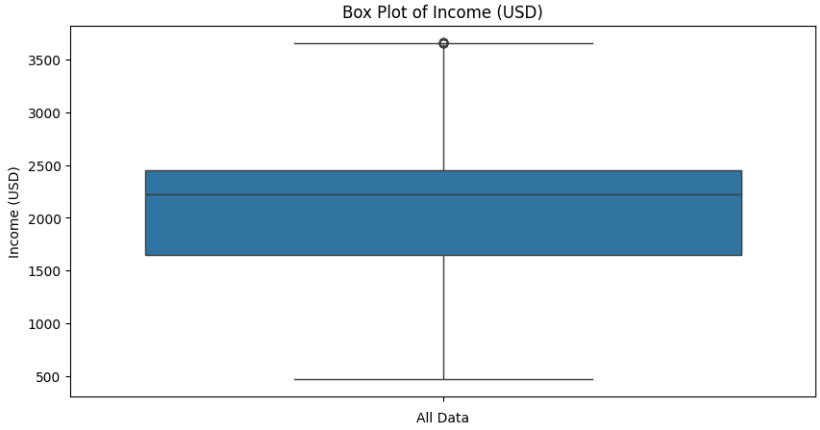
```python
# Heatmap for correlation matrix
plt.figure(figsize=(12, 8))     "figsize": Unknown word.
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Heatmap')
plt.show()
```



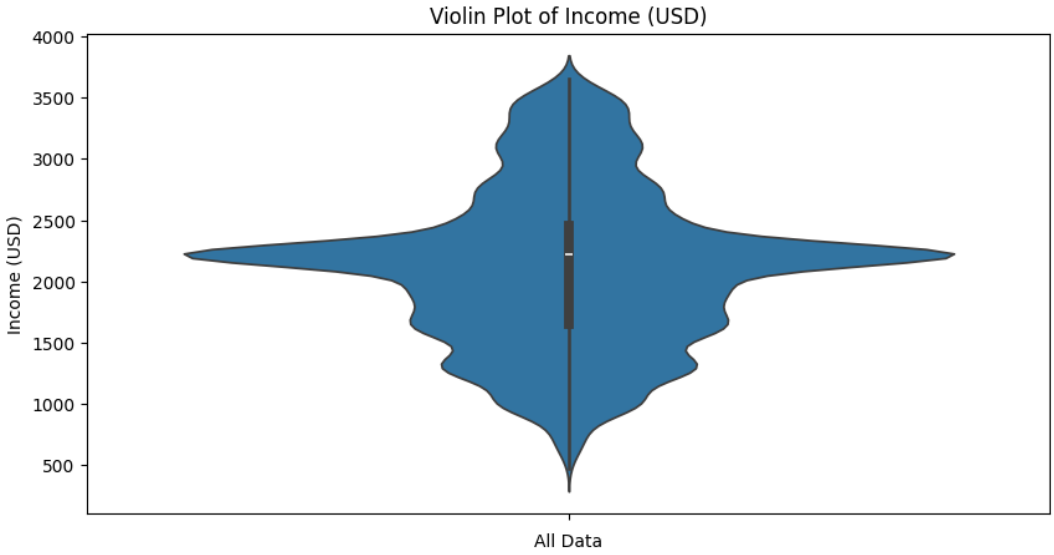
```python
# Scatter plot for each pair of numerical features
for i, column1 in enumerate(numerical_columns):
    for column2 in numerical_columns[i+1:]:
        plt.figure(figsize=(10, 5))    "figsize": Unknown word.
        sns.scatterplot(x=column1, y=column2, data=data)    "scatterplot": Unknown word.
        plt.title(f'Scatter Plot between {column1} and {column2}')
        plt.xlabel(column1)    "xlabel": Unknown word.
        plt.ylabel(column2)    "ylabel": Unknown word.
        plt.show()
```




```python
# Box Plots Numerical Data
for numerical_col in numerical_columns:
    plt.figure(figsize=(10, 5))    "figsize": Unknown word.
    sns.boxplot(y=numerical_col, data=data)    "boxplot": Unknown word.
    plt.title(f'Box Plot of {numerical_col}')
    plt.xlabel('All Data')    "xlabel": Unknown word.
    plt.ylabel(numerical_col)    "ylabel": Unknown word.
    plt.show()
```

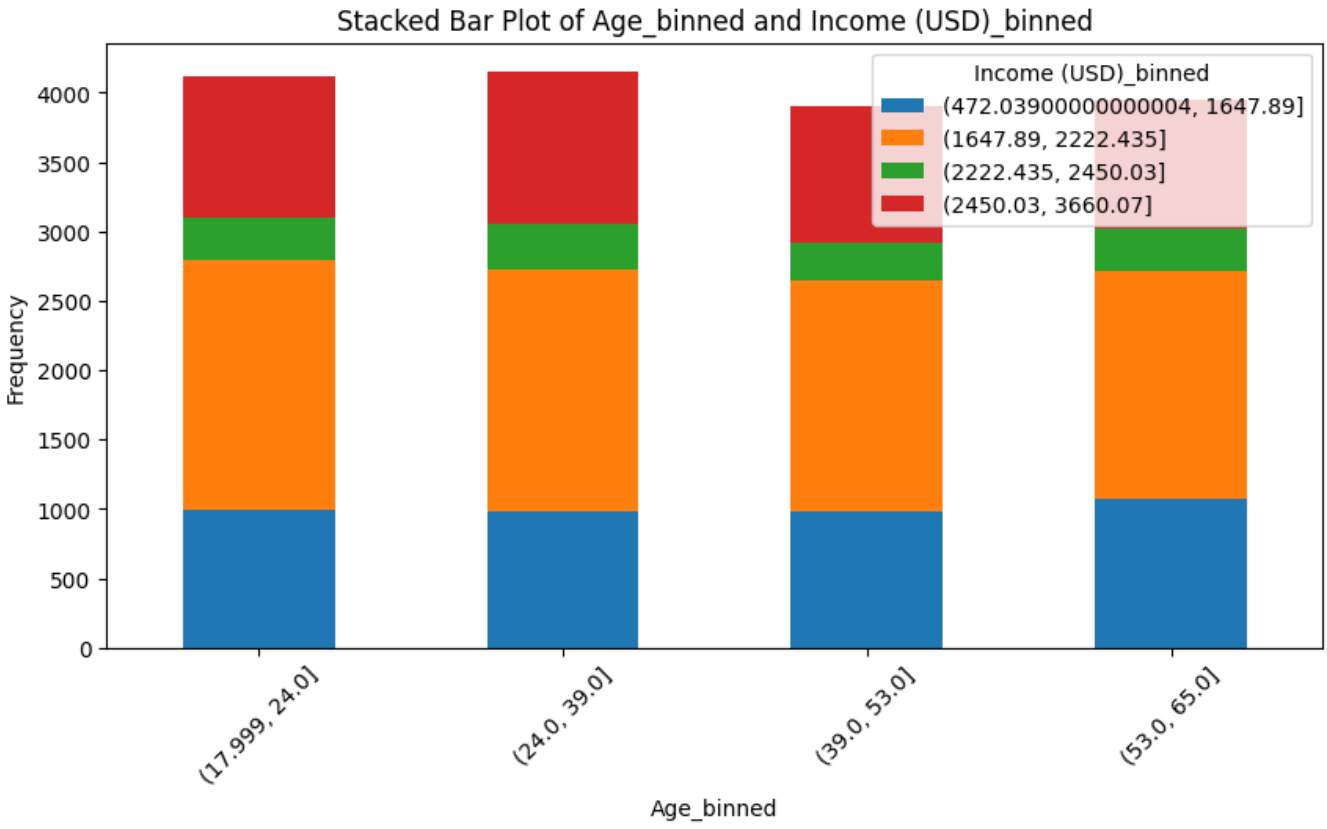


```python
# Violin Plots for Numerical Data
for numerical_col in numerical_columns:
    plt.figure(figsize=(10, 5))    "figsize": Unknown word.
    sns.violinplot(y=numerical_col, data=data)    "violinplot": Unknown word.
    plt.title(f'Violin Plot of {numerical_col}')
    plt.xlabel('All Data')    "xlabel": Unknown word.
    plt.ylabel(numerical_col)    "ylabel": Unknown word.
    plt.show()
```




```python
# Select numerical columns
numerical_columns = data.select_dtypes(include=['float64', 'int64']).columns    "dtypes": Unknown word.

# Discretize numerical data into bins
for column in numerical_columns:
    data[f'{column}_binned'] = pd.qcut(data[column], q=4, duplicates='drop')    "qcut": Unknown word.

# Contingency Tables and Chi-Square Tests
binned_columns = [f'{col}_binned' for col in numerical_columns]

print("\nChi-Square Test Results:")
for i, column1 in enumerate(binned_columns):
    for column2 in binned_columns[i+1:]:
        contingency_table = pd.crosstab(data[column1], data[column2])    "crosstab": Unknown word.
        if contingency_table.size == 0:
            print(f"\nContingency Table between {column1} and {column2} is empty. Skipping chi-square test.")
            continue
        print(f"\nContingency Table between {column1} and {column2}:")
        print(contingency_table)

        chi2, p, dof, ex = chi2_contingency(contingency_table)
        print(f"Chi-Square Test between {column1} and {column2}:")
        print(f"Chi2 Statistic: {chi2}, p-value: {p}")

        # Bar Plot for contingency table
        contingency_table.plot(kind='bar', stacked=True, figsize=(10, 5))    "figsize": Unknown word.
        plt.title(f'Stacked Bar Plot of {column1} and {column2}')
        plt.xlabel(column1)    "xlabel": Unknown word.
        plt.ylabel('Frequency')    "ylabel": Unknown word.
        plt.xticks(rotation=45)    "xticks": Unknown word.
        plt.show()
```

```
Chi-Square Test Results:

Contingency Table between Age_binned and Income (USD)_binned:
Income (USD)_binned  (472.03900000000004, 1647.89]  (1647.89, 2222.435]  \
Age_binned
(17.999, 24.0]                                998                1793
(24.0, 39.0]                                  977                1744
(39.0, 53.0]                                  980                1664
(53.0, 65.0]                                 1075                1637

Income (USD)_binned  (2222.435, 2450.03]  (2450.03, 3660.07]
Age_binned
(17.999, 24.0]                       309                1015
(24.0, 39.0]                         328                1102
(39.0, 53.0]                         278                 979
(53.0, 65.0]                         305                 933
Chi-Square Test between Age_binned and Income (USD)_binned:
Chi2 Statistic: 23.57985070997467, p-value: 0.005017390685121089
```



Multivariate Analysis

It involves the observation and analysis of more than one statistical outcome variable at a time. In simple terms, it deals with the analysis of more than one variable to understand the effect and relationship between them.

Multivariate analysis is used for:

- Understanding Relationships: It helps in understanding the relationships between different variables in the dataset.
- Pattern Recognition: It identifies patterns and structures in data which are not apparent in univariate or bivariate analysis.
- Predictive Modeling: It is used in predictive modeling to predict the outcome of one variable based on multiple other variables.
- Dimensionality Reduction: Techniques like PCA (Principal Component Analysis) are used to reduce the dimensionality of the data while retaining as much variance as possible.

Features of Multivariate Analysis:

- Correlation Matrix: Shows the correlation coefficients between pairs of variables.
- Scatter Plot Matrix: A grid of scatter plots that show relationships between pairs of variables.
- Heatmap: Visual representation of the correlation matrix.
- Principal Component Analysis (PCA): Reduces the dimensionality of the data while retaining most of the variance.
- Pair Plot: Pairwise relationships in a dataset.

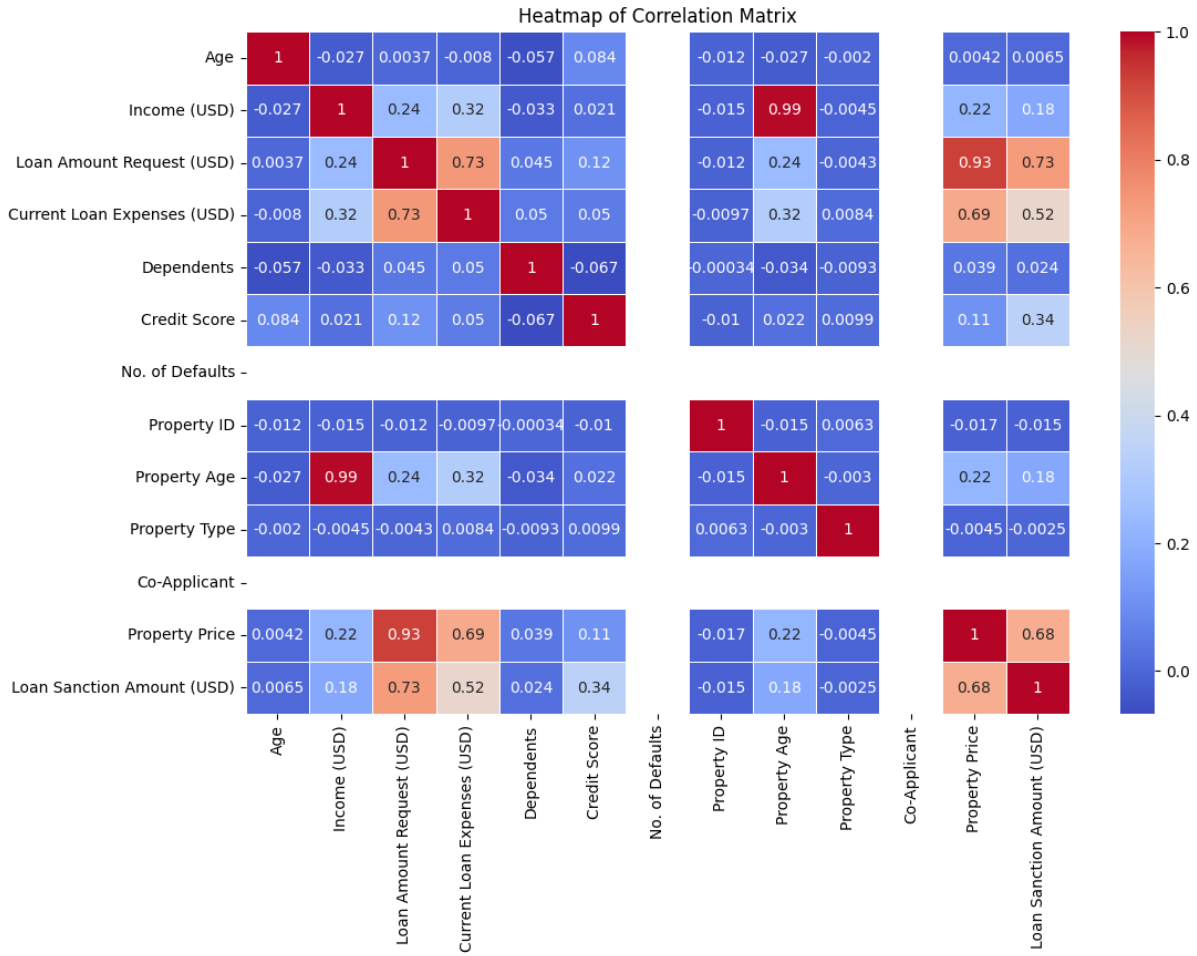
```
#  Select numerical columns
numerical_columns = data.select_dtypes(include=['float64', 'int64']).columns
```

```python
# Correlation Matrix
correlation_matrix = data[numerical_columns].corr()
print("\nCorrelation Matrix:")
print(correlation_matrix)
```

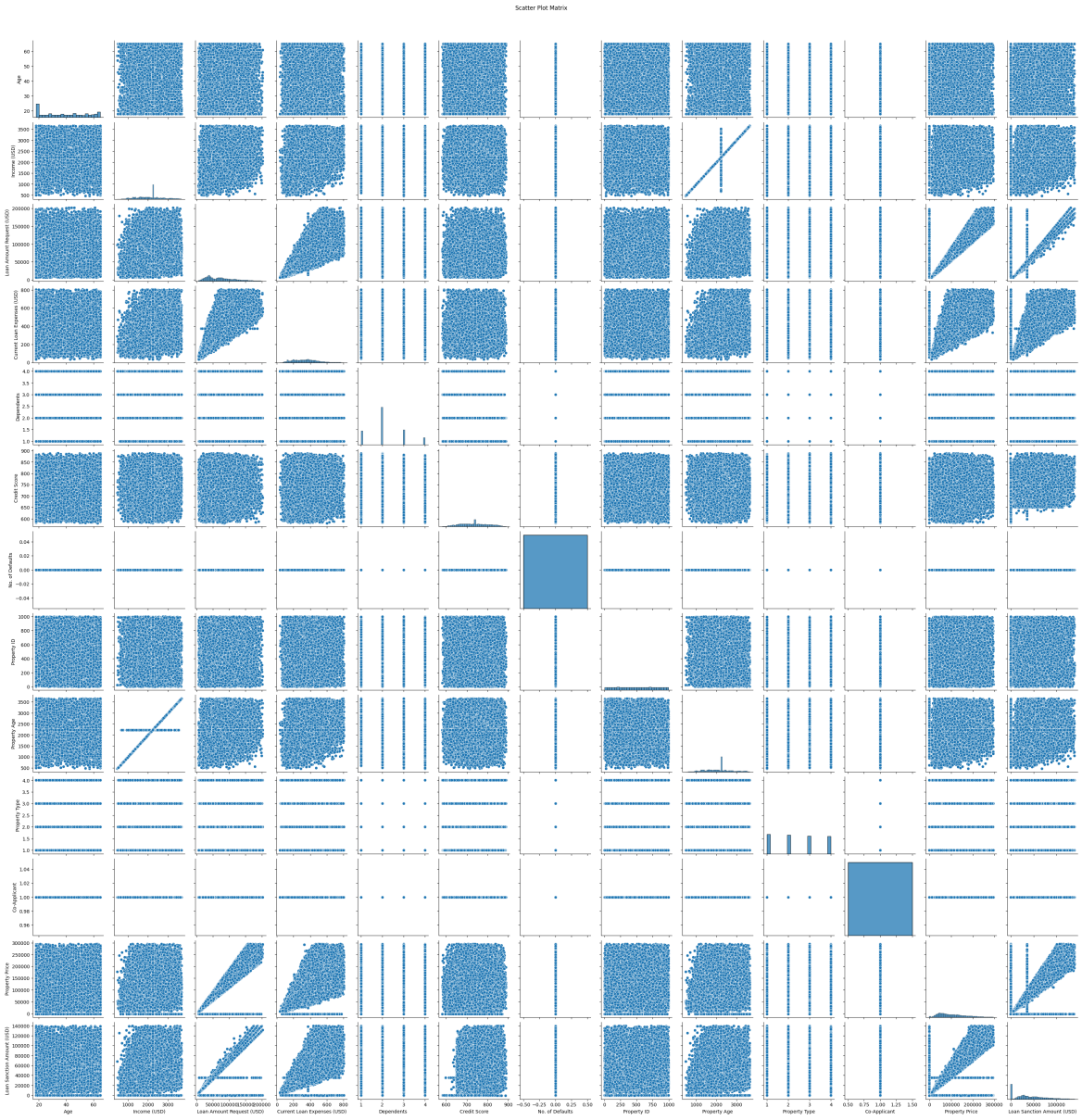
```
Correlation Matrix:
                                Age  Income (USD)  \
Age                        1.000000     -0.026527
Income (USD)              -0.026527      1.000000
Loan Amount Request (USD)  0.003691      0.240544
Current Loan Expenses (USD) -0.007977      0.319590
Dependents                -0.056981     -0.033185
Credit Score               0.084496      0.021051
No. of Defaults                 NaN           NaN
Property ID               -0.012239     -0.014644
Property Age              -0.026711      0.993999
Property Type             -0.001978     -0.004515
Co-Applicant                    NaN           NaN
Property Price             0.004209      0.224053
Loan Sanction Amount (USD)  0.006529      0.176932

                           Loan Amount Request (USD)  \
Age                                         0.003691
Income (USD)                                0.240544
Loan Amount Request (USD)                   1.000000
Current Loan Expenses (USD)                 0.734165
Dependents                                  0.044845
Credit Score                                0.115891
No. of Defaults                                  NaN
...
Property Type                         -0.004485                  -0.002520
Co-Applicant                                NaN                        NaN
Property Price                         1.000000                   0.675525
Loan Sanction Amount (USD)             0.675525                   1.000000
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
# Heatmap of the Correlation Matrix
plt.figure(figsize=(12, 8))    "figsize": Unknown word.
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Heatmap of Correlation Matrix')
plt.show()
```

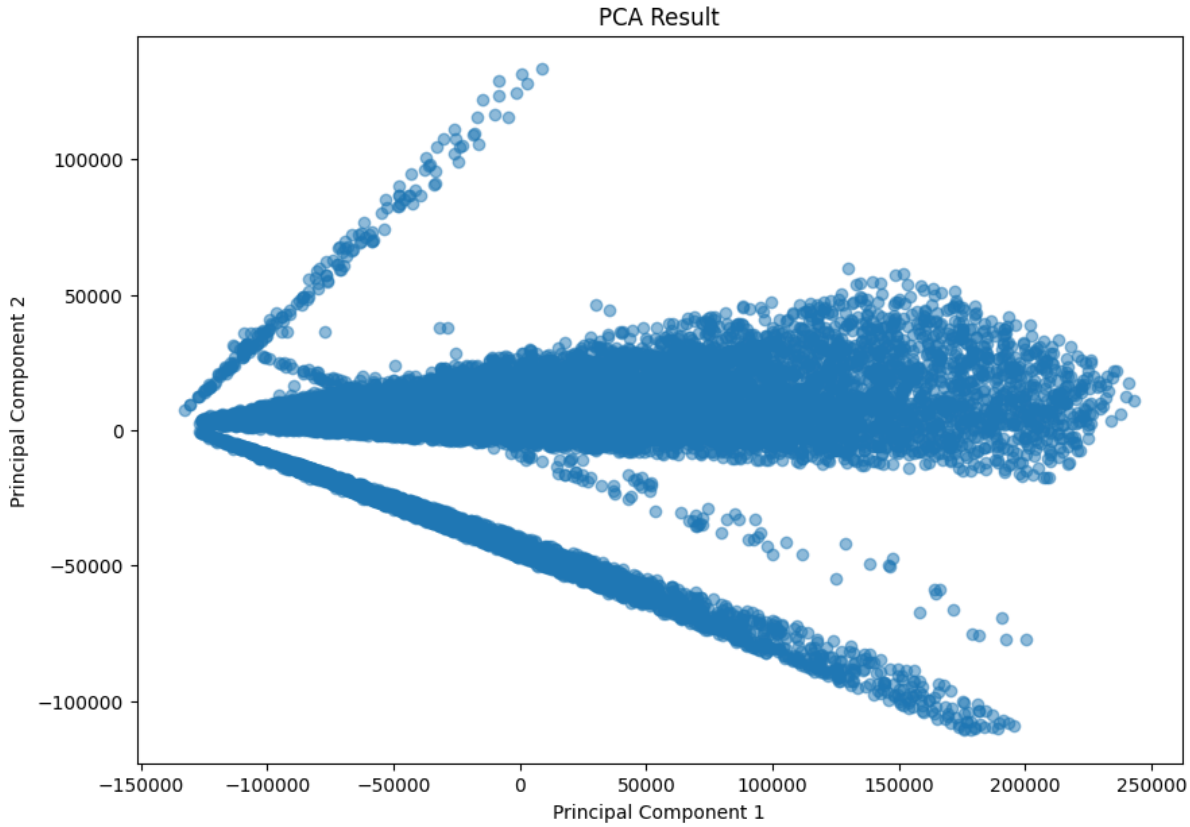



```
# Scatter Plot Matrix (Pair Plot)
sns.pairplot(data[numerical_columns])    "pairplot": Unknown word.
plt.suptitle('Scatter Plot Matrix', y=1.02)    "suptitle": Unknown word.
plt.show()
```




```
# Principal Component Analysis (PCA)
pca = PCA(n_components=2)  # Reduce to 2 dimensions for visualization
pca_result = pca.fit_transform(data[numerical_columns].dropna())    "dropna": Unknown word.
pca_df = pd.DataFrame(data=pca_result, columns=['Principal Component 1', 'Principal Component 2'])
```

```
# Scatter Plot of PCA Result
plt.figure(figsize=(10, 7))    "figsize": Unknown word.
plt.scatter(pca_df['Principal Component 1'], pca_df['Principal Component 2'], alpha=0.5)
plt.title('PCA Result')
plt.xlabel('Principal Component 1')    "xlabel": Unknown word.
plt.ylabel('Principal Component 2')    "ylabel": Unknown word.
plt.show()
```



Conclusion

The Exploratory Data Analysis (EDA) on the loan dataset provided valuable insights into the data. By examining individual variables and their relationships, we were able to uncover patterns and trends that can inform further analysis and decision-making.

Key Findings:

1. **Distribution of Variables**: By examining individual numerical and categorical features, we identified important trends and distributions. The summary statistics revealed the central tendencies and dispersion of numerical data, highlighting features with significant variability.

2. **Handling of Missing Values and Outliers**: Proper handling of missing values ensured the dataset's completeness, while addressing outliers helped in maintaining the integrity of the data, preventing skewed analyses.
3. **Correlations and Relationships**: Through bivariate and multivariate analyses, we uncovered significant correlations between features. These relationships are crucial for predictive modeling and further statistical analyses.
4. **Visual Insights**: The various visualizations, including histograms, bar plots, scatter plots, box plots, violin plots, and heat-maps, provided a clearer understanding of the data's distribution and relationships. These visual tools are essential for communicating findings effectively.

Practical Implications:

1. **Data-Driven Decisions**: The insights gained from this EDA can inform data-driven decision-making processes within financial institutions. Understanding borrower profiles, loan characteristics, and repayment behaviors can lead to more effective risk management strategies.
2. **Model Development**: The identified correlations and patterns serve as a foundation for developing robust predictive models. Features with strong relationships can be prioritised in model building, improving the accuracy and reliability of predictions.
3. **Further Research**: The findings from this analysis highlight areas that warrant further investigation. Additional research can delve deeper into specific patterns and relationships, uncovering more nuanced insights.

Future Work:

1. **Advanced Modeling**: Building on the insights gained, future work could involve developing and validating predictive models to forecast loan defaults, interest rates, and other key metrics.
2. **Feature Engineering**: Further exploration of feature engineering techniques can enhance the dataset, potentially leading to improved model performance.
3. **Integration with External Data**: Integrating the loan dataset with external data sources, such as economic indicators and credit scores, can provide a more comprehensive view and lead to more accurate predictions.

The EDA process has laid a strong foundation for further analysis and model development. By systematically examining the data and uncovering key patterns, we are better equipped to make informed decisions and drive meaningful outcomes.

Acknowledgments

I would like to extend my heartfelt thanks to my mentor and data provider, Mr. Ziyaad Mahudawala. Your guidance, leadership, vision and expertise have been invaluable throughout this project and also inspired me to push my boundaries and strive for excellence.
A special thank you to Futurense Technology for providing me with this opportunity to work on such an exciting project. I am grateful for the support and encouragement from the entire team.

References

- https://pandas.pydata.org/docs/user_guide/index.html
- https://numpy.org/doc/stable/user/basics.html
- https://matplotlib.org/stable/index.html
- https://seaborn.pydata.org/
- https://medium.com/@lamsampathkumar0/eda-exploratory-data-analysis-project-using-python-de90cbf4e128