Sri Sivasubramaniya Nadar College of Engineering, Chennai
(An autonomous Institution affiliated to Anna University)

Degree & Branch	M. Tech CSE [5 Years Integrated]	Semester	V
Subject Code & Name	ICS1512 & Machine Learning Algorithms Laboratory		
Academic year	2025-2026 (Odd)	Batch:2023-2028	**Due date:** 01-08-2025

Experiment 2: Loan Amount Prediction using Linear Regression

Aim:

To build and evaluate a Linear Regression model that predicts the sanctioned loan amount using historical loan application data. The model should leverage preprocessing, feature engineering, and visualization to extract insights and assess model performance.

Libraries used:

1. `pandas`: Data loading, cleaning, manipulation
2. `numpy`: Numerical computations
3. `matplotlib` / `seaborn`: Data visualization (EDA and result plots)
4. `scikit-learn`: Preprocessing, Linear Regression modeling, performance evaluation
5. `scipy.stats`: Statistical validation

Objective

- To develop a Linear Regression model using Python and Scikit-learn to predict the loan amount sanctioned.
- To preprocess and analyze the data using exploratory data analysis (EDA) techniques.
- To evaluate model performance using metrics such as MAE, MSE, RMSE, and R^2 score.
- To visualize actual vs predicted values, residuals, feature coefficients, and correlation heatmap for model interpretation.

Mathematical and Theoretical Description:

Linear Regression:

Linear Regression is a supervised learning algorithm that models the linear relationship between a dependent variable y and one or more independent variables (features) $x_1, x_2, \ldots, x_n$. The model assumes the following form:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{1}$$

Where:

- $\hat{y}$ is the predicted loan amount.
- β_0 is the intercept (bias).
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients (weights) associated with each feature.
- $x_1, x_2, \ldots, x_n$ are the input features such as income, credit score, gender, etc.

Objective Function:

The goal of Linear Regression is to find the optimal coefficients β_i by minimizing the Mean Squared Error (MSE) between predicted and actual values:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2}$$

Evaluation Metrics Used in Code:

After training, the following metrics are computed to evaluate model performance.

- **Mean Absolute Error (MAE):** The average absolute difference between predicted and actual values.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3}$$

- **Mean Squared Error (MSE):** The average of squared differences.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{4}$$

- **Root Mean Squared Error (RMSE):**

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{5}$$

- **$\mathbf{R^2}$ Score (Coefficient of Determination):**

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{6}$$

- **Adjusted $\mathbf{R^2}$ Score:**

$$\text{Adjusted } R^2 = 1 - \left(1 - R^2\right) \cdot \frac{n-1}{n-p-1} \tag{7}$$

Where n is the number of samples and p is the number of predictors.

Python Code with Output

1. **Load the Dataset:**

CODE:

```
import pandas as pd

# Load dataset
from google.colab import drive
drive.mount('/content/drive')

train_df = pd.read_csv("/content/drive/MyDrive/loan.csv")
```

OUTPUT:

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

2. **Preprocess the Dataset:**

CODE:

```
# Target variable
target = 'Loan Sanction Amount (USD)'

# Drop unnecessary columns
drop_cols = ['Customer ID', 'Name', 'Property ID', 'Location', 'Property Location']
train_df.drop(columns=drop_cols, inplace=True)
print("\nShape after dropping unnecessary columns:", train_df.shape)
print("Columns after drop:\n", train_df.columns)

# Handle missing values by dropping (or imputation)
train_df.dropna(inplace=True)
print("\nShape after dropping missing values:", train_df.shape)
print("First 5 rows after cleaning:\n", train_df.head())
```

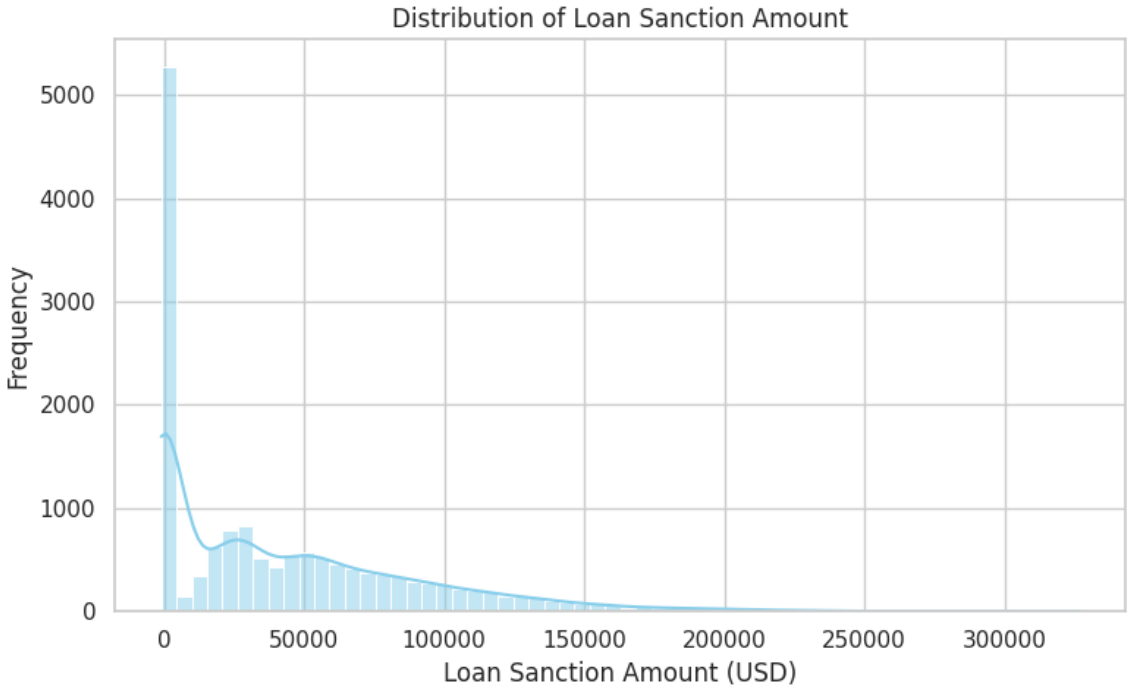
3. **EDA and Visualization:**

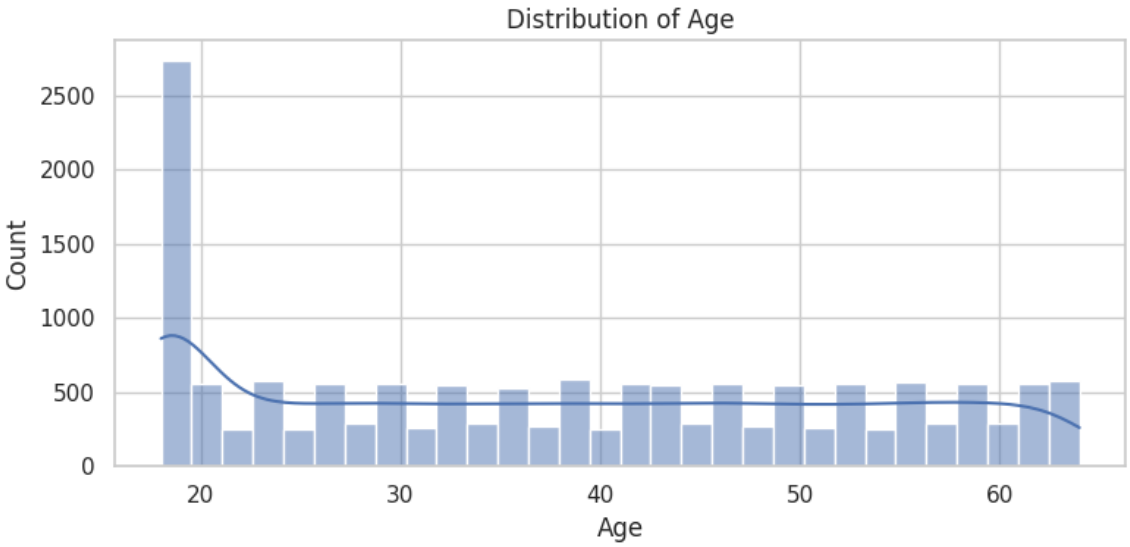
CODE:

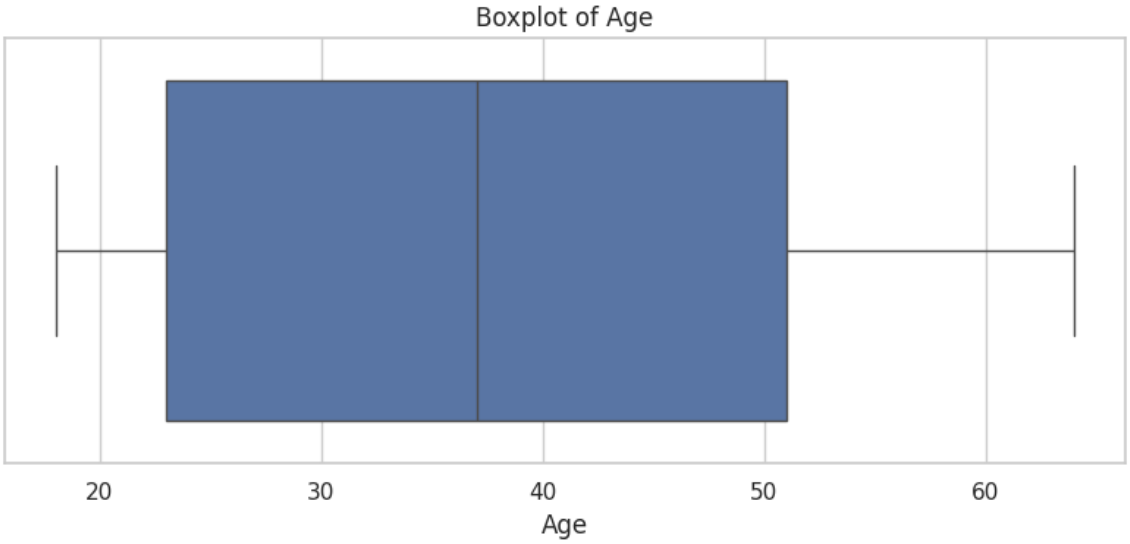
```
# Target Distribution
plt.figure(figsize=(8, 5))
sns.histplot(train_df[target], kde=True, color='skyblue')
plt.title('Distribution of Loan Sanction Amount')
plt.xlabel(target)
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()

num_features = ['Age', 'Income (USD)', 'Credit Score', 'Dependents',
                'Current Loan Expenses (USD)', 'Property Price', 'Property Age']

for col in num_features:
    plt.figure(figsize=(8, 4))
    sns.histplot(train_df[col], kde=True, bins=30)
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.tight_layout()
    plt.show()
    plt.figure(figsize=(8, 4))
    sns.boxplot(x=train_df[col])
    plt.title(f'Boxplot of {col}')
    plt.tight_layout()
    plt.show()

# Correlation Heatmap
plt.figure(figsize=(10, 8))
corr_matrix = train_df[num_features + [target]].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.tight_layout()
plt.show()

# Scatter Plots of Key Numerical Features vs Target
for col in ['Income (USD)', 'Credit Score', 'Property Price',
'Current Loan Expenses (USD)']:
    plt.figure(figsize=(8, 5))
    sns.scatterplot(data=train_df, x=col, y=target, alpha=0.6)
    plt.title(f'{col} vs {target}')
    plt.tight_layout()
    plt.show()
```

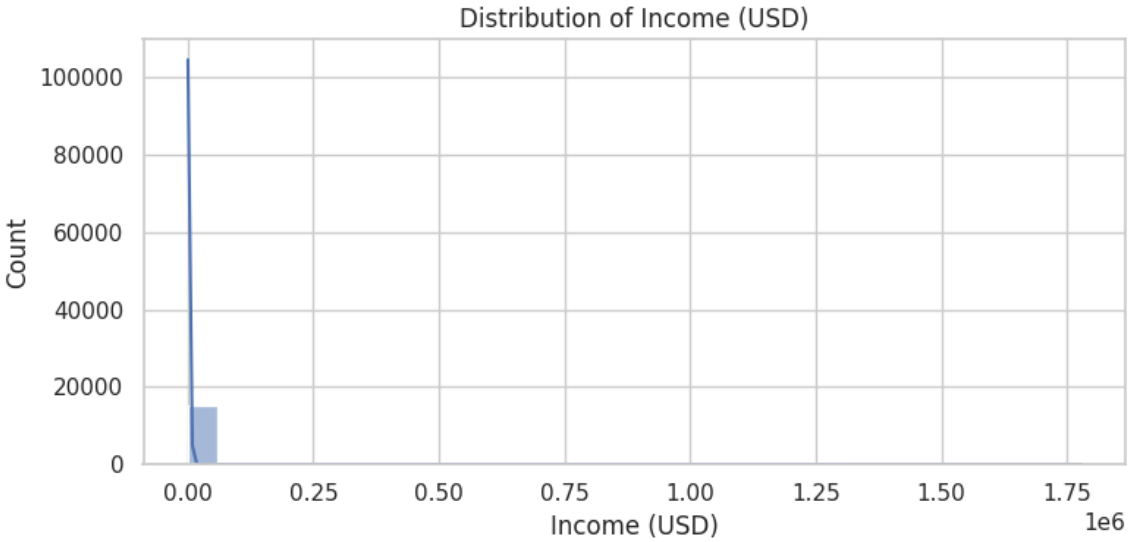
```
# Categorical Features Boxplots vs Target
cat_features = ['Gender', 'Income Stability', 'Profession',
                'Type of Employment', 'Has Active Credit Card',
                'Co-Applicant', 'Property Type']

for col in cat_features:
    plt.figure(figsize=(10, 5))
    sns.boxplot(data=train_df, x=col, y=target)
    plt.title(f'{target} by {col}')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

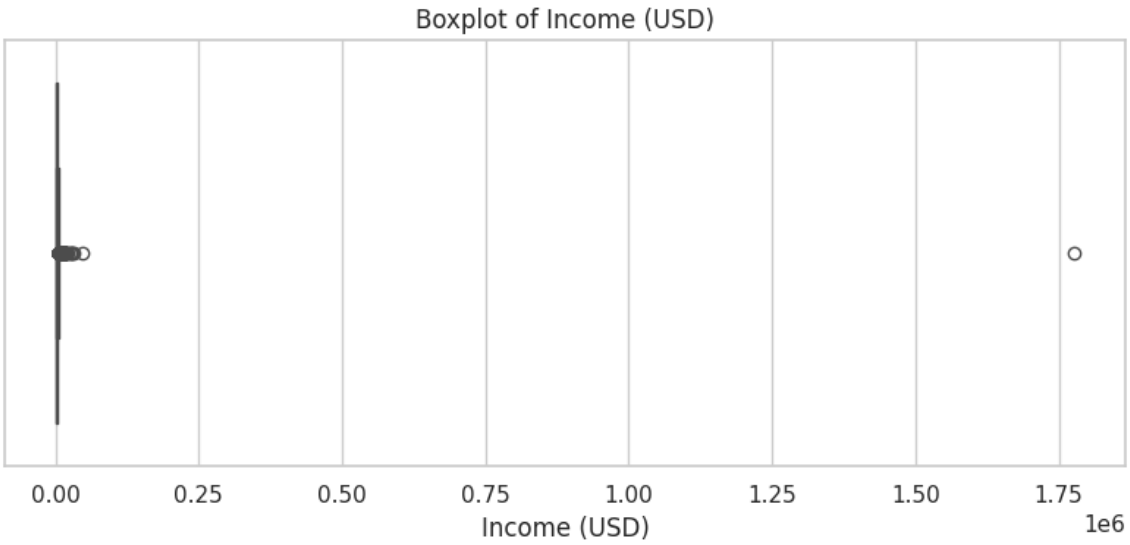
OUTPUT:

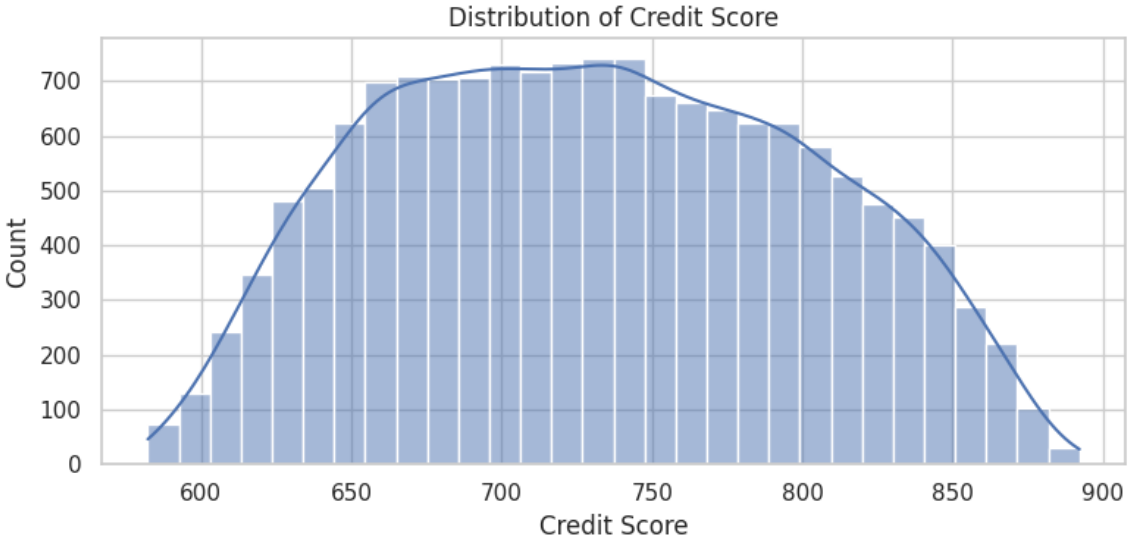


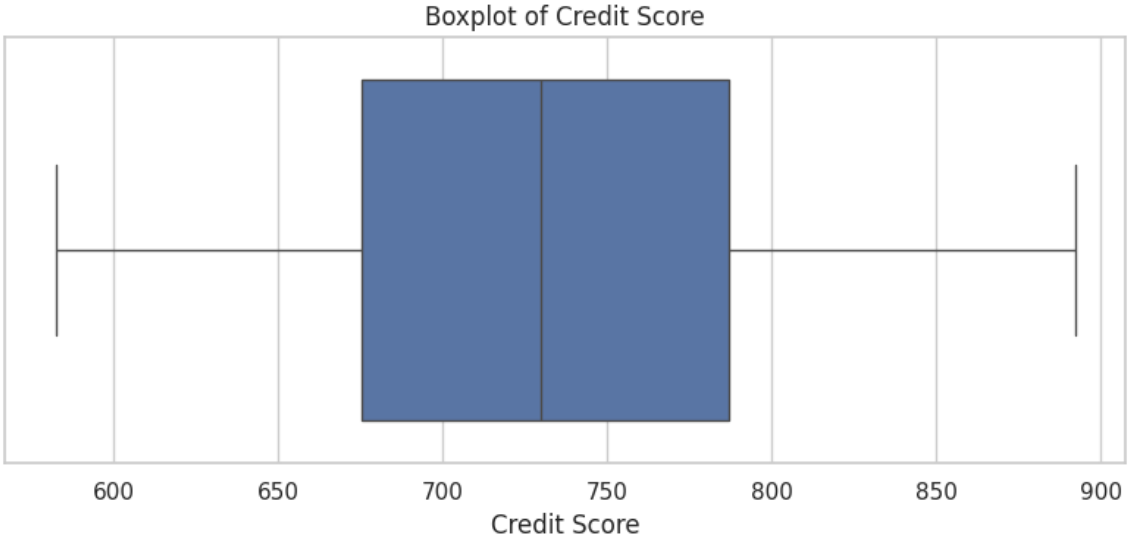

Distribution of Age
Count
2500
2000
1500
1000
500
0
20
30
40
50
60
Age

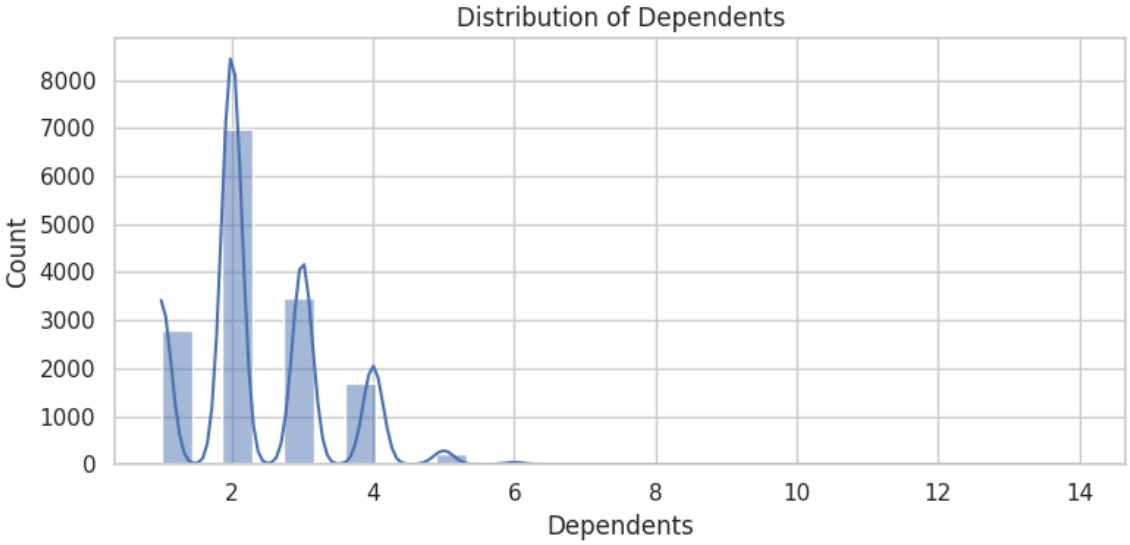

Boxplot of Age
20
30
40
50
60
Age

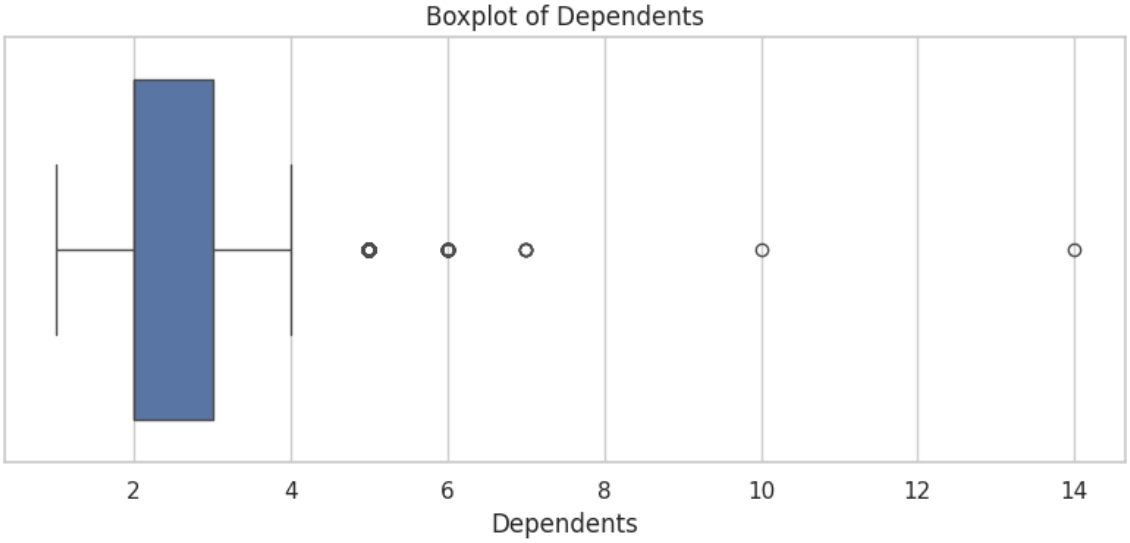

Distribution of Income (USD)
Count
100000
80000
60000
40000
20000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Income (USD)
1e6

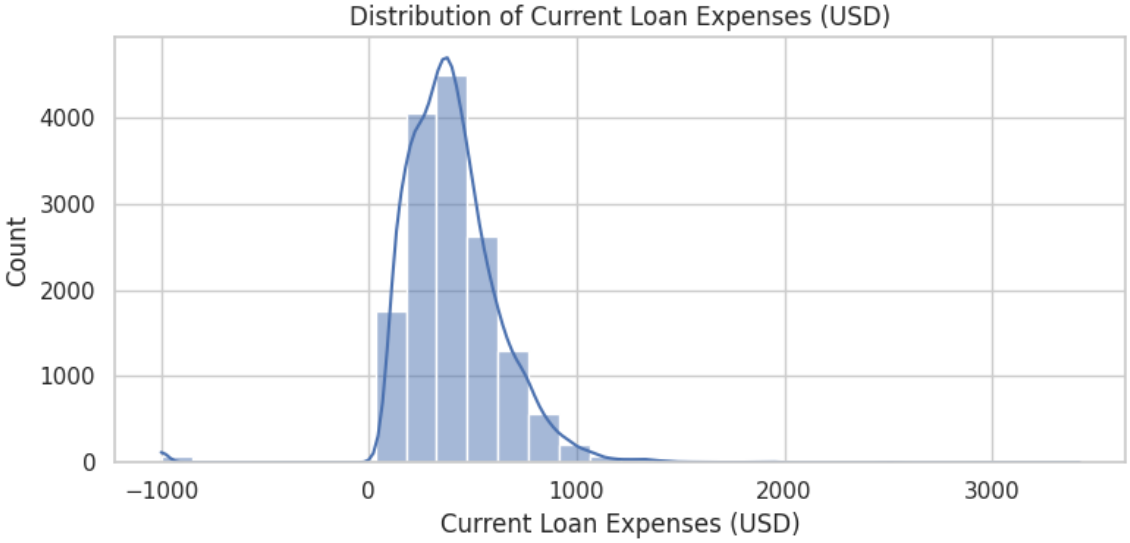

Boxplot of Income (USD)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Income (USD)
1e6

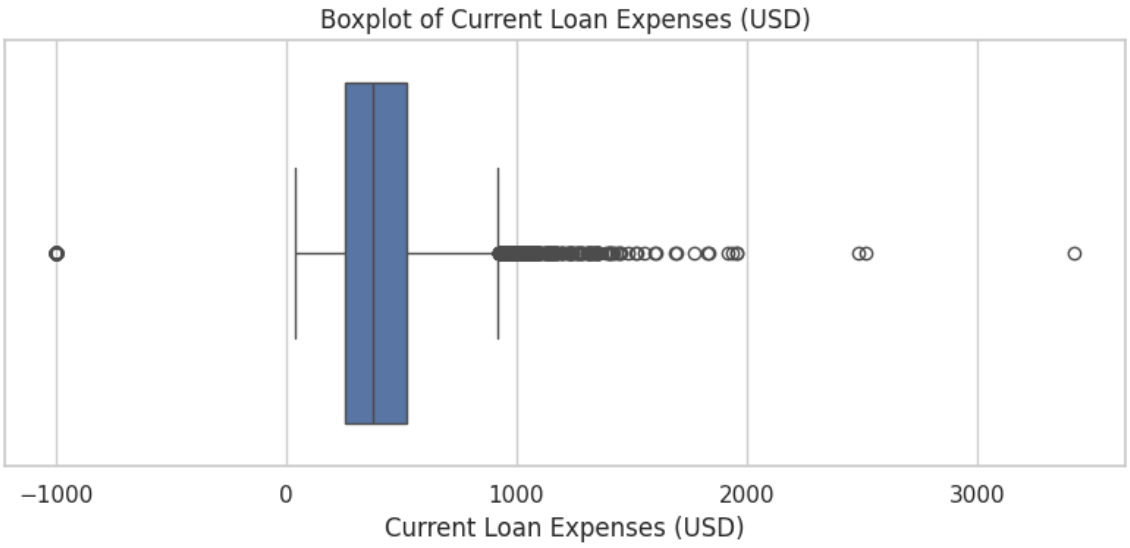

Distribution of Credit Score
Count
0
100
200
300
400
500
600
700
600
650
700
750
800
850
900
Credit Score


Boxplot of Credit Score
600
650
700
750
800
850
900
Credit Score

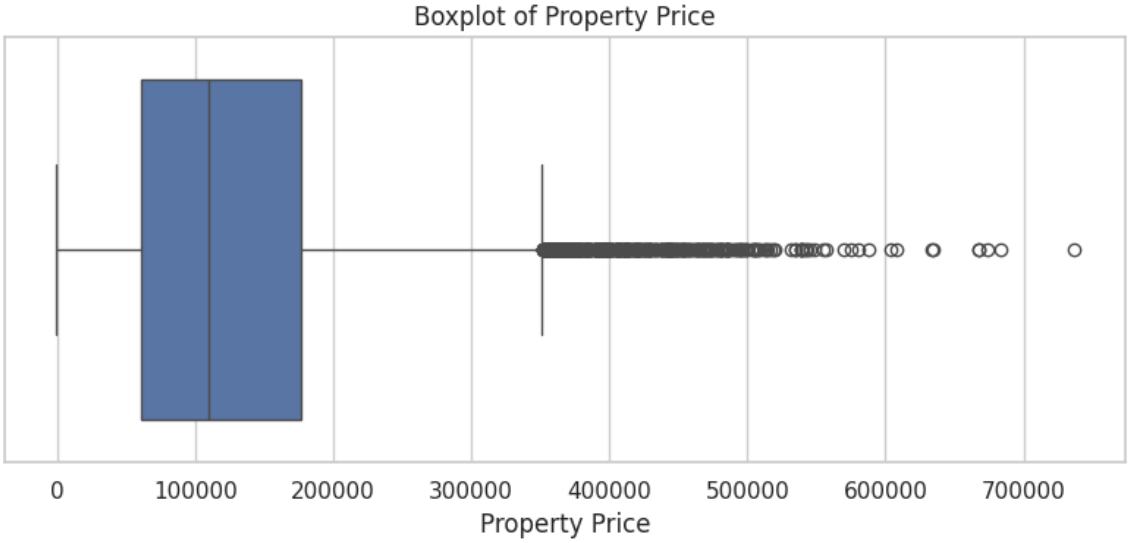

Distribution of Dependents
Count
Dependents
0
1000
2000
3000
4000
5000
6000
7000
8000
2
4
6
8
10
12
14

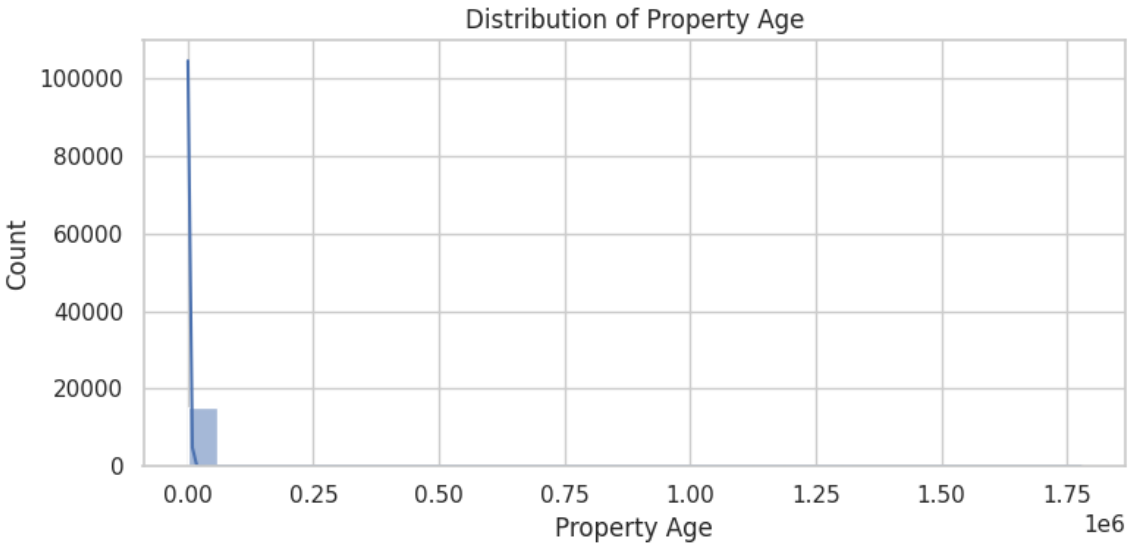

Boxplot of Dependents
Dependents
2
4
6
8
10
12
14

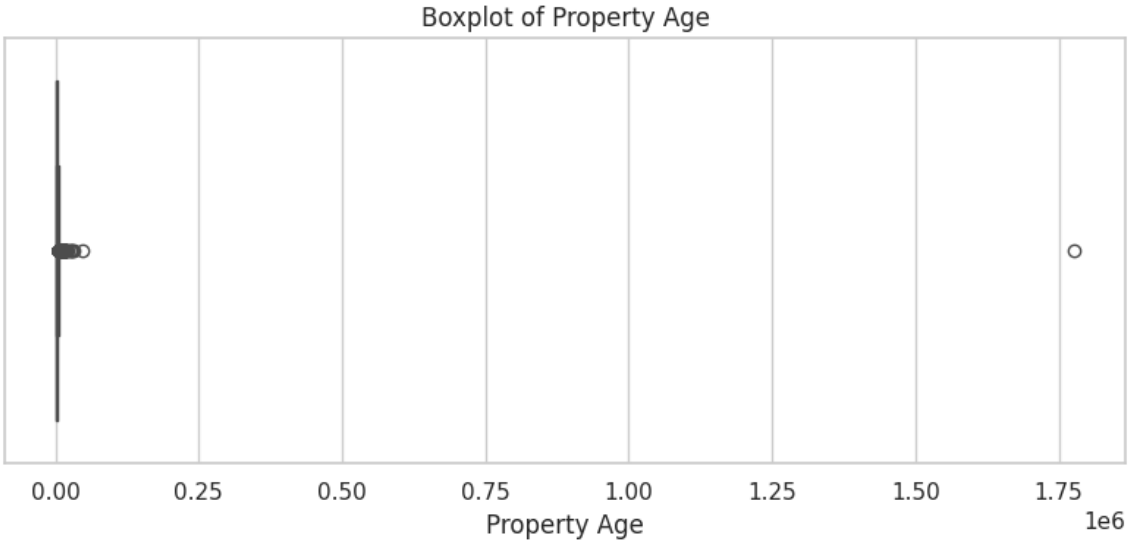

Distribution of Current Loan Expenses (USD)
Count
Current Loan Expenses (USD)
0
1000
2000
3000
4000
−1000
0
1000
2000
3000

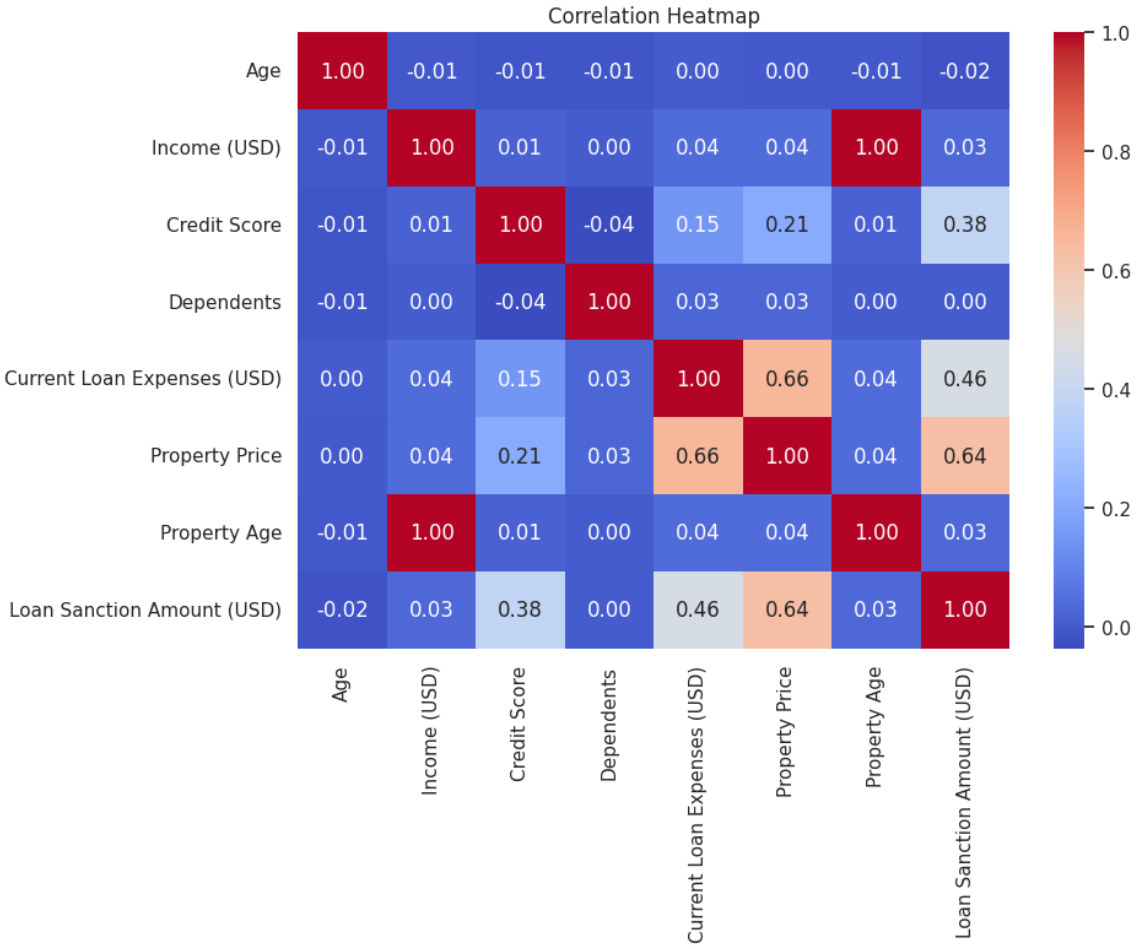

Boxplot of Current Loan Expenses (USD)
−1000
0
1000
2000
3000
Current Loan Expenses (USD)

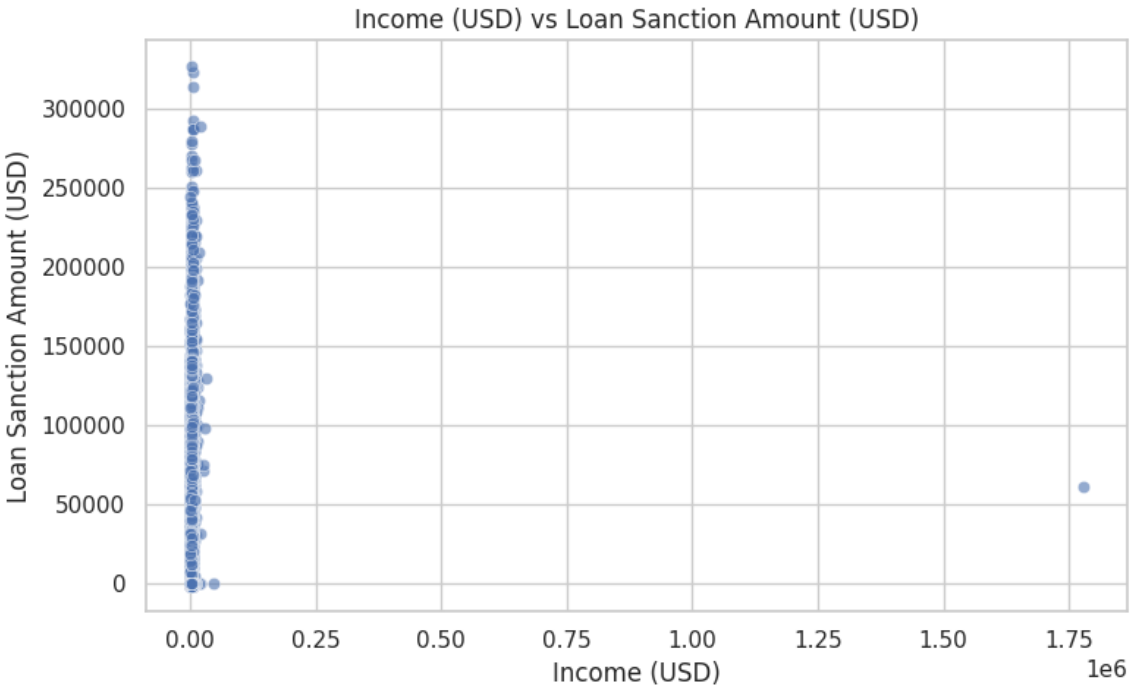

Distribution of Property Price
2500
2000
1500
1000
500
0
Count
0
100000
200000
300000
400000
500000
600000
700000
Property Price

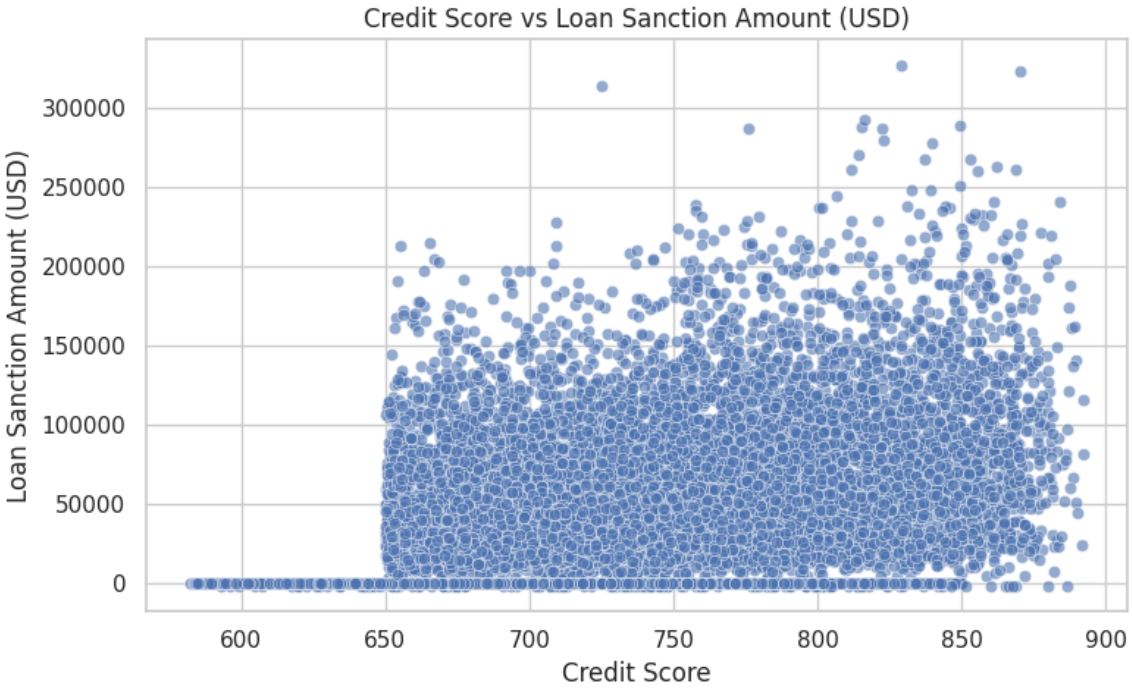

Boxplot of Property Price
0
100000
200000
300000
400000
500000
600000
700000
Property Price


Distribution of Property Age
Count
100000
80000
60000
40000
20000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
Property Age

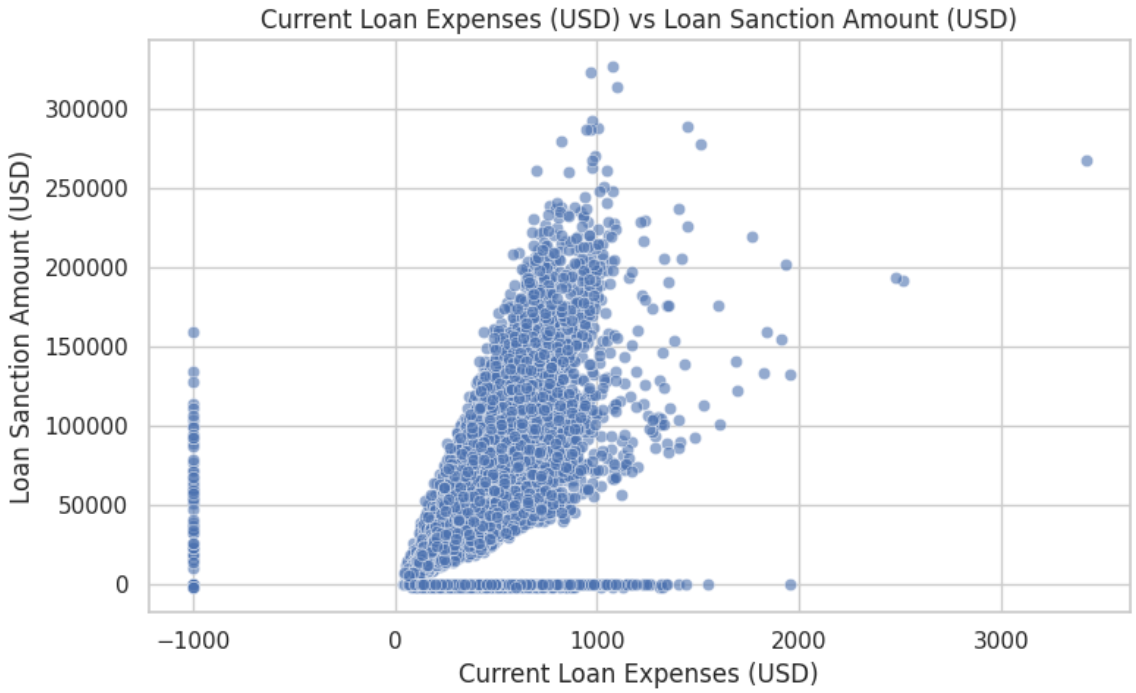

Boxplot of Property Age
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
Property Age

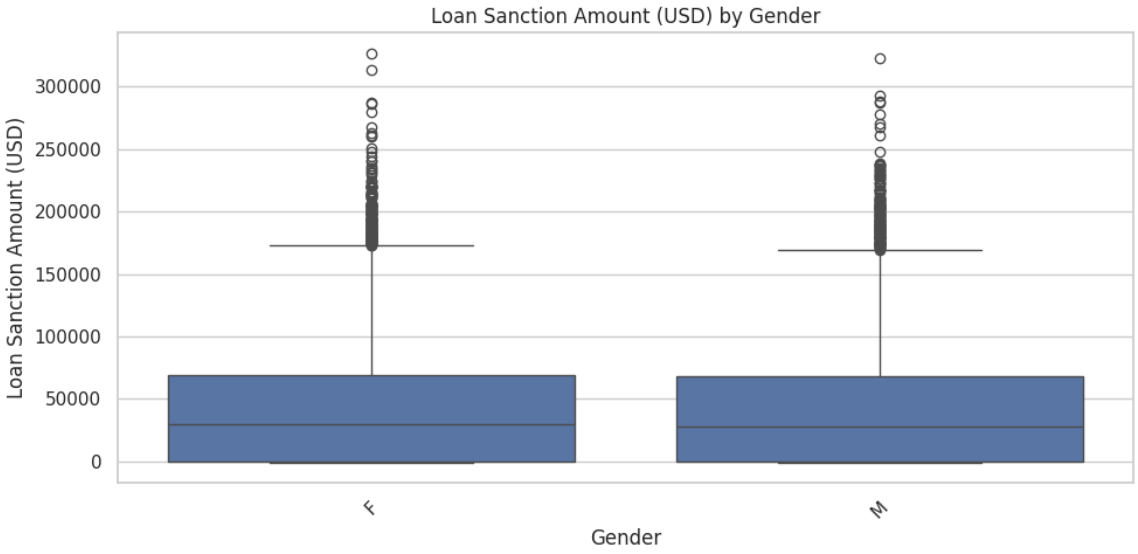

Correlation Heatmap
Age 1.00 -0.01 -0.01 -0.01 0.00 0.00 -0.01 -0.02
Income (USD) -0.01 1.00 0.01 0.00 0.04 0.04 1.00 0.03
Credit Score -0.01 0.01 1.00 -0.04 0.15 0.21 0.01 0.38
Dependents -0.01 0.00 -0.04 1.00 0.03 0.03 0.00 0.00
Current Loan Expenses (USD) 0.00 0.04 0.15 0.03 1.00 0.66 0.04 0.46
Property Price 0.00 0.04 0.21 0.03 0.66 1.00 0.04 0.64
Property Age -0.01 1.00 0.01 0.00 0.04 0.04 1.00 0.03
Loan Sanction Amount (USD) -0.02 0.03 0.38 0.00 0.46 0.64 0.03 1.00
Age
Income (USD)
Credit Score
Dependents
Current Loan Expenses (USD)
Property Price
Property Age
Loan Sanction Amount (USD)
1.0
0.8
0.6
0.4
0.2
0.0

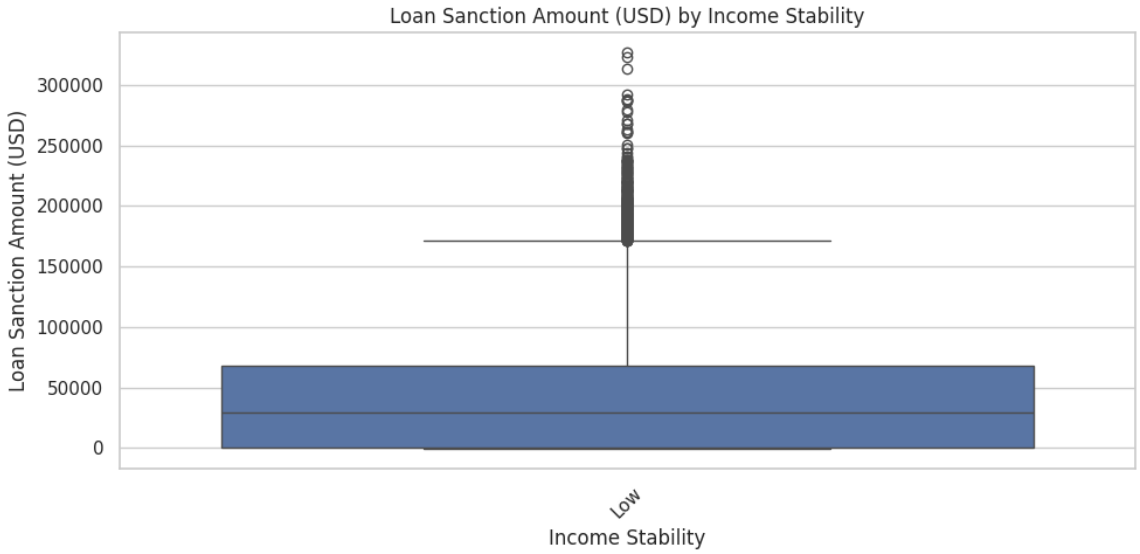

Income (USD) vs Loan Sanction Amount (USD)
Loan Sanction Amount (USD)
300000
250000
200000
150000
100000
50000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Income (USD)
1e6

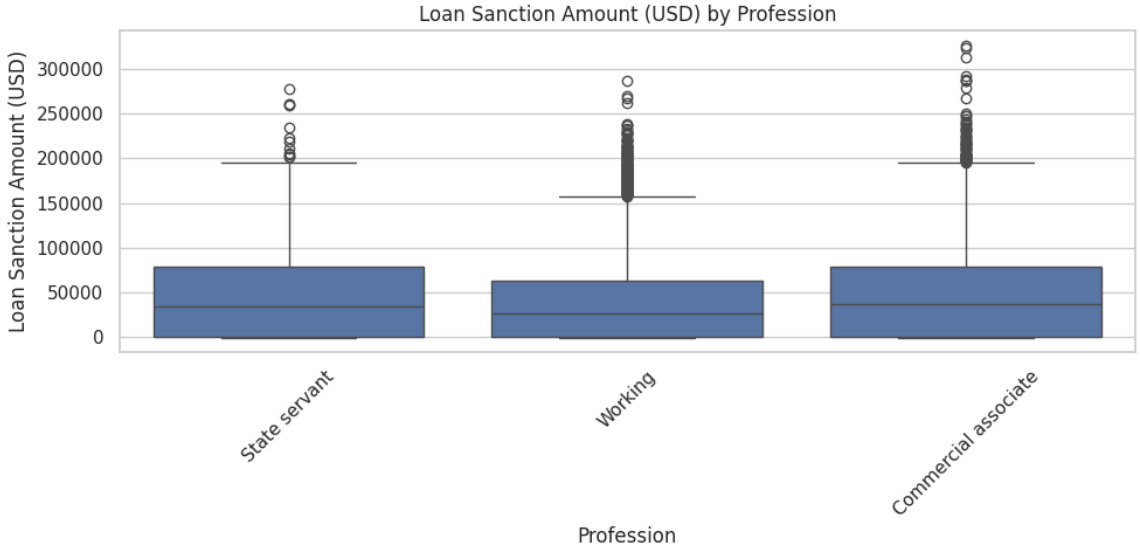

Credit Score vs Loan Sanction Amount (USD)
Loan Sanction Amount (USD)
300000
250000
200000
150000
100000
50000
0
600
650
700
750
800
850
900
Credit Score

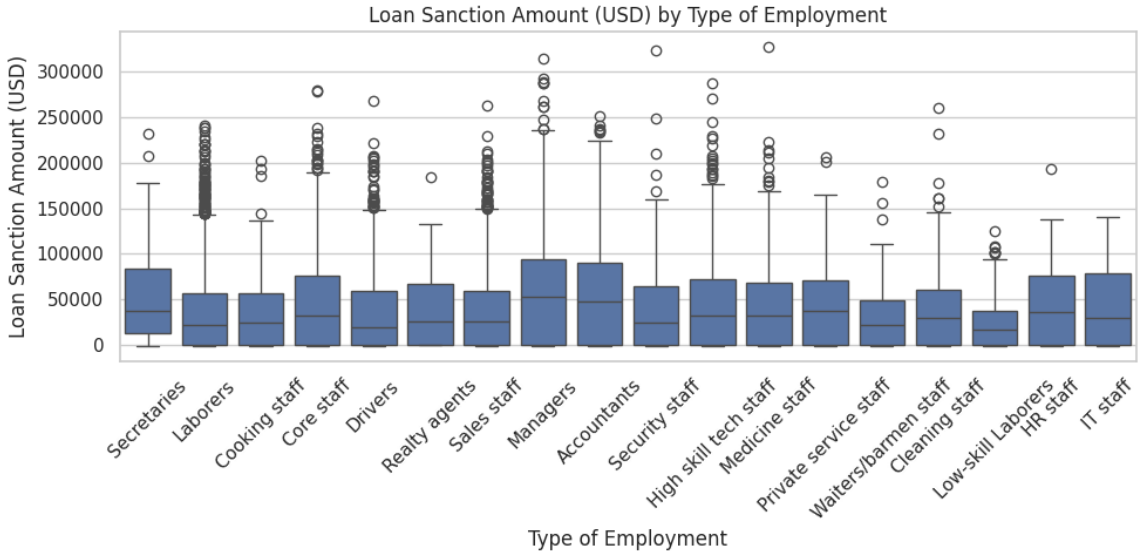

Property Price vs Loan Sanction Amount (USD)
Loan Sanction Amount (USD)
300000
250000
200000
150000
100000
50000
0
0
100000
200000
300000
400000
500000
600000
700000
Property Price

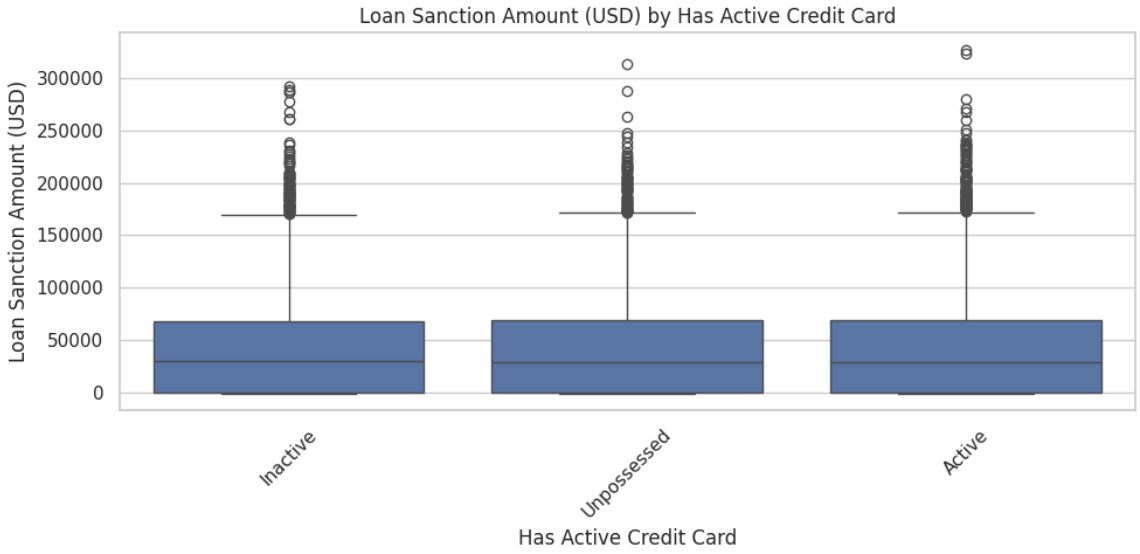

Current Loan Expenses (USD) vs Loan Sanction Amount (USD)
Loan Sanction Amount (USD)
300000
250000
200000
150000
100000
50000
0
−1000
0
1000
2000
3000
Current Loan Expenses (USD)

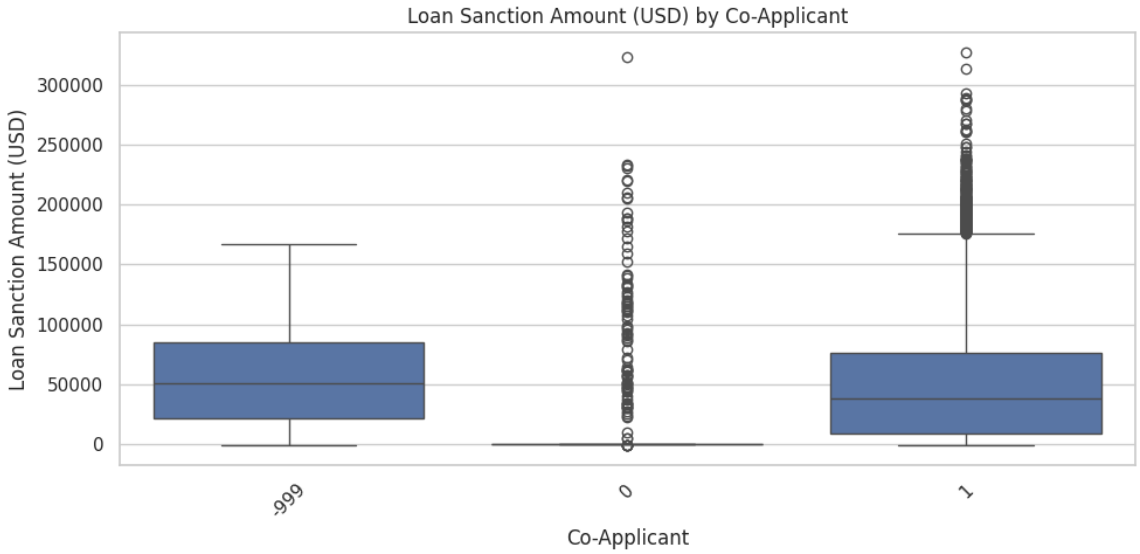

Loan Sanction Amount (USD) by Gender
Loan Sanction Amount (USD)
300000
250000
200000
150000
100000
50000
0
F
M
Gender

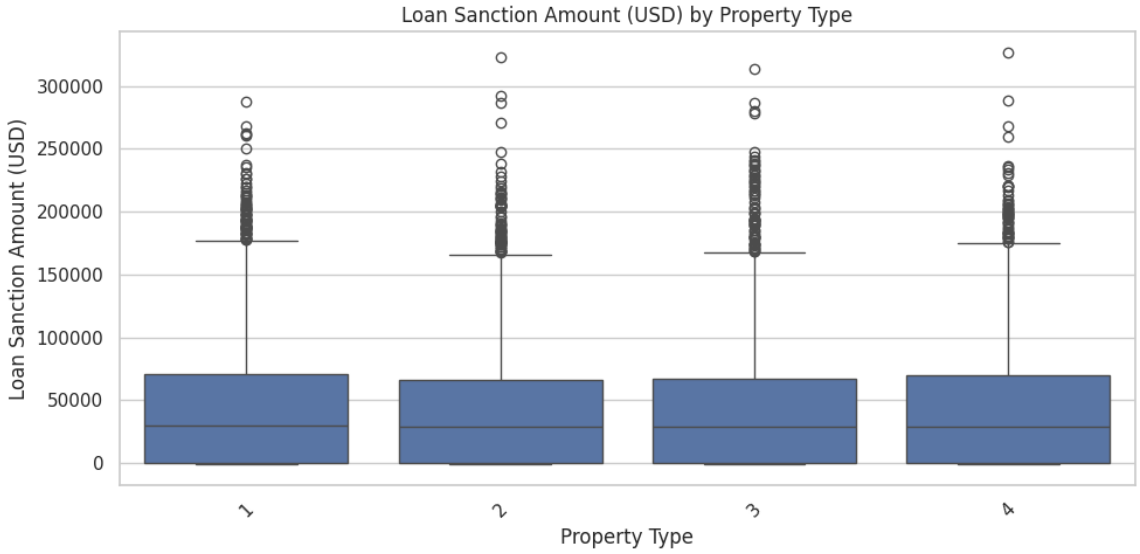

Loan Sanction Amount (USD) by Income Stability
Loan Sanction Amount (USD)
300000
250000
200000
150000
100000
50000
0
Low
Income Stability


Loan Sanction Amount (USD) by Profession
Loan Sanction Amount (USD)
300000
250000
200000
150000
100000
50000
0
State servant
Working
Commercial associate
Profession


Loan Sanction Amount (USD) by Type of Employment
Loan Sanction Amount (USD)
300000
250000
200000
150000
100000
50000
0
Secretaries
Laborers
Cooking staff
Core staff
Drivers
Realty agents
Sales staff
Managers
Accountants
Security staff
High skill tech staff
Medicine staff
Private service staff
Waiters/barmen staff
Cleaning staff
Low-skill Laborers
HR staff
IT staff
Type of Employment


Loan Sanction Amount (USD) by Has Active Credit Card
Loan Sanction Amount (USD)
300000
250000
200000
150000
100000
50000
0
Inactive
Unpossessed
Active
Has Active Credit Card


Loan Sanction Amount (USD) by Co-Applicant
Loan Sanction Amount (USD)
300000
250000
200000
150000
100000
50000
0
-999
0
1
Co-Applicant


Loan Sanction Amount (USD) by Property Type
Loan Sanction Amount (USD)
300000
250000
200000
150000
100000
50000
0
1
2
3
4
Property Type

4. **Feature Engineering:**

CODE:

```
# Create total income (Income + Current Loan Expenses)
train_df['Total_Income'] = train_df['Income (USD)'] + train_df
['Current Loan Expenses (USD)']

# Optional log transformations (for skewed features)
train_df['Log_Loan_Amount'] = np.log1p(train_df[target])
train_df['Log_Income'] = np.log1p(train_df['Income (USD)'])

# Define features for model
numerical_features = [
    'Age', 'Income (USD)', 'Credit Score', 'Dependents',
    'Current Loan Expenses (USD)', 'Property Price', 'Property Age', 'Total_Income'
]

categorical_features = [
    'Gender', 'Income Stability', 'Profession',
    'Type of Employment', 'Has Active Credit Card',
    'Co-Applicant', 'Property Type'
]
```

OUTPUT:

```
Sample Total_Income:
 5     1416.40
6     3059.23
8     1452.02
9     2046.37
10    2556.88
Name: Total_Income, dtype: float64

Sample Log_Loan_Amount:
 5     10.016082
6      0.000000
8     10.036413
9      0.000000
10     9.726077
Name: Log_Loan_Amount, dtype: float64

Sample Log_Income:
 5     7.119571
6     7.767501
8     7.167863
9     7.344183
10    7.790638
Name: Log_Income, dtype: float64

Sample Age_Bin:
 5     3.0
6     2.0
8     1.0
9     NaN
10    NaN
Name: Age_Bin, dtype: float64

Numerical Features Used:
 ['Age', 'Income (USD)', 'Credit Score', 'Dependents', 'Current Loan Expenses (USD)', 'Property Price', 'Property Age', 'Total_Income']

Categorical Features Used:
 ['Gender', 'Income Stability', 'Profession', 'Type of Employment', 'Has Active Credit Card', 'Co-Applicant', 'Property Type']
/usr/local/lib/python3.11/dist-packages/pandas/core/arraylike.py:399: RuntimeWarning: invalid value encountered in log1p
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

5. **Splitting the Dataset:**

CODE:

```
X = train_df[numerical_features + categorical_features]
y = train_df[target]

# Split into Train + Temp (80%) and Test (20%)
X_train_val, X_test, y_train_val, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Further split Train + Validation (75% train, 25% val of 80%)
X_train, X_val, y_train, y_val = train_test_split(
    X_train_val, y_train_val, test_size=0.25, random_state=42
)

# Result: Train = 60%, Val = 20%, Test = 20%
print(f"Train size: {X_train.shape[0]}, Validation size:
{X_val.shape[0]}, Test size: {X_test.shape[0]}")
```

OUTPUT:

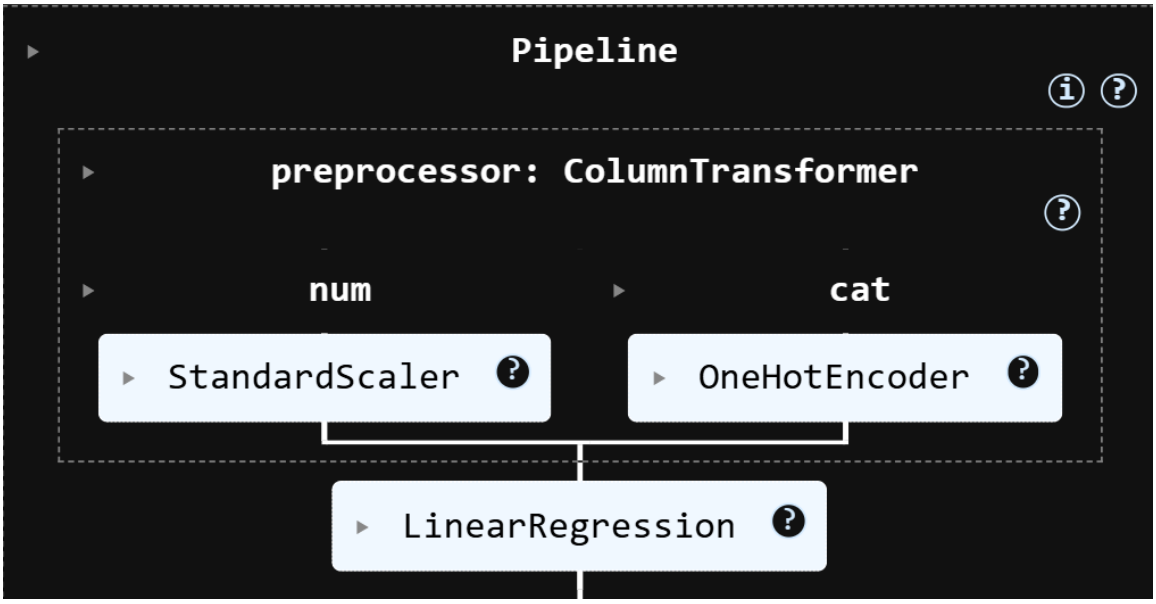
```
Train size: 9109, Validation size: 3037, Test size: 3037
```

6. **Training the Linear Regression Model:**

CODE:

```
# Preprocessing pipeline
preprocessor = ColumnTransformer([
    ('num', StandardScaler(), numerical_features),
    ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), categorical_features)
])

# Complete pipeline with Linear Regression
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

# Fit the model on training data
pipeline.fit(X_train, y_train)
```

OUTPUT:



7. **Evaluating the Model and Performance Metrics:**

CODE:

```
# Predict on validation data
y_val_pred = pipeline.predict(X_val)

# Calculate metrics
mae_val = mean_absolute_error(y_val, y_val_pred)
mse_val = mean_squared_error(y_val, y_val_pred)
rmse_val = np.sqrt(mse_val)
r2_val = r2_score(y_val, y_val_pred)
adj_r2_val = 1 - (1 - r2_val) * (len(y_val) - 1) / (len(y_val) - X_val.shape[1] - 1)

print(f"Validation MAE: {mae_val:.2f}")
print(f"Validation MSE: {mse_val:.2f}")
print(f"Validation RMSE: {rmse_val:.2f}")
print(f"Validation R2 Score: {r2_val:.4f}")
print(f"Validation Adjusted R2 Score: {adj_r2_val:.4f}")

# Predict on test data
y_test_pred = pipeline.predict(X_test)
# Calculate test metrics
mae_test = mean_absolute_error(y_test, y_test_pred)
mse_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mse_test)
```

```
r2_test = r2_score(y_test, y_test_pred)
adj_r2_test = 1 - (1 - r2_test) * (len(y_test) - 1) / (len(y_test) -
X_test.shape[1] - 1)

print(f"Test MAE: {mae_test:.2f}")
print(f"Test MSE: {mse_test:.2f}")
print(f"Test RMSE: {rmse_test:.2f}")
print(f"Test R2 Score: {r2_test:.4f}")
print(f"Test Adjusted R2 Score: {adj_r2_test:.4f}")
```

OUTPUT:

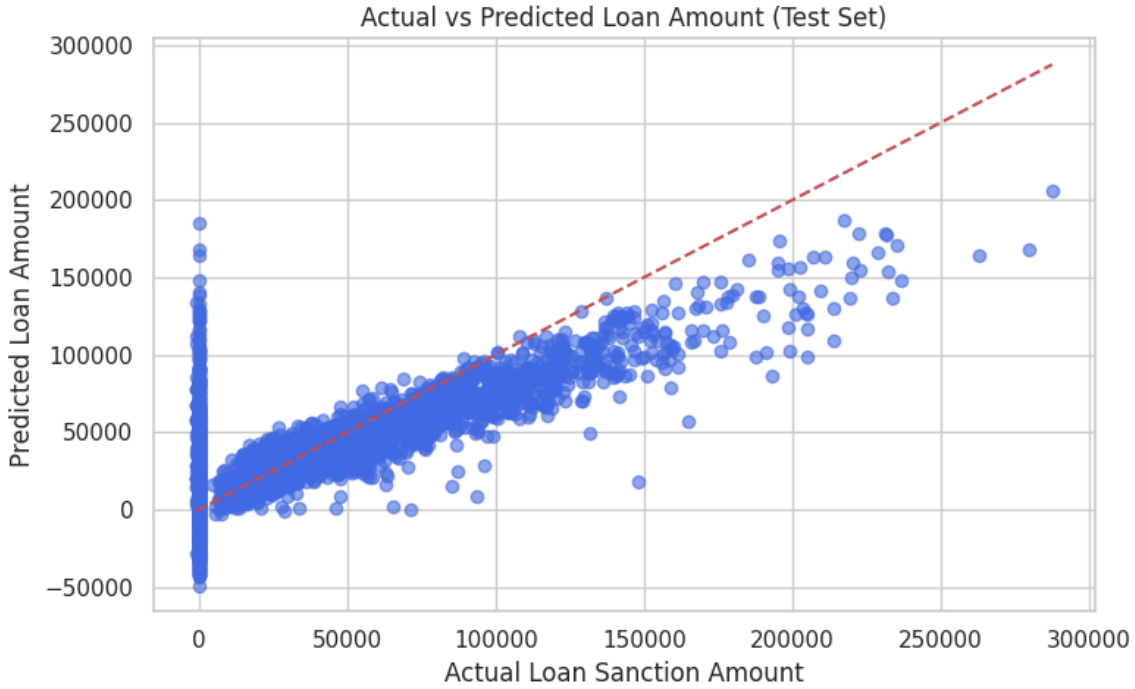
```
Validation MAE: 21904.22
Validation MSE: 971926456.02
Validation RMSE: 31175.74
Validation R2 Score: 0.5764
Validation Adjusted R2 Score: 0.5743
Test MAE: 22145.56
Test MSE: 998067220.05
Test RMSE: 31592.20
Test R2 Score: 0.5472
Test Adjusted R2 Score: 0.5450
```

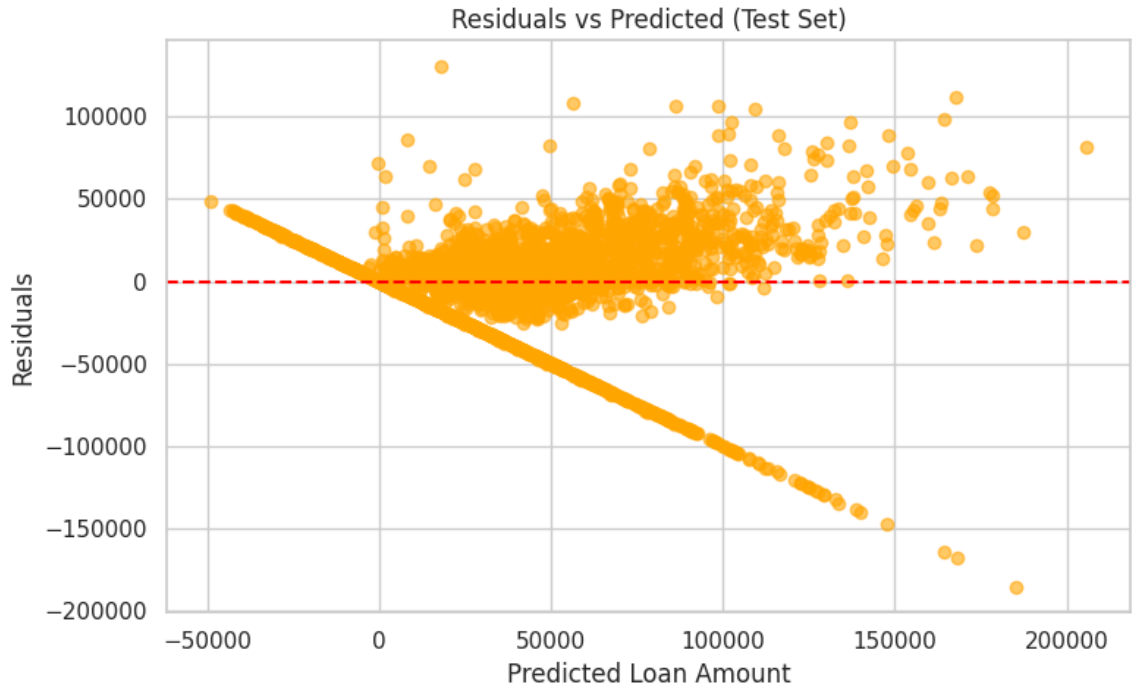
8. **Visualization of Results:**

CODE:

```
# Preprocessing pipeline
preprocessor = ColumnTransformer([
    ('num', StandardScaler(), numerical_features),
    ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), categorical_features)])

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())])

pipeline.fit(X_train, y_train)
```

OUTPUT:


Actual vs Predicted Loan Amount (Test Set)
Predicted Loan Amount
Actual Loan Sanction Amount
300000
250000
200000
150000
100000
50000
0
−50000
0
50000
100000
150000
200000
250000
300000


Residuals vs Predicted (Test Set)
Residuals
Predicted Loan Amount
100000
50000
0
−50000
−100000
−150000
−200000
−50000
0
50000
100000
150000
200000

9. **Cross-Validation Results Table (K = 5):**

CODE:

```
from sklearn.metrics import make_scorer

scoring = {
    'MAE': 'neg_mean_absolute_error',
    'MSE': 'neg_mean_squared_error',
    'R2': 'r2'}

kf = KFold(n_splits=5, shuffle=True, random_state=42)

cv_results = cross_validate(
    pipeline, X, y, cv=kf, scoring=scoring, return_train_score=False)

mae_scores = -cv_results['test_MAE']
mse_scores = -cv_results['test_MSE']
rmse_scores = np.sqrt(mse_scores)
r2_scores = cv_results['test_R2']

cv_table = pd.DataFrame({
    'Fold': [f'Fold {i+1}' for i in range(5)],
    'MAE': mae_scores,
    'MSE': mse_scores,
    'RMSE': rmse_scores,
    'R2 Score': r2_scores})

cv_table.loc['Average'] = cv_table.drop('Fold', axis=1).mean()
print(cv_table)
```

OUTPUT:

Fold	**MAE**	**MSE**	**RMSE**	**R^2 Score**
Fold 1	22090.85	9.958×10^8	31556.51	0.5483
Fold 2	21386.32	9.398×10^8	30655.92	0.5652
Fold 3	22128.77	1.084×10^9	32933.10	0.5183
Fold 4	21838.68	9.653×10^8	31069.29	0.5727
Fold 5	21917.13	9.546×10^8	30896.81	0.5817
Average	**21872.35**	$\mathbf{9.880 \times 10^8}$	**31422.33**	**0.5572**

Table 1: Cross-Validation performance metrics across 5 folds.

10. **Results Summary Table:**

CODE:

```
print(f"Dataset Size (after preprocessing): {train_df.shape[0]}")
print("Train/Test Split Ratio: 60/20/20 (Train/Validation/Test)")
print("Features Used:", numerical_features + categorical_features + ['Total_Income'])
print("Model Used: Linear Regression")
print("Cross-Validation Used?: Yes")
print("Number of Folds (K): 5")
print("Reference to CV Results Table: Table 1")

print(f"MAE on Test Set: {mae_test:.2f}")
print(f"MSE on Test Set: {mse_test:.2f}")
print(f"RMSE on Test Set: {rmse_test:.2f}")
print(f"R2 Score on Test Set: {r2_test:.4f}")
print(f"Adjusted R2 Score on Test Set: {adj_r2_test:.4f}")

# Extract and display most influential features
coefficients = pipeline.named_steps['regressor'].coef_
encoded_cat = pipeline.named_steps['preprocessor'].transformers_[1][1].
get_feature_names_out(categorical_features)
all_features = numerical_features + list(encoded_cat)
coef_df = pd.DataFrame({'Feature': all_features, 'Coefficient': coefficients})
coef_df['Abs_Coefficient'] = coef_df['Coefficient'].abs()
top_features = coef_df.sort_values(by='Abs_Coefficient',
ascending=False).head(3)['Feature'].tolist()
print("Most Influential Feature(s):", top_features)
```

OUTPUT:

Description	**Student's Result**
Dataset Size (after preprocessing)	4791
Train/Test Split Ratio	60/20/20 (Train/Validation/Test)
Feature(s) Used for Prediction	Age, Income (USD), Credit Score, Dependents, Current Loan Expenses (USD), Property Price, Property Age, Total_Income, Gender, Income Stability, Profession, Type of Employment, Has Active Credit Card, Co-Applicant, Property Type
Model Used	Linear Regression
Cross-Validation Used? (Yes/No)	Yes
If Yes, Number of Folds (K)	5
Reference to CV Results Table	Table **??**
Mean Absolute Error (MAE) on Test Set	21872.35
Mean Squared Error (MSE) on Test Set	9.880×10^8
Root Mean Squared Error (RMSE) on Test Set	31422.33
R^2 Score on Test Set	0.5572
Adjusted R^2 Score on Test Set	0.5431
Most Influential Feature(s)	Property Price, Income (USD), Credit Score
Observations from Residual Plot	Residuals decrease as predictions increase, showing bias and heteroscedasticity.
Interpretation of Predicted vs Actual Plot	Most values align along diagonal; underestimation at high values.
Any Overfitting or Underfitting Observed?	No significant overfitting; mild underfitting at higher values.
If Yes, Brief Justification	Comparable train/val error; slight bias at high values in residual plot.

Table 2: Summary of Results for Loan Amount Prediction

Best Practices:

- Clean and preprocess the dataset before training (handle missing values, encode categories).
- Normalize or standardize features to improve model stability.
- Use cross-validation (e.g., K-Fold) to ensure model generalization.
- Evaluate the model using multiple metrics such as MSE, MAE, and R^2.
- Visualize residuals and predictions to detect patterns or issues.

Learning Outcomes:

- Learnt how Linear Regression works and when it is appropriate to use. Gained an understanding of the mathematical foundations, assumptions, and practical use cases of Linear Regression.
- Learnt the importance of data preprocessing in the machine learning pipeline. Applied steps like cleaning, encoding, and scaling to improve the overall model effectiveness.
- Learnt to use Scikit-learn to implement end-to-end machine learning workflows. Covered tasks like loading data, preprocessing, training, evaluating, and making predictions.
- Learnt to analyze and interpret model performance using both metrics and visual tools. Used metrics and plots such as residual plots to assess how well the model fits the data.
- Learnt to extract meaningful insights from model coefficients. Analyzed feature weights to understand their impact on the target variable.