Experiment 2: Loan Amount Prediction using Linear Regression

Shinigdapriya Sathish

July 30, 2025

Aim

To predict the loan amount sanctioned to users using Linear Regression on a real-world dataset and interpret the results through proper evaluation and visualization.

Libraries Used

- pandas
- numpy
- matplotlib
- seaborn
- scikit-learn

Objective

Apply Linear Regression to predict the sanctioned loan amount, evaluate the model using error metrics and cross-validation, and visualize the results to gain insights into model performance.

Mathematical Background

The hypothesis function for Linear Regression is:

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

The model minimizes the Mean Squared Error (MSE):

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

The coefficients are learned using the Normal Equation:

$$w = (X^T X)^{-1} X^T y$$

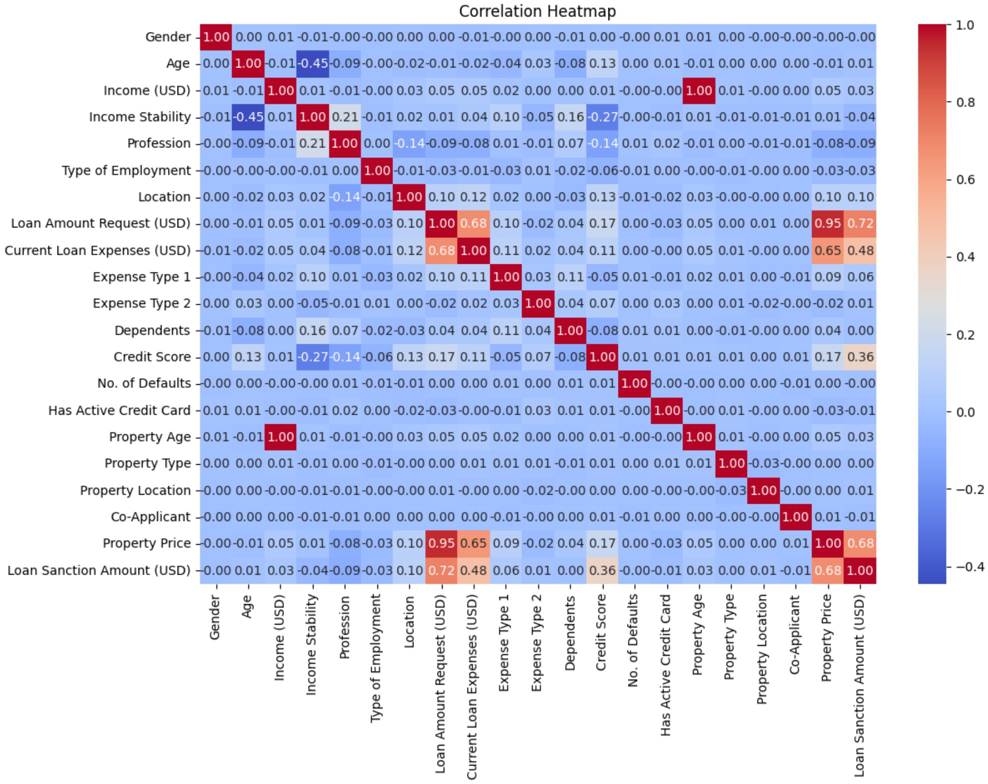
Preprocessing

- Dropped irrelevant columns (e.g., Customer ID, Name, Property ID)
- Filled missing values (mode for categorical, median for numeric)
- Encoded categorical variables using Label Encoding
- Standardized numerical features using StandardScaler

Exploratory Data Analysis

Visualizations include:

- Histogram and Boxplot of `Income (USD)` and `Loan Sanction Amount (USD)`
- Correlation heatmap of numerical features
- Scatter plots for feature-target relationships
- Residual plots to assess linearity assumptions



Model Training

Used `LinearRegression` from scikit-learn. The dataset was split into 80% training and 20% validation sets.

Evaluation Metrics

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R^2 Score
- Adjusted R^2 Score

[ ]:
```python
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error,␣
↪r2_score
```

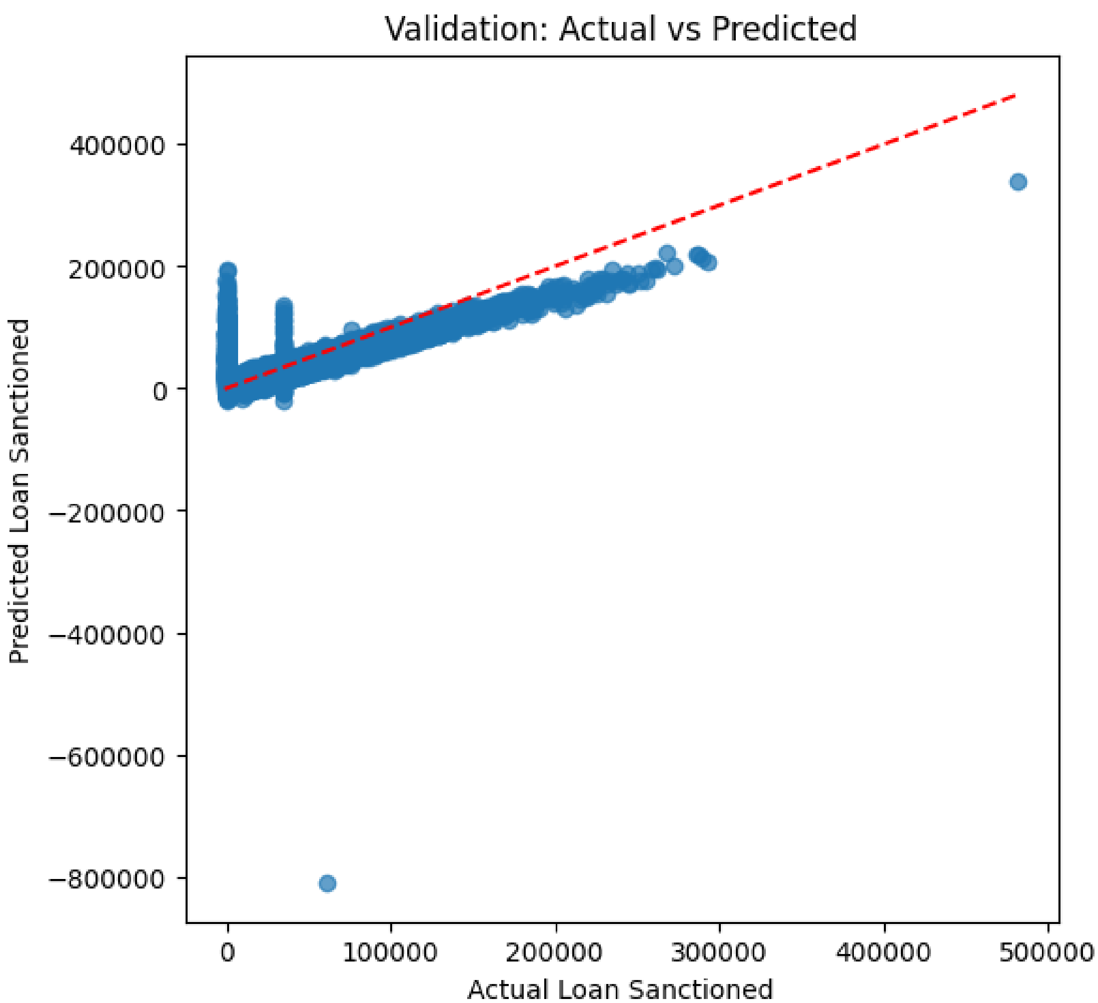
[ ]:
```
train_path = '/content/drive/MyDrive/Docs/archive/train.csv'
df = pd.read_csv(train_path)
```

[ ]:
```
# Step 1: Drop irrelevant columns
df.drop(columns=['Customer ID', 'Name', 'Property ID'], inplace=True)
```

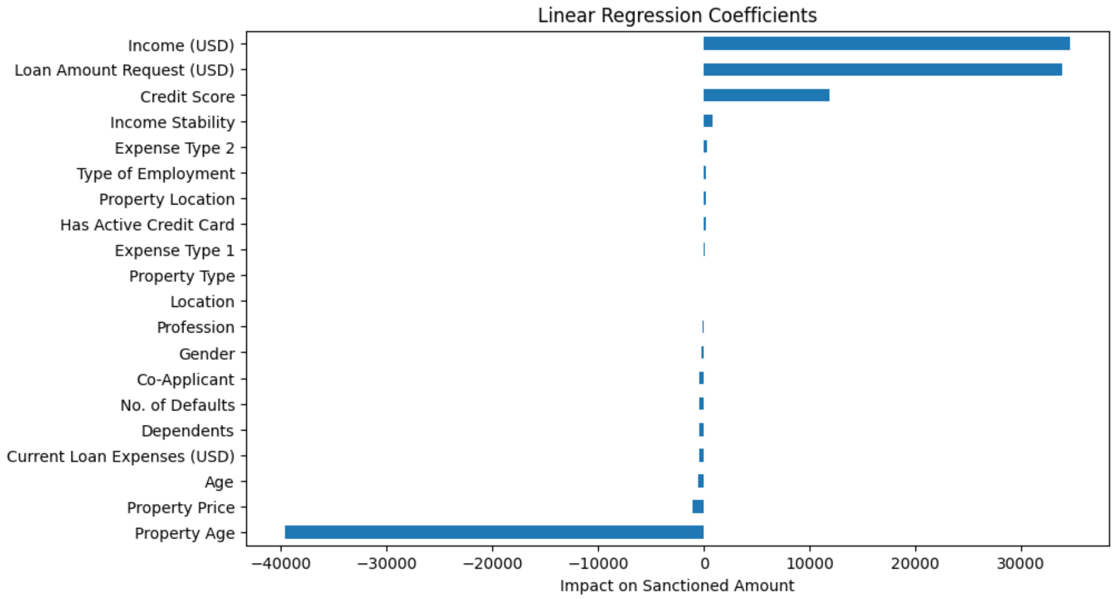
[ ]:
```
# Step 2: Handle missing values (simple strategy: fill with mode for␣
↪categorical, median for numeric)
for col in df.columns:
    if df[col].dtype == 'object':
        df[col] = df[col].fillna(df[col].mode()[0])
    else:
        df[col] = df[col].fillna(df[col].median())

# Step 3: Encode categorical columns
label_encoders = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Step 4: Feature / Target split
target = 'Loan Sanction Amount (USD)'
X = df.drop(columns=[target])
y = df[target]

# Step 5: Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 6: Train-validation split
X_train, X_val, y_train, y_val = train_test_split(X_scaled, y,␣
↪test_size=0.2, random_state=42)

# Step 7: Train model
lr = LinearRegression()
lr.fit(X_train, y_train)

# Step 8: Evaluate
```

```python
y_pred = lr.predict(X_val)

print("MAE:", mean_absolute_error(y_val, y_pred))
print("MSE:", mean_squared_error(y_val, y_pred))
print("R² Score:", r2_score(y_val, y_pred))

# Step 9: Visualization
plt.figure(figsize=(6,6))
plt.scatter(y_val, y_pred, alpha=0.7)
plt.xlabel("Actual Loan Sanctioned")
plt.ylabel("Predicted Loan Sanctioned")
plt.title("Validation: Actual vs Predicted")
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'r--')
plt.show()
```

```
MAE: 21799.224445462914
MSE: 1076701697.7600782
R² Score: 0.531982511846756
```

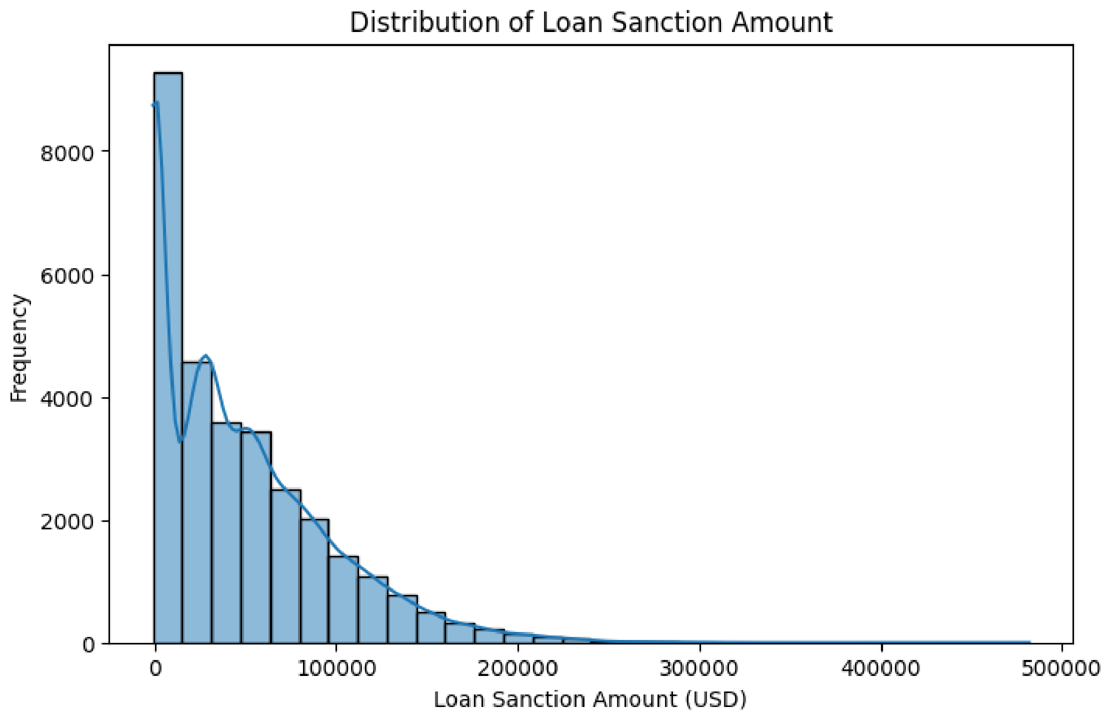


[ ]:
```
# Feature importance
coef = pd.Series(lr.coef_, index=X.columns)
coef.sort_values().plot(kind='barh', figsize=(10,6), title="Linear␣
↪Regression Coefficients")
plt.xlabel("Impact on Sanctioned Amount")
plt.show()
```



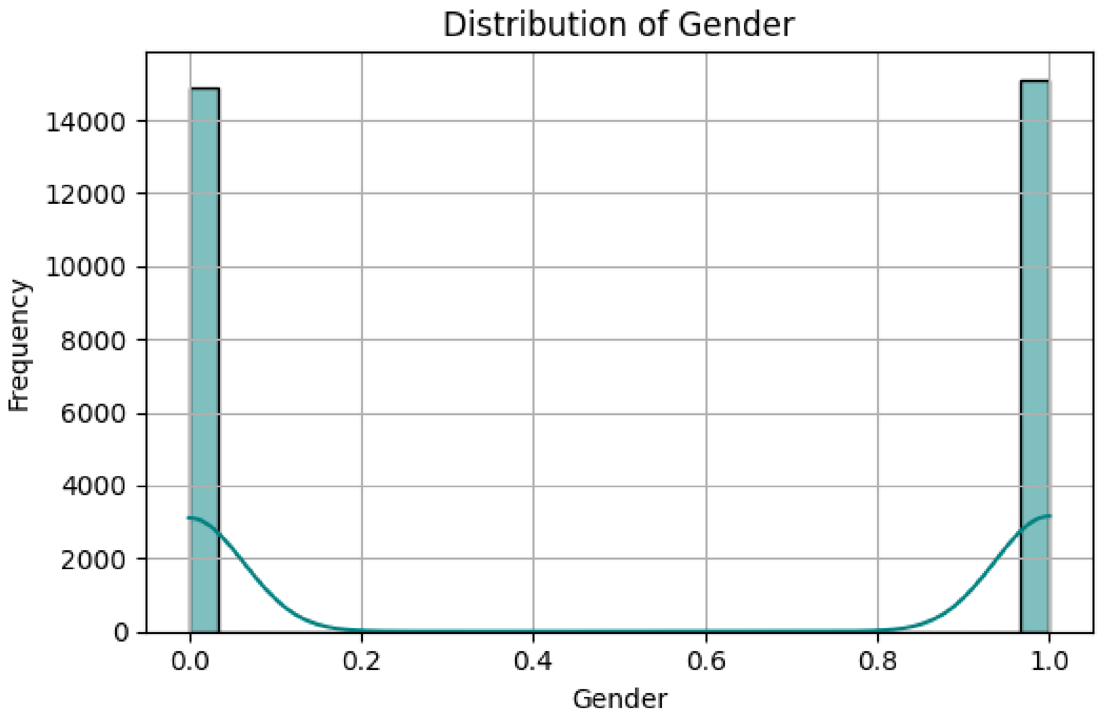
[ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Distribution of Loan Sanction Amount
plt.figure(figsize=(8, 5))
sns.histplot(df['Loan Sanction Amount (USD)'], kde=True, bins=30)
plt.title('Distribution of Loan Sanction Amount')
plt.xlabel('Loan Sanction Amount (USD)')
plt.ylabel('Frequency')
plt.show()


plt.figure(figsize=(8, 5))

numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
print(numeric_cols)

for col in numeric_cols:
    plt.figure(figsize=(6, 4))
    sns.histplot(df[col], kde=True, bins=30, color='teal')
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.grid(True)
    plt.tight_layout()
```

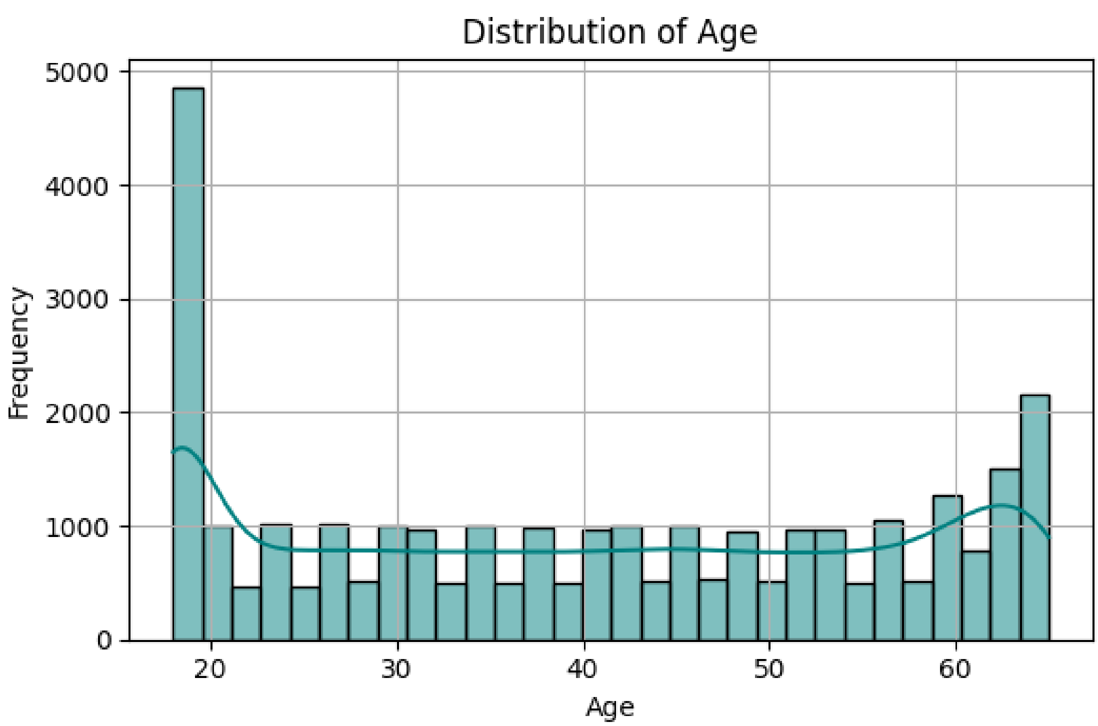
```
plt.show()
```

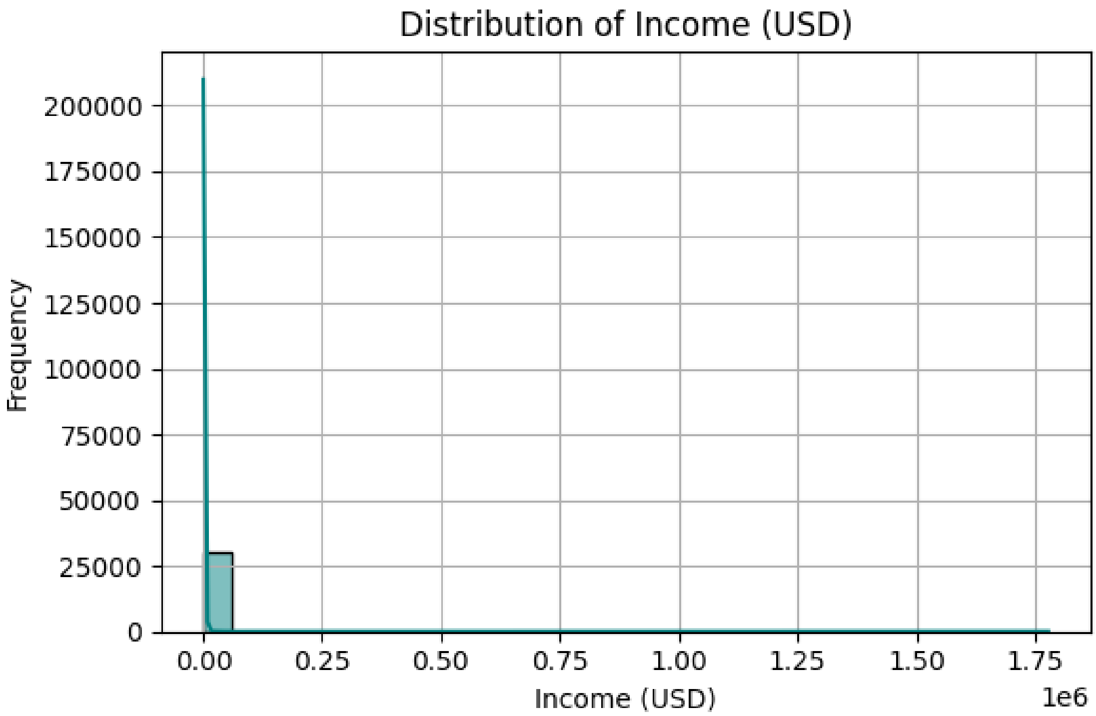



```
Index(['Gender', 'Age', 'Income (USD)', 'Income Stability', 'Profession',
       'Type of Employment', 'Location', 'Loan Amount Request (USD)',
       'Current Loan Expenses (USD)', 'Expense Type 1', 'Expense Type 2',
       'Dependents', 'Credit Score', 'No. of Defaults',
       'Has Active Credit Card', 'Property Age', 'Property Type',
       'Property Location', 'Co-Applicant', 'Property Price',
       'Loan Sanction Amount (USD)'],
      dtype='object')
```

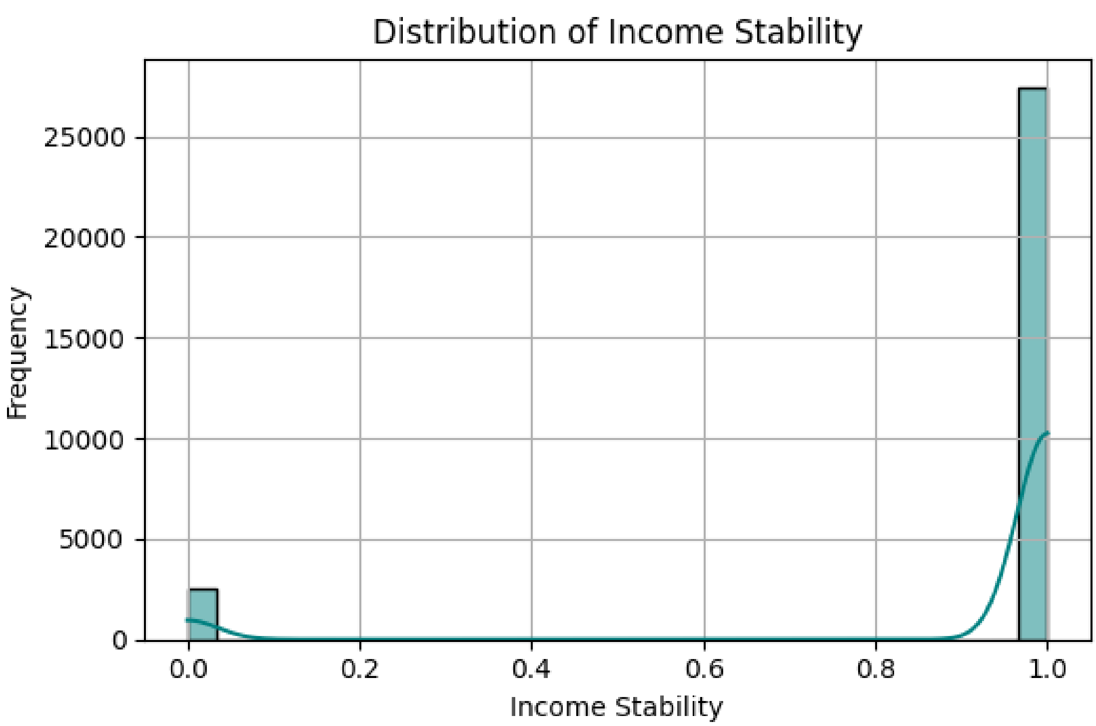
```
<Figure size 800x500 with 0 Axes>
```


Distribution of Gender
14000
12000
10000
8000
6000
4000
2000
0
Frequency
0.0
0.2
0.4
0.6
0.8
1.0
Gender

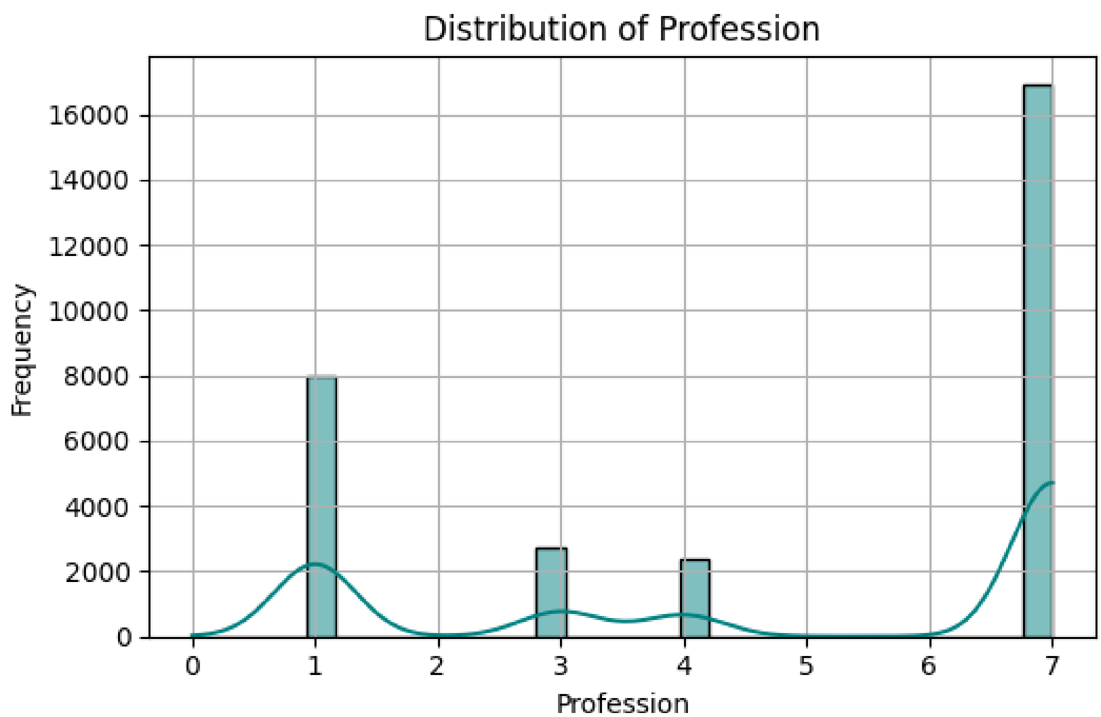

Distribution of Age
5000
4000
3000
2000
1000
0
Frequency
20
30
40
50
60
Age

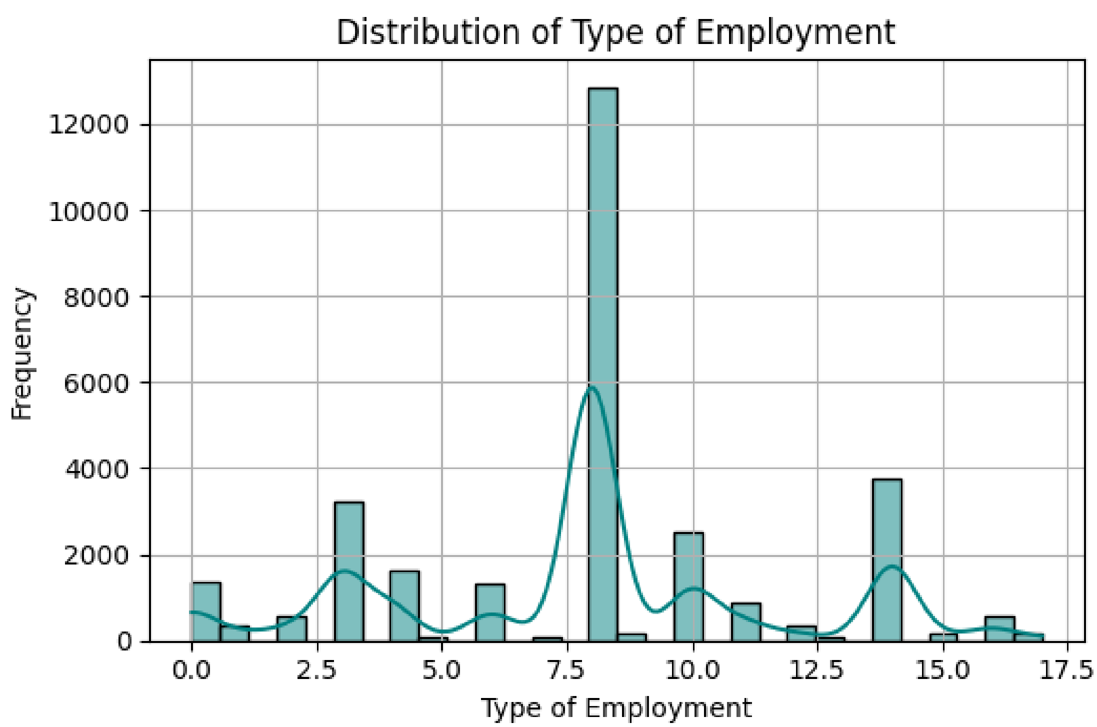


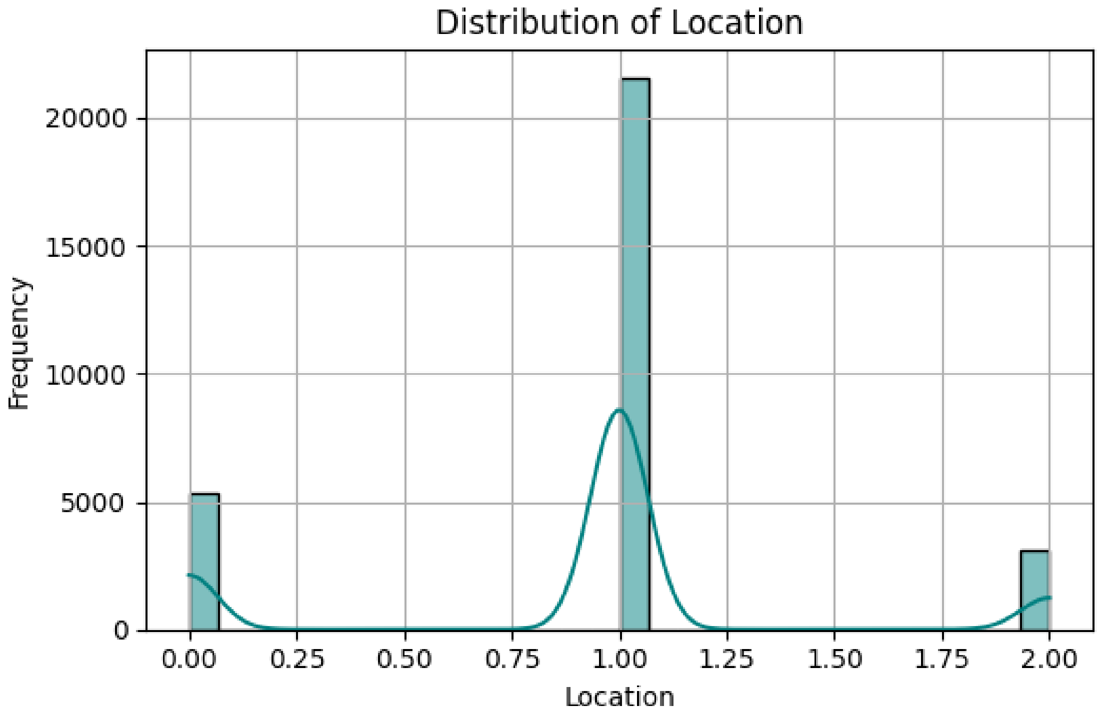
Distribution of Income (USD)
Frequency
200000
175000
150000
125000
100000
75000
50000
25000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
Income (USD)

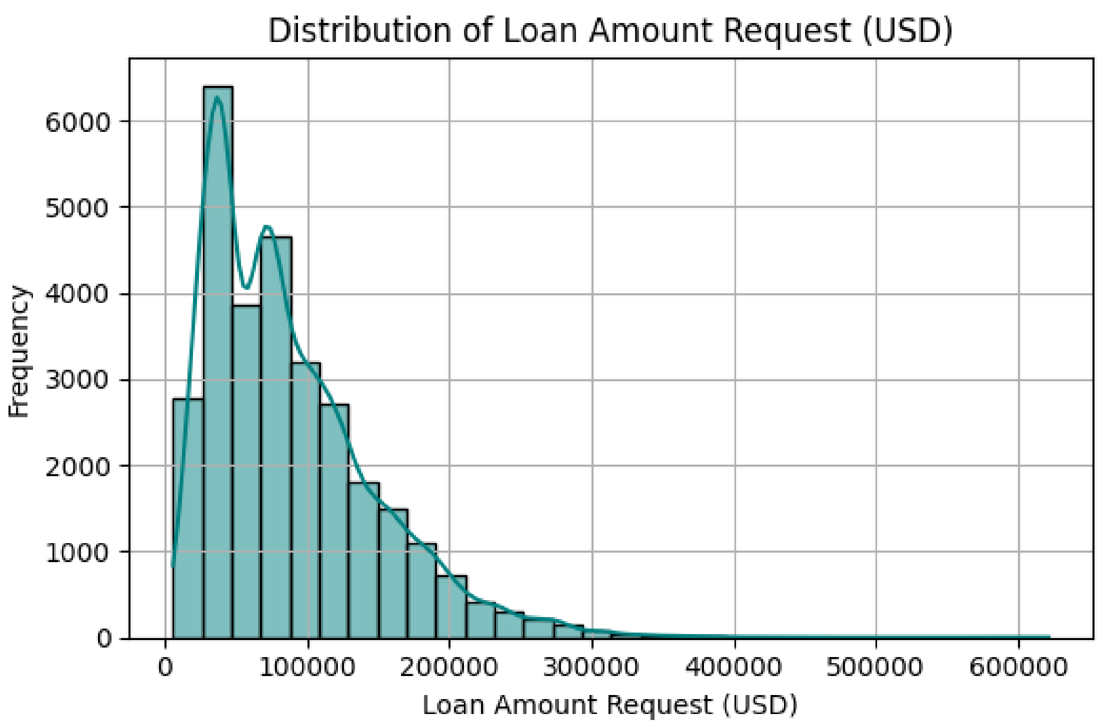


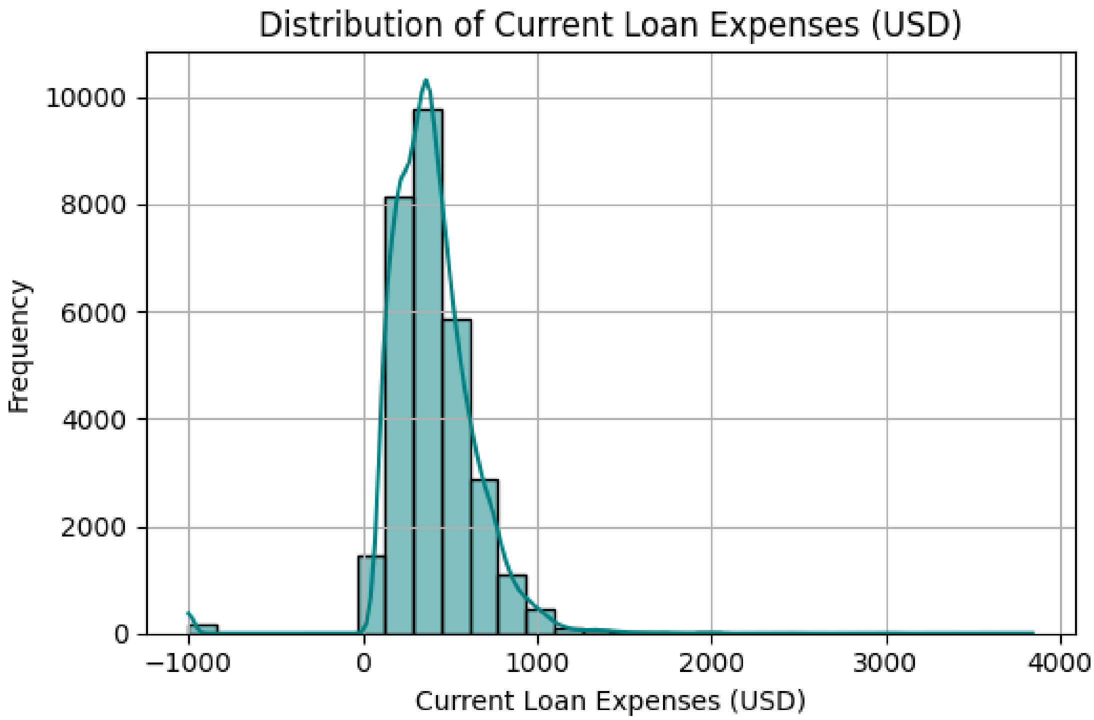
Distribution of Income Stability
Frequency
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0
Income Stability

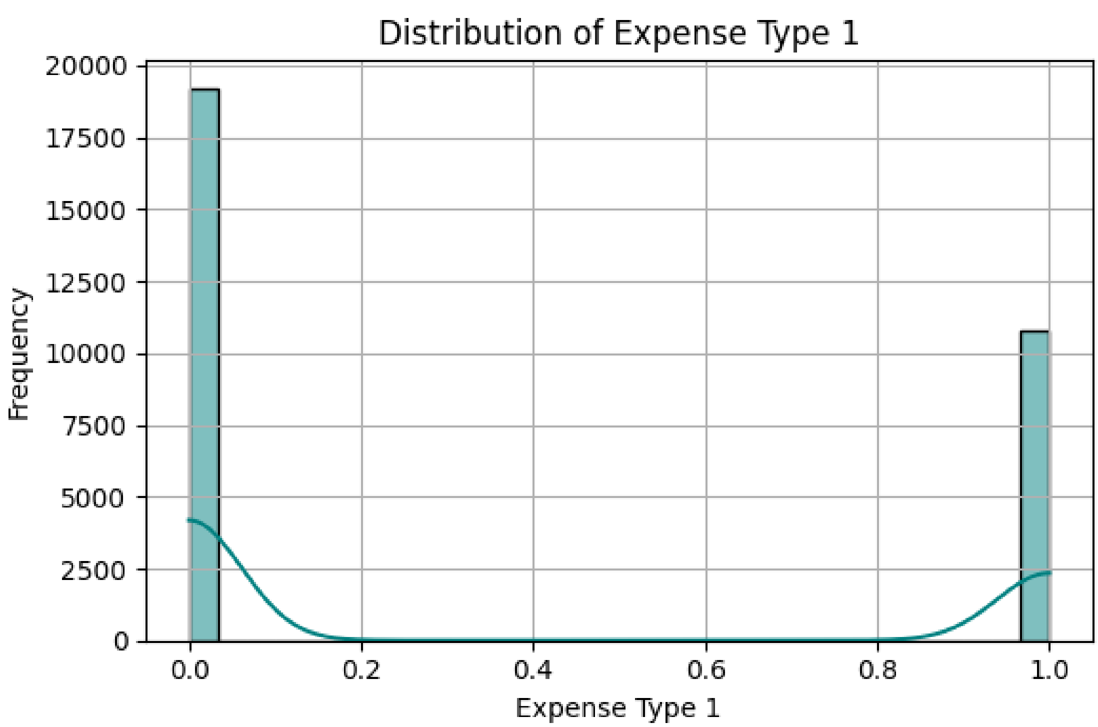

Distribution of Profession
Frequency
0
2000
4000
6000
8000
10000
12000
14000
16000
0
1
2
3
4
5
6
7
Profession

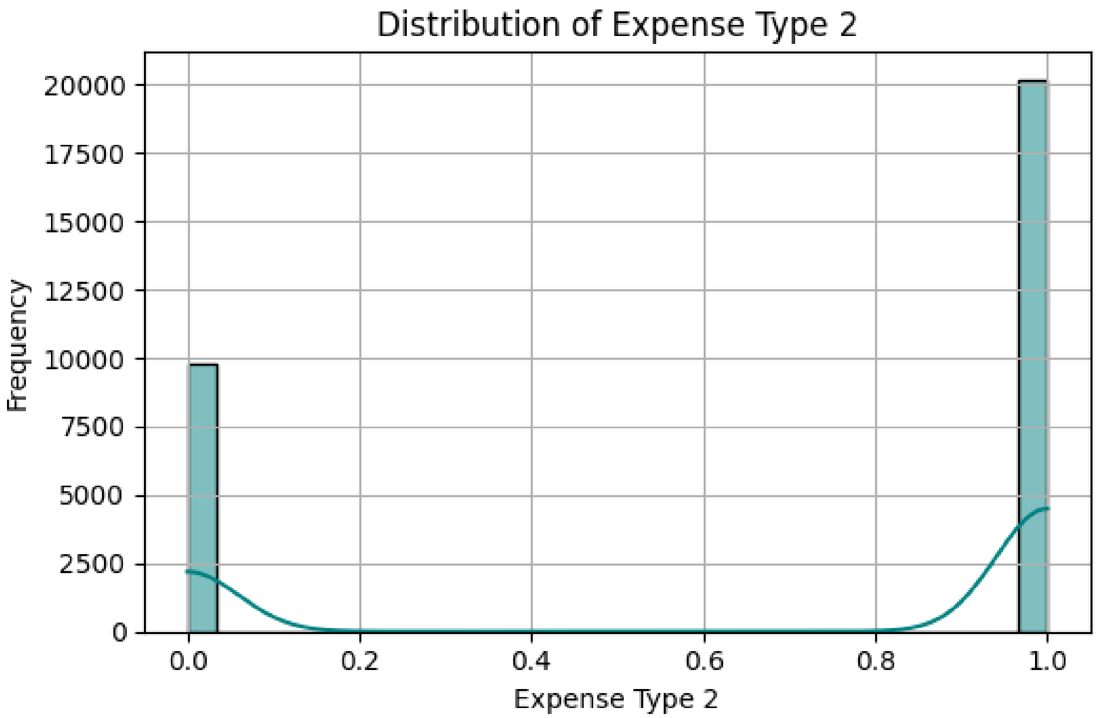

Distribution of Type of Employment
Frequency
0
2000
4000
6000
8000
10000
12000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Type of Employment

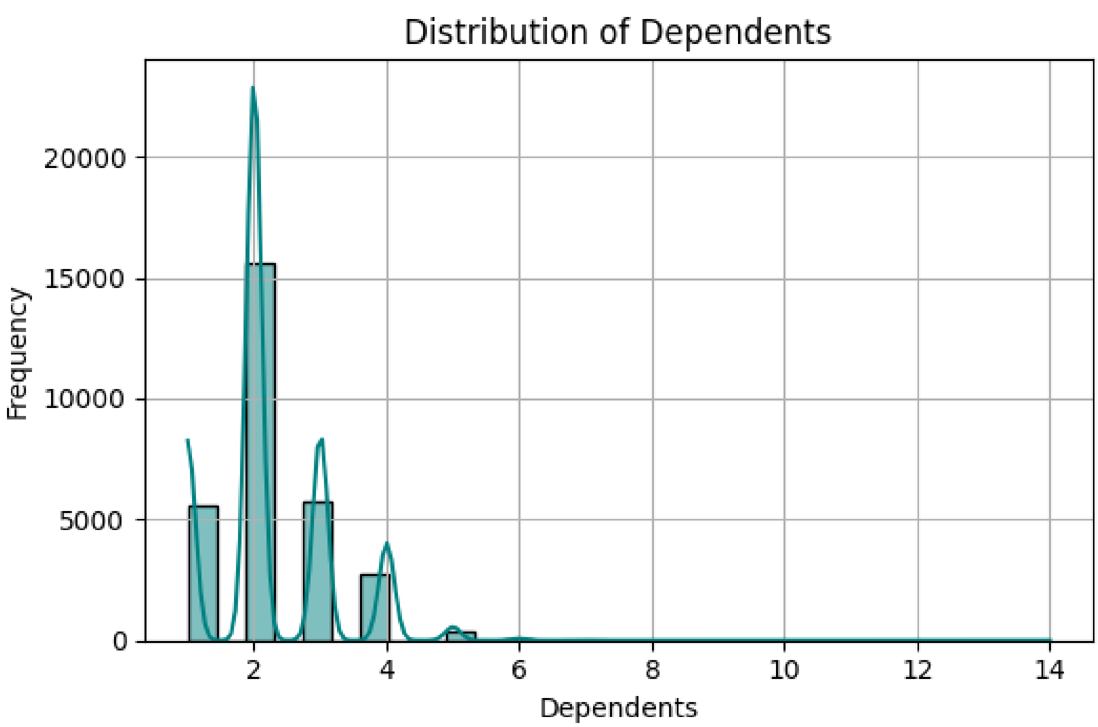

Distribution of Location
Frequency
20000
15000
10000
5000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Location

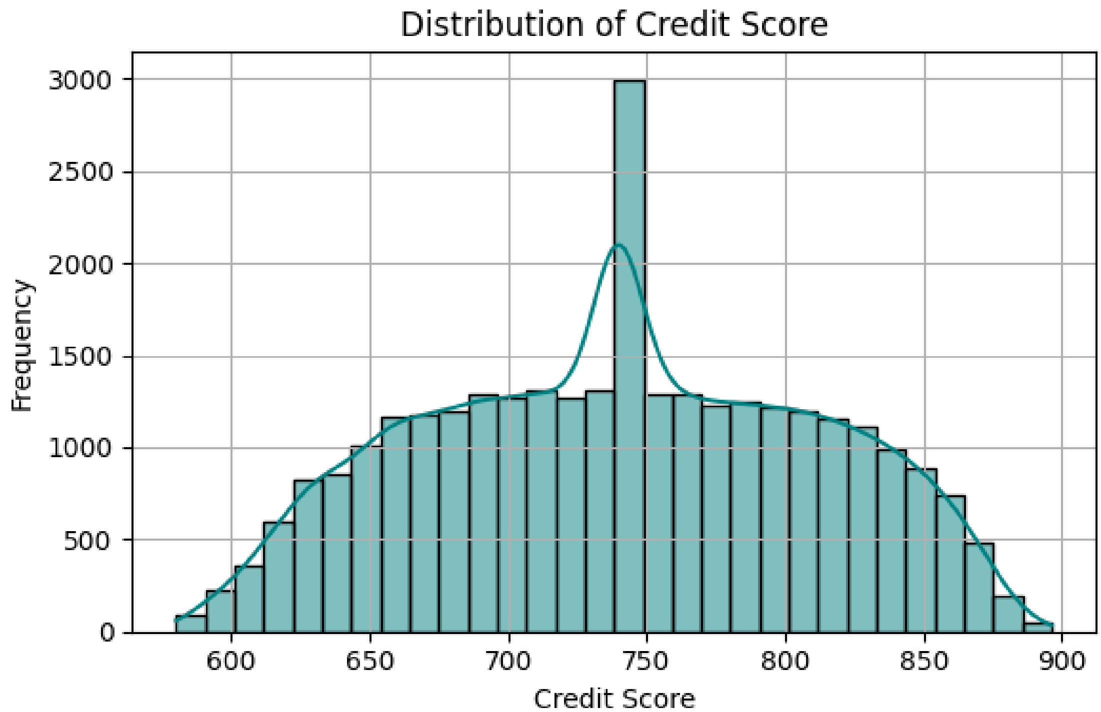

Distribution of Loan Amount Request (USD)
Frequency
6000
5000
4000
3000
2000
1000
0
0
100000
200000
300000
400000
500000
600000
Loan Amount Request (USD)

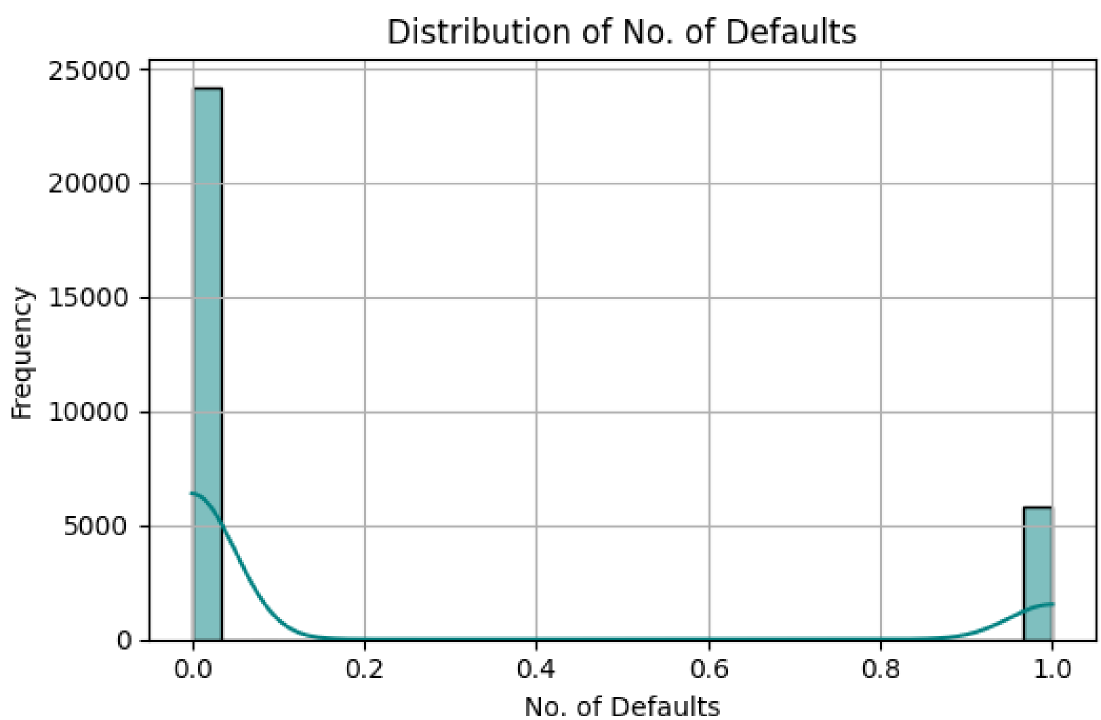

Distribution of Current Loan Expenses (USD)
Frequency
10000
8000
6000
4000
2000
0
−1000
0
1000
2000
3000
4000
Current Loan Expenses (USD)

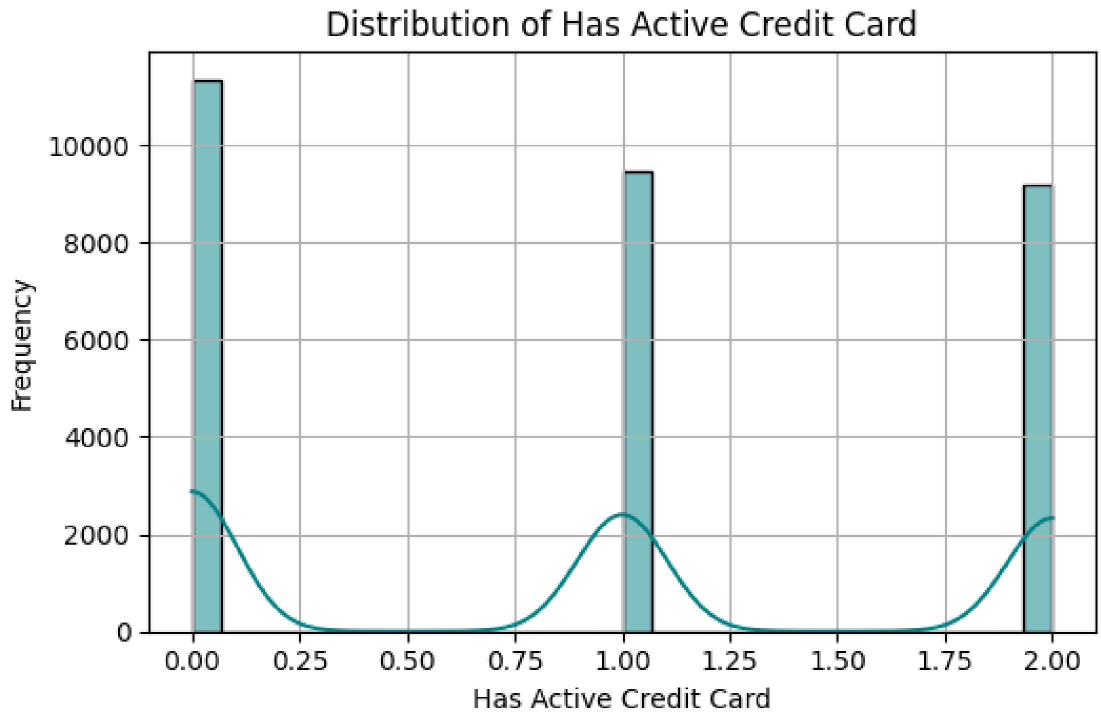

Distribution of Expense Type 1
Frequency
20000
17500
15000
12500
10000
7500
5000
2500
0
0.0
0.2
0.4
0.6
0.8
1.0
Expense Type 1


Distribution of Expense Type 2
Frequency
20000
17500
15000
12500
10000
7500
5000
2500
0
0.0
0.2
0.4
0.6
0.8
1.0
Expense Type 2

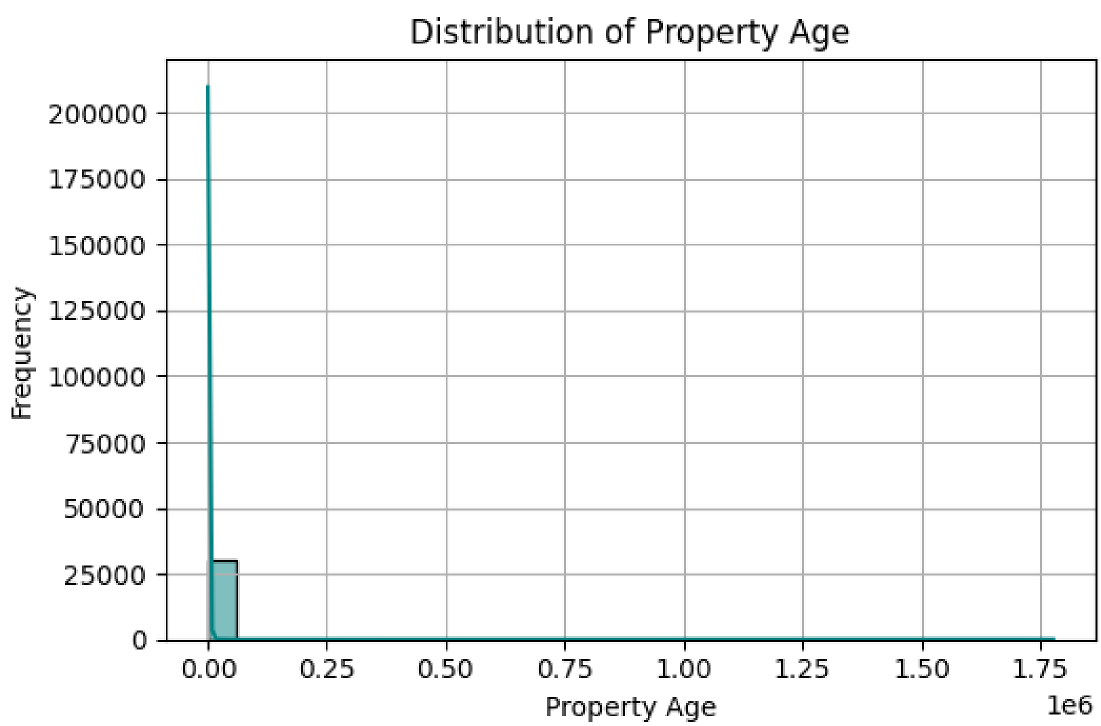

Distribution of Dependents
Frequency
20000
15000
10000
5000
0
2
4
6
8
10
12
14
Dependents


Distribution of Credit Score
3000
2500
2000
1500
1000
500
0
Frequency
600
650
700
750
800
850
900
Credit Score

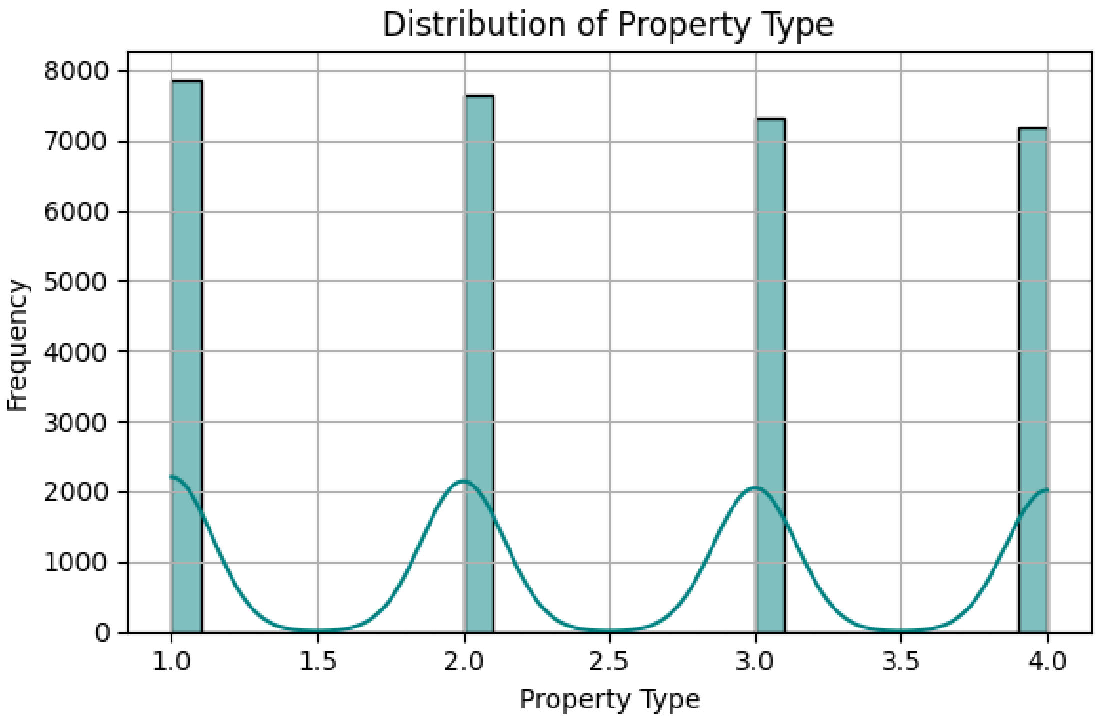

Distribution of No. of Defaults
25000
20000
15000
10000
5000
0
Frequency
0.0
0.2
0.4
0.6
0.8
1.0
No. of Defaults

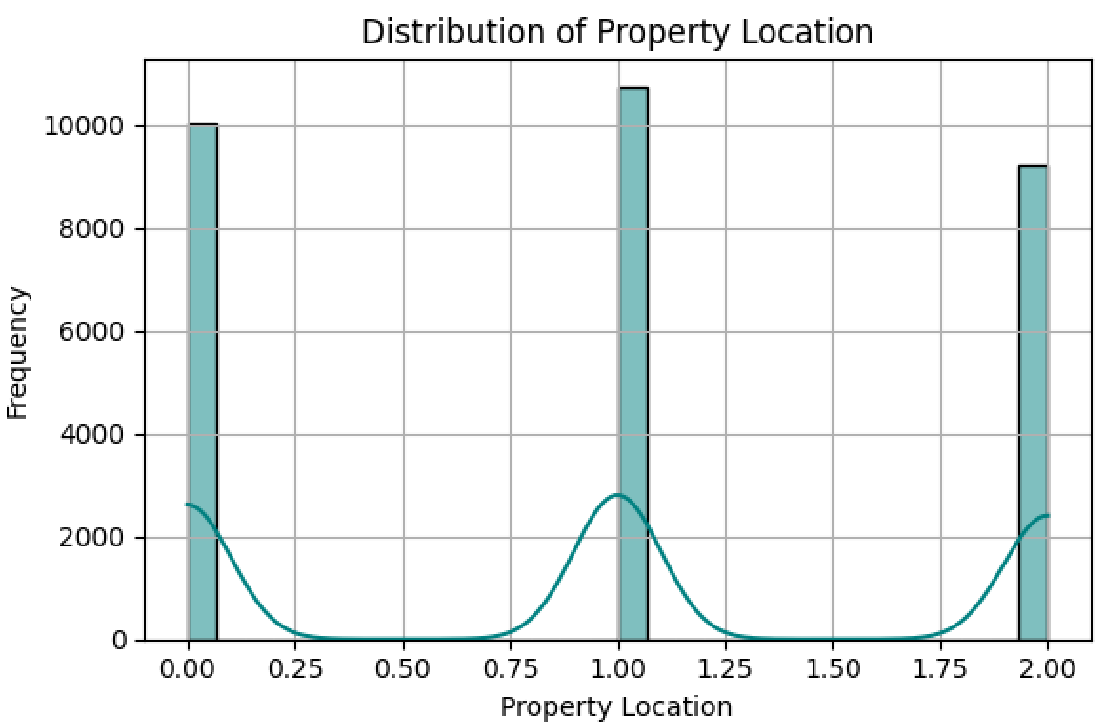


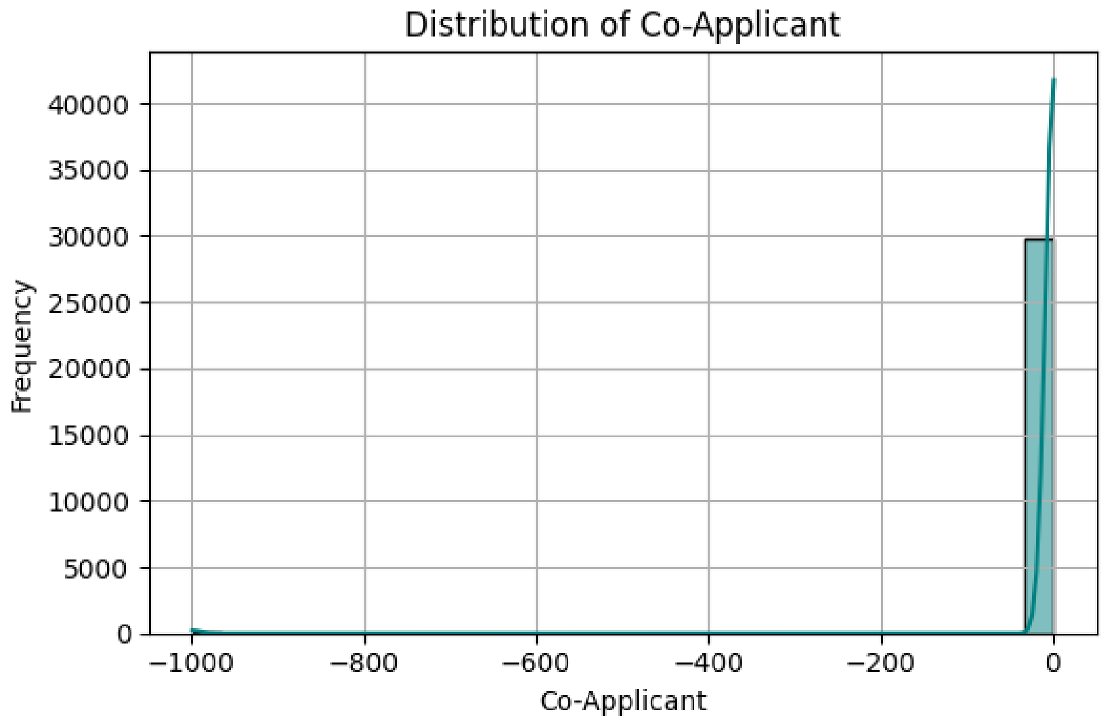
Distribution of Has Active Credit Card
Frequency
10000
8000
6000
4000
2000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Has Active Credit Card



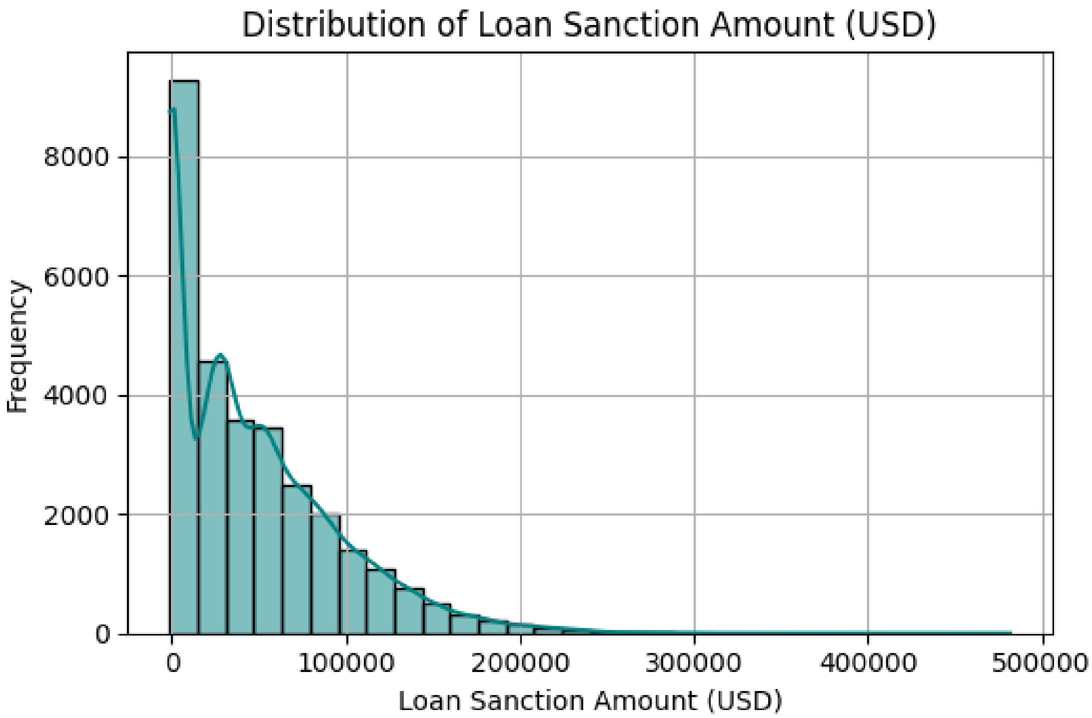
Distribution of Property Age
Frequency
200000
175000
150000
125000
100000
75000
50000
25000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Property Age
1e6


Distribution of Property Type
Frequency
8000
7000
6000
5000
4000
3000
2000
1000
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Property Type

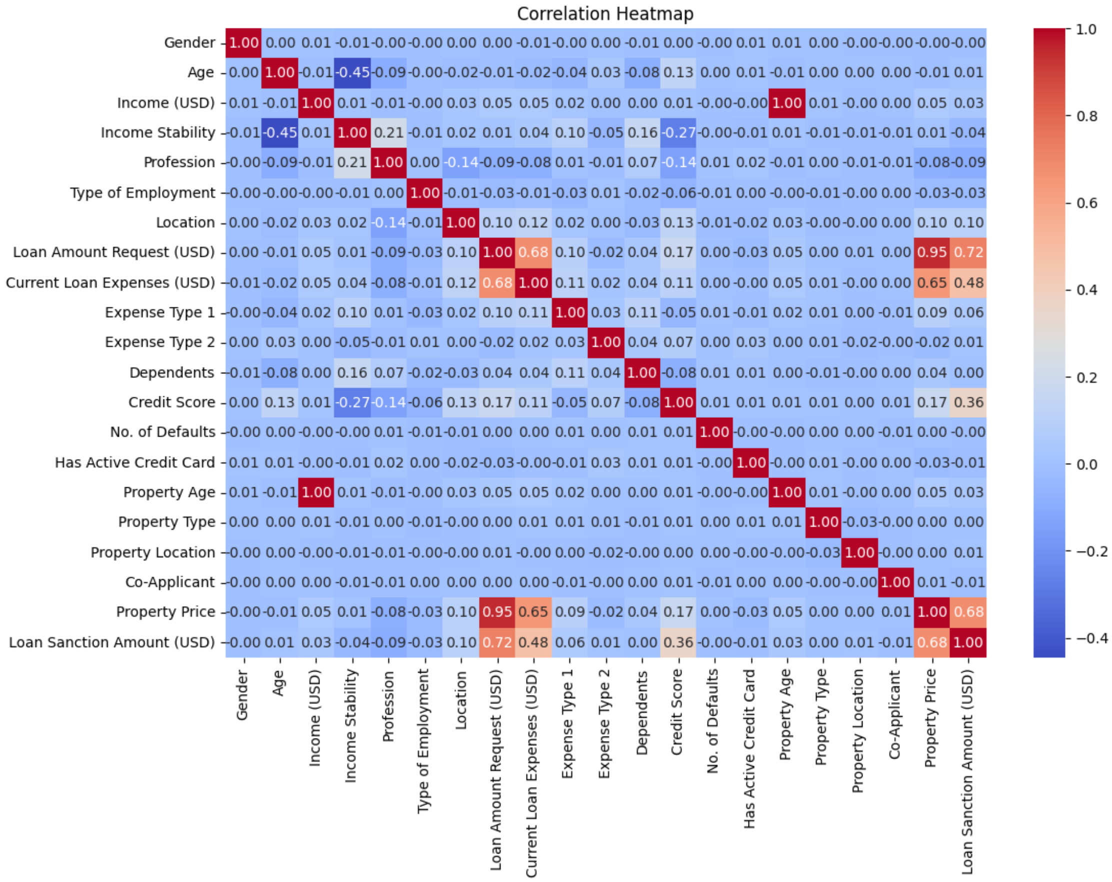

Distribution of Property Location
Frequency
10000
8000
6000
4000
2000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Property Location


Distribution of Co-Applicant
Frequency
40000
35000
30000
25000
20000
15000
10000
5000
0
−1000
−800
−600
−400
−200
0
Co-Applicant

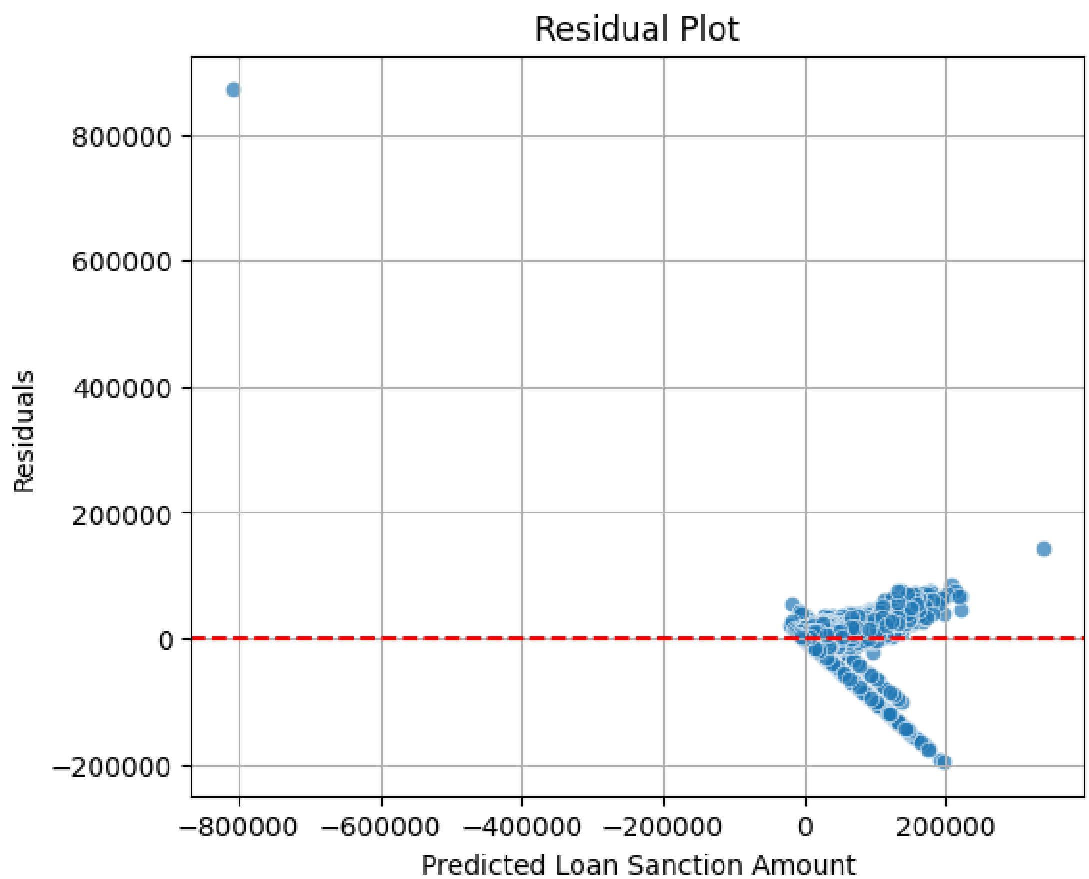

Distribution of Property Price
Frequency
7000
6000
5000
4000
3000
2000
1000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e6
Property Price



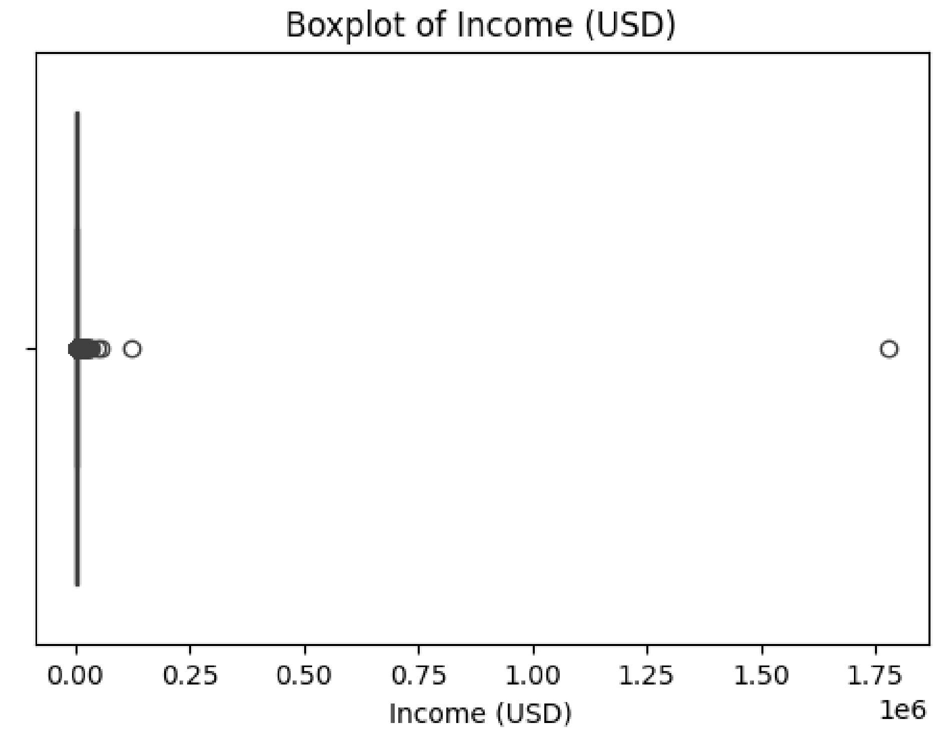
[ ]:
```
plt.figure(figsize=(12, 8))
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

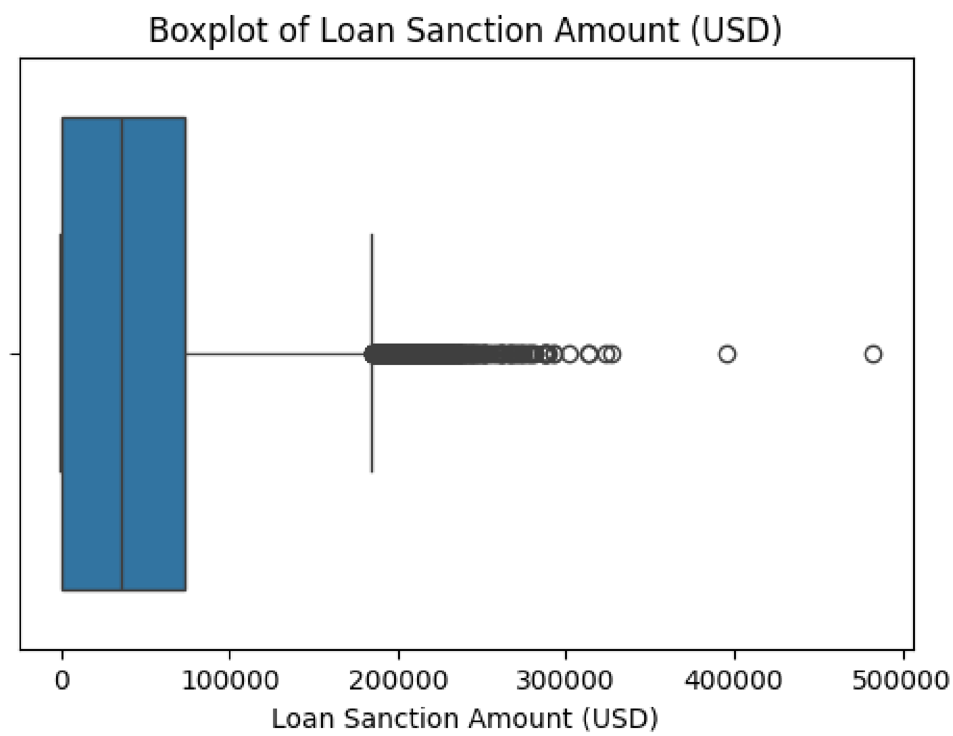


[ ]:
```
residuals = y_val - y_pred
plt.figure(figsize=(6, 5))
sns.scatterplot(x=y_pred, y=residuals, alpha=0.7)
plt.axhline(0, color='red', linestyle='--')
plt.title('Residual Plot')
plt.xlabel('Predicted Loan Sanction Amount')
plt.ylabel('Residuals')
plt.grid(True)
plt.show()
```



[ ]:
```
plt.figure(figsize=(6, 4))
sns.boxplot(x=df['Income (USD)'])
plt.title('Boxplot of Income (USD)')
plt.show()

plt.figure(figsize=(6, 4))
sns.boxplot(x=df['Loan Sanction Amount (USD)'])
plt.title('Boxplot of Loan Sanction Amount (USD)')
plt.show()
```


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


Boxplot of Loan Sanction Amount (USD)
0
100000
200000
300000
400000
500000
Loan Sanction Amount (USD)

[ ]:
```python
n = X_val.shape[0]  # number of validation samples
p = X_val.shape[1]  # number of features
r2 = r2_score(y_val, y_pred)
adjusted_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)
print("Adjusted R²:", adjusted_r2)
```

```
Adjusted R²: 0.5304169741710469
```

[ ]:
```python
from sklearn.model_selection import KFold, cross_val_score
from sklearn.metrics import make_scorer, mean_squared_error,␣
 ↪mean_absolute_error

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

mae_scores = []
mse_scores = []
r2_scores = []

for train_idx, test_idx in kf.split(X_scaled):
    X_tr, X_te = X_scaled[train_idx], X_scaled[test_idx]
    y_tr, y_te = y[train_idx], y[test_idx]
    model = LinearRegression()
    model.fit(X_tr, y_tr)
    y_pred_kf = model.predict(X_te)

    mae_scores.append(mean_absolute_error(y_te, y_pred_kf))
    mse_scores.append(mean_squared_error(y_te, y_pred_kf))
    r2_scores.append(r2_score(y_te, y_pred_kf))

print("K-Fold Cross Validation Results (5 Folds):")
for i in range(5):
    print(f"Fold {i+1}: MAE={mae_scores[i]:.2f}, MSE={mse_scores[i]:.2f},␣
 ↪RMSE={np.sqrt(mse_scores[i]):.2f}, R²={r2_scores[i]:.4f}")

print("Average:")
print(f"MAE={np.mean(mae_scores):.2f}, MSE={np.mean(mse_scores):.2f},␣
 ↪RMSE={np.sqrt(np.mean(mse_scores)):.2f}, R²={np.mean(r2_scores):.4f}")
```

```
K-Fold Cross Validation Results (5 Folds):
Fold 1: MAE=21799.22, MSE=1076701697.76, RMSE=32813.13, R²=0.5320
Fold 2: MAE=21927.37, MSE=982547266.65, RMSE=31345.61, R²=0.5678
Fold 3: MAE=22414.60, MSE=1066252447.36, RMSE=32653.52, R²=0.5397
Fold 4: MAE=21833.54, MSE=995734912.54, RMSE=31555.27, R²=0.5768
Fold 5: MAE=21024.88, MSE=881939443.57, RMSE=29697.47, R²=0.6097
Average:
```

`MAE=21799.92, MSE=1000635153.57, RMSE=31632.82, R²=0.5652`

Results Summary

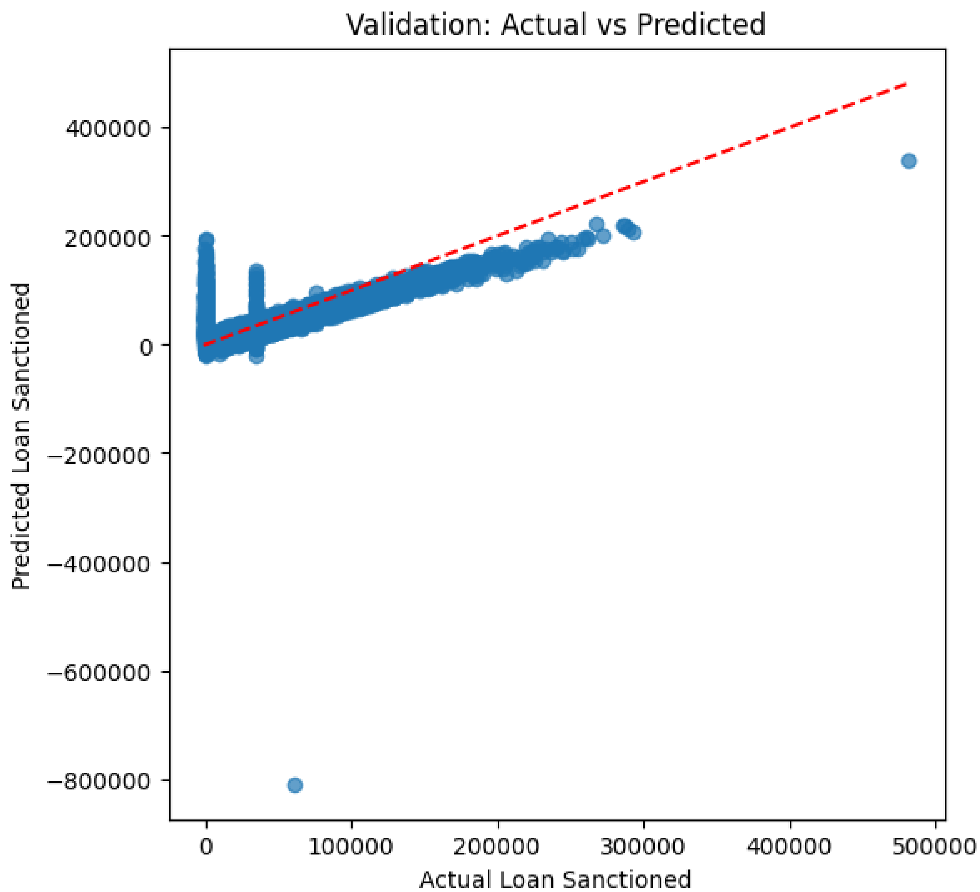
Description	**Student's Result**
Dataset Size (after preprocessing)	30000
Train/Test Split Ratio	80:20
Feature(s) Used	All numeric + encoded categoricals
Model Used	Linear Regression
Cross-Validation Used?	Yes
No. of Folds	5
MAE on Test Set	21,799.92 USD
MSE on Test Set	1,000,635,153.57 USD^2
RMSE on Test Set	31,632.82 USD
R^2 Score	0.5652
Adjusted R^2 Score	0.5304169741710469
Most Influential Features	Income, Credit Score, Loan Request
Overfitting Observed?	No significant overfitting

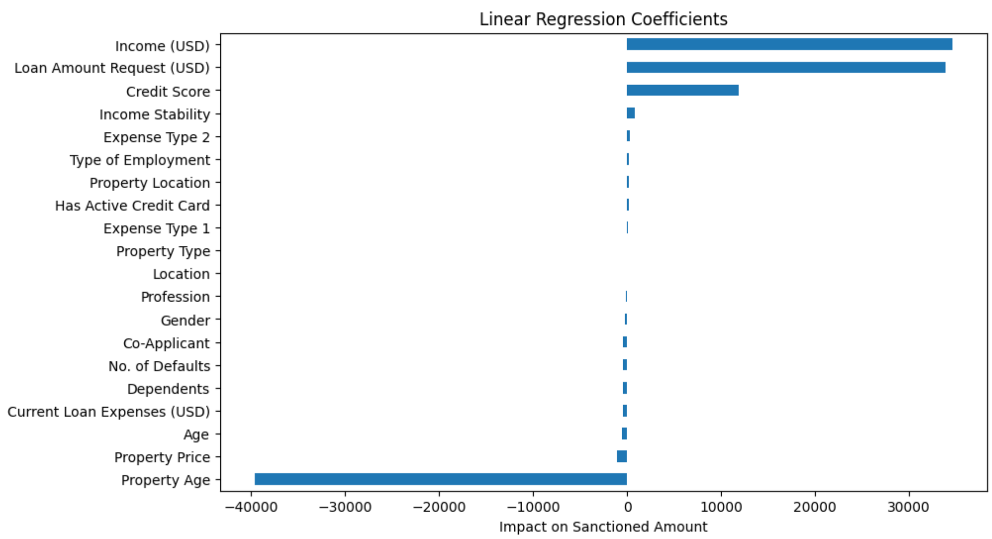
K-Fold Cross Validation (K = 5)

Fold	MAE	MSE	RMSE	R^2 Score
Fold 1	21799.22	1076701697.76	32813.13	0.5320
Fold 2	21927.37	982547266.65	31345.61	0.5678
Fold 3	22414.60	1066252447.36	32653.52	0.5397
Fold 4	21833.54	995734912.54	31555.27	0.5768
Fold 5	21024.88	881939443.57	29697.47	0.6097
Average	**21799.92**	**1000635153.57**	**31632.82**	**0.5652**

Key Visualizations

- `Actual vs Predicted Plot`
- `Feature Coefficient Bar Chart`
- `Residual Plot`





Observations

- The model explains about 56.5% of the variance in sanctioned loan amount.

- Residuals appear randomly distributed, supporting linearity assumption.
- Features like Income and Credit Score are most influential.
- Slight underfitting observed – could be improved using regularized models.

Learning Outcomes

- Understood data preprocessing and encoding for regression.
- Applied Linear Regression to a real dataset.
- Evaluated model using MAE, MSE, RMSE, R^2.
- Visualized predictions and model behavior.
- Used cross-validation for robust evaluation.

References

- Scikit-learn documentation: https://scikit-learn.org/stable/modules/linear_model.html
- StackAbuse: Linear Regression using scikit-learn
- Kaggle dataset: Predict Loan Amount Dataset