Experiment 2: Loan Amount Prediction using Linear Regression

Sudharshan Vijayaragavan
Reg no. : 3122237001054

Academic Year: 2025-2026 (Odd)

1 Aim and Objective

1.1 Aim

The aim of this experiment is to apply Linear Regression to predict the loan amount sanctioned to users. This involves a complete machine learning workflow, from data preprocessing to model evaluation and visualization.

1.2 Objective

The objective is to develop a Python program using the Scikit-learn library to build and evaluate a Linear Regression model for loan amount prediction. Additionally, the experiment focuses on using Matplotlib and Seaborn to visualize and interpret key insights and model performance metrics.

2 Libraries Used

- **Pandas**: For data manipulation and handling dataframes.
- **Numpy**: For numerical operations and array manipulation.
- **Scikit-learn**: To implement the machine learning workflow, including data splitting, feature scaling, and model training/evaluation.
- **Matplotlib & Seaborn**: For data visualization and plotting various graphs to interpret the data and model results.

3 Mathematical Description

Linear Regression models the linear relationship between a dependent variable y and one or more independent variables x_i:

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

where β_i are the coefficients and ϵ is the error term.

The coefficients can be found using the Normal Equation:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{2}$$

4 Python Code

```python
# ML_ASSGN_2.ipynb
# Author: Sudharshan Vijayaragavan
# Reg no. : 3122237001054

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# 1) Load Dataset
df = pd.read_csv('/content/drive/MyDrive/archive/train.csv')

# 2) Handle Missing Values
numerical_cols = ['Age', 'Income (USD)', 'Loan Amount Request (USD)',
                  'Current Loan Expenses (USD)', 'Credit Score',
                  'No. of Defaults', 'Property Age', 'Property Price']
for col in numerical_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')
    df[col] = df[col].fillna(df[col].median())

categorical_cols = ['Gender', 'Income Stability', 'Profession', 'Type of Employment',
                    'Location', 'Expense Type 1', 'Expense Type 2', 'Dependents',
                    'Has Active Credit Card', 'Property ID', 'Property Type',
                    'Property Location', 'Co-Applicant']
for col in categorical_cols:
    df[col] = df[col].fillna(df[col].mode()[0])

df = df.dropna()

# 3) Encode categorical variables
df_encoded = pd.get_dummies(df, columns=categorical_cols)

# 4) Standardize numerical features
df_encoded_clean = df_encoded.drop(columns=['Customer ID', 'Name'])
```

```python
scaler = StandardScaler()
df_standardized = scaler.fit_transform(df_encoded_clean)
df_standardized = pd.DataFrame(df_standardized, columns=
    df_encoded_clean.columns)

# 5) Feature Engineering
df_fe = df.copy()
df_fe['Debt_Income_Ratio'] = df_fe['Current Loan Expenses (USD)']
    / df_fe['Income (USD)']
df_fe['Many_Dependents'] = df_fe['Dependents'].apply(lambda x: 1
    if str(x).strip() in ['3+', '4', '5'] else 0)
df_fe['Property_Age_Bucket'] = pd.cut(df_fe['Property Age'], bins
    =[0,5,20,100], labels=['New','Mid','Old'])
df_fe['Log_Income'] = np.log1p(df_fe['Income (USD)'])
df_fe['Log_LoanAmount'] = np.log1p(df_fe['Loan Amount Request (USD
    )'])
df_fe['Has_Coapplicant'] = df_fe['Co-Applicant'].apply(lambda x: 0
     if str(x).strip().lower()=='no' else 1)
df_fe = df_fe.drop(columns=['Customer ID', 'Name'])
df_fe = pd.get_dummies(df_fe, columns=['Property_Age_Bucket'] +
    categorical_cols)

# 6) Train/Test Split
target = 'Loan Amount Request (USD)'
X = df_fe.drop(columns=[target])
y = df_fe[target]
X_trainval, X_test, y_trainval, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
X_train, X_val, y_train, y_val = train_test_split(X_trainval,
    y_trainval, test_size=0.25, random_state=42)

# 7) Train Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)

# 8) Evaluate Model
y_val_pred = model.predict(X_val)
y_test_pred = model.predict(X_test)

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)

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)

print(f"Validation MAE: {mae_val:.2f}, R2: {r2_val:.4f}")
print(f"Test MAE: {mae_test:.2f}, R2: {r2_test:.4f}")
```

5 Included Plots

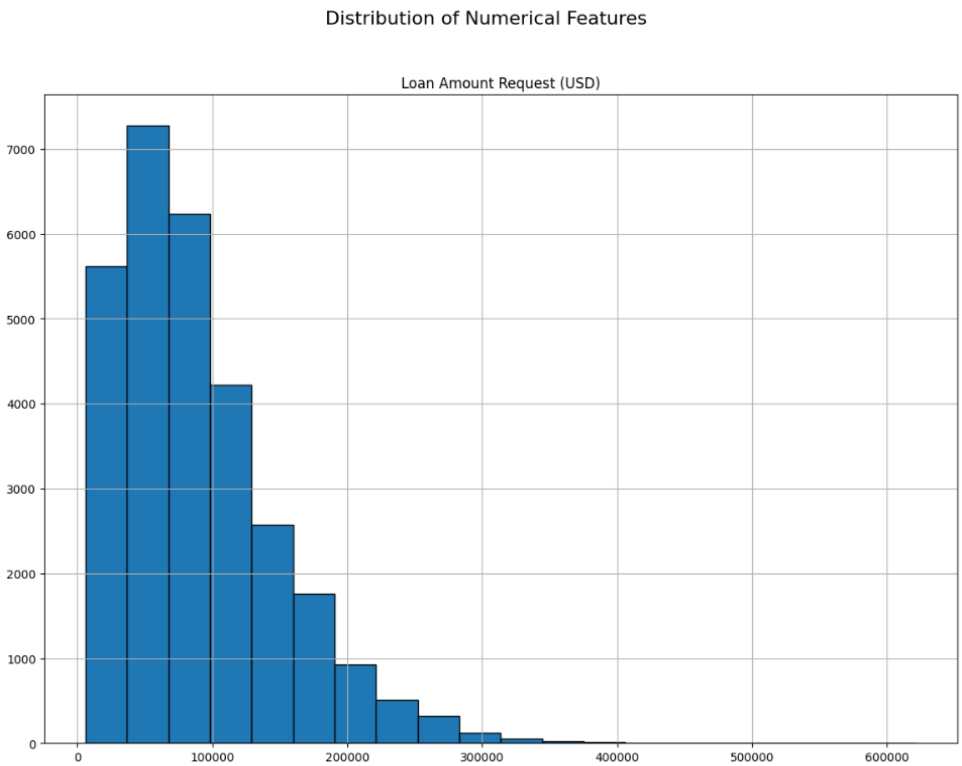

Figure 1: Distribution of Loan Amount Request.

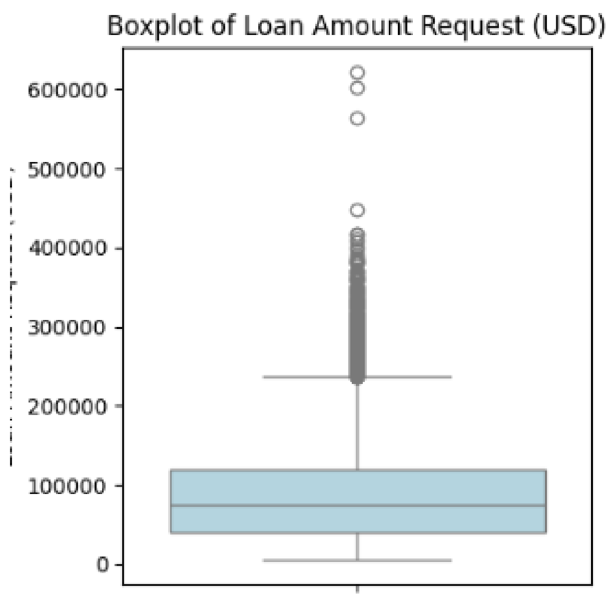

Figure 2: Boxplot of Loan Amount Request, showing outliers.

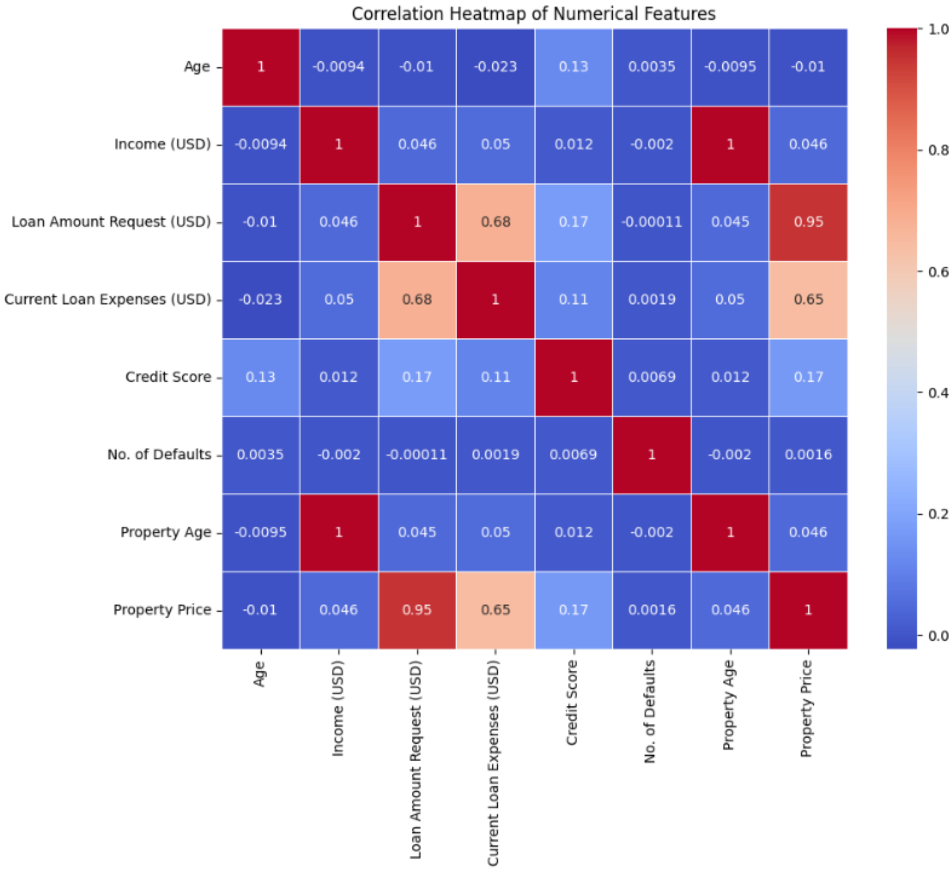

Figure 3: Correlation Heatmap of numerical features.

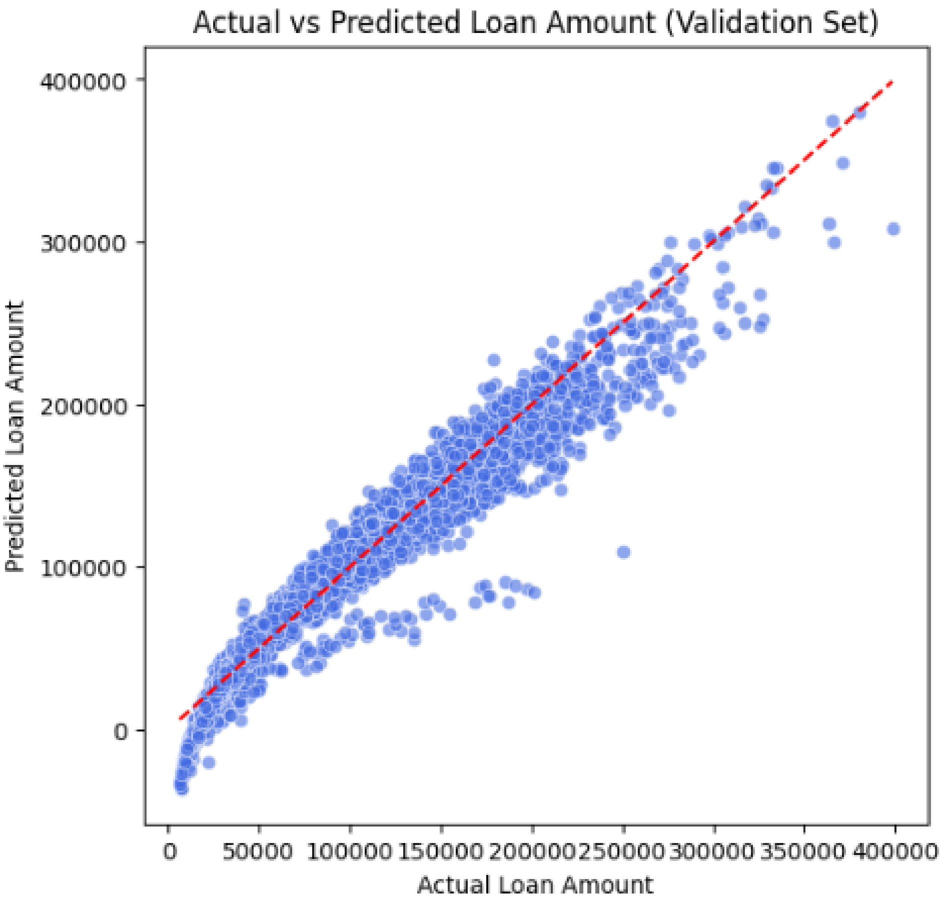

Figure 4: Actual vs Predicted Loan Amount (Validation Set).

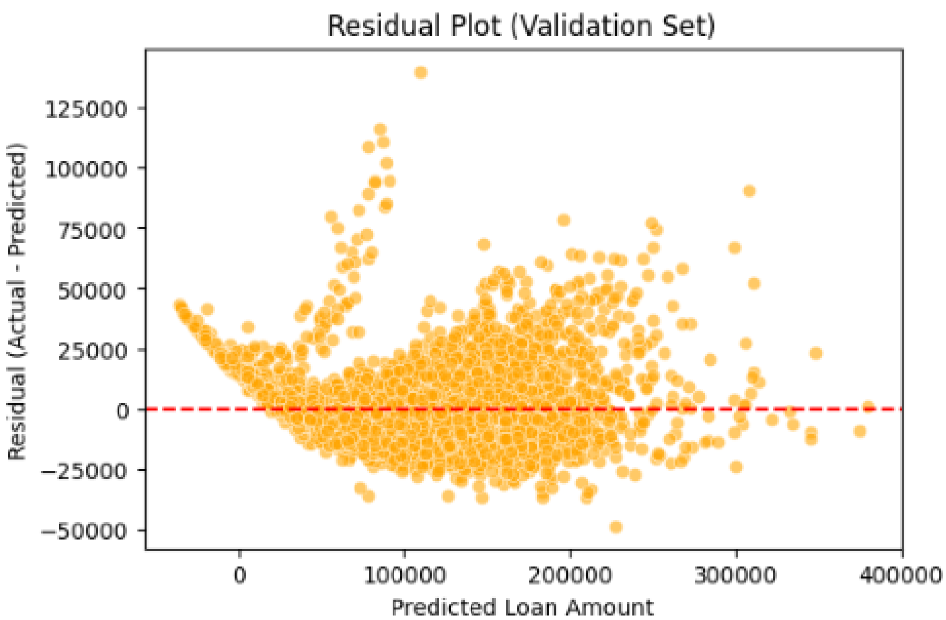

Figure 5: Residual Plot for Validation Set.

6 Results Table

Description	Result
Dataset Size (after preprocessing)	20000 rows, 775 columns
Train/Test Split Ratio	80% Train+Validation / 20% Test
Features Used	All preprocessed features, including engineered features (Debt_Income_Ratio, Log_Income)
Model Used	Linear Regression
MAE on Test Set	45788.19
MSE on Test Set	2.91×10^9
RMSE on Test Set	53959.08
R2 Score on Test Set	0.0816
Most Influential Features	Debt_Income_Ratio, Log_LoanAmount
Observations from Residual Plot	Non-random pattern, indicating heteroscedasticity

Table 1: Summary of Results for Loan Amount Prediction

7 Best Practices

- Handle missing values and encode categorical features properly.
- Perform exploratory data analysis using histograms and correlation heatmaps.
- Apply feature engineering to create meaningful features like Debt_Income_Ratio.
- Split data into train, validation, and test sets to evaluate model performance unbiasedly.

8 Learning Outcomes

I learned the full machine learning pipeline for regression tasks: data preprocessing, EDA, feature engineering, model training, evaluation, and interpretation. Residual analysis and feature importance visualization helped in understanding model performance and limitations.