

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 :

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n + \epsilon \quad (1)$$

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

The coefficients can be found using the Normal Equation:

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

4 Python Code

```
1 # ML_ASSGN_2.ipynb
2 # Author: Sudharshan Vijayaragavan
3 # Reg no. : 3122237001054
4
5 import pandas as pd
6 import numpy as np
7 import matplotlib.pyplot as plt
8 import seaborn as sns
9 from sklearn.preprocessing import StandardScaler
10 from sklearn.model_selection import train_test_split
11 from sklearn.linear_model import LinearRegression
12 from sklearn.metrics import mean_absolute_error,
    mean_squared_error, r2_score
13
14 # 1) Load Dataset
15 df = pd.read_csv('/content/drive/MyDrive/archive/train.csv')
16
17 # 2) Handle Missing Values
18 numerical_cols = ['Age', 'Income (USD)', 'Loan Amount Request (USD
    )',
19                   'Current Loan Expenses (USD)', 'Credit Score',
20                   'No. of Defaults', 'Property Age', 'Property
    Price']
21 for col in numerical_cols:
22     df[col] = pd.to_numeric(df[col], errors='coerce')
23     df[col] = df[col].fillna(df[col].median())
24
25 categorical_cols = ['Gender', 'Income Stability', 'Profession', '
    Type of Employment',
26                    'Location', 'Expense Type 1', 'Expense Type 2'
    , 'Dependents',
27                    'Has Active Credit Card', 'Property ID', '
    Property Type',
28                    'Property Location', 'Co-Applicant']
29 for col in categorical_cols:
30     df[col] = df[col].fillna(df[col].mode()[0])
31
32 df = df.dropna()
33
34 # 3) Encode categorical variables
35 df_encoded = pd.get_dummies(df, columns=categorical_cols)
36
37 # 4) Standardize numerical features
38 df_encoded_clean = df_encoded.drop(columns=['Customer ID', 'Name'
    ])
```

```

39 scaler = StandardScaler()
40 df_standardized = scaler.fit_transform(df_encoded_clean)
41 df_standardized = pd.DataFrame(df_standardized, columns=
    df_encoded_clean.columns)
42
43 # 5) Feature Engineering
44 df_fe = df.copy()
45 df_fe['Debt_Income_Ratio'] = df_fe['Current Loan Expenses (USD)']
    / df_fe['Income (USD)']
46 df_fe['Many_Dependents'] = df_fe['Dependents'].apply(lambda x: 1
    if str(x).strip() in ['3+', '4', '5'] else 0)
47 df_fe['Property_Age_Bucket'] = pd.cut(df_fe['Property Age'], bins
    =[0,5,20,100], labels=['New','Mid','Old'])
48 df_fe['Log_Income'] = np.log1p(df_fe['Income (USD)'])
49 df_fe['Log_LoanAmount'] = np.log1p(df_fe['Loan Amount Request (USD
    )'])
50 df_fe['Has_Coapplicant'] = df_fe['Co-Applicant'].apply(lambda x: 0
    if str(x).strip().lower()=='no' else 1)
51 df_fe = df_fe.drop(columns=['Customer ID', 'Name'])
52 df_fe = pd.get_dummies(df_fe, columns=['Property_Age_Bucket'] +
    categorical_cols)
53
54 # 6) Train/Test Split
55 target = 'Loan Amount Request (USD)'
56 X = df_fe.drop(columns=[target])
57 y = df_fe[target]
58 X_trainval, X_test, y_trainval, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
59 X_train, X_val, y_train, y_val = train_test_split(X_trainval,
    y_trainval, test_size=0.25, random_state=42)
60
61 # 7) Train Linear Regression Model
62 model = LinearRegression()
63 model.fit(X_train, y_train)
64
65 # 8) Evaluate Model
66 y_val_pred = model.predict(X_val)
67 y_test_pred = model.predict(X_test)
68
69 mae_val = mean_absolute_error(y_val, y_val_pred)
70 mse_val = mean_squared_error(y_val, y_val_pred)
71 rmse_val = np.sqrt(mse_val)
72 r2_val = r2_score(y_val, y_val_pred)
73
74 mae_test = mean_absolute_error(y_test, y_test_pred)
75 mse_test = mean_squared_error(y_test, y_test_pred)
76 rmse_test = np.sqrt(mse_test)
77 r2_test = r2_score(y_test, y_test_pred)
78
79 print(f"Validation MAE: {mae_val:.2f}, R2: {r2_val:.4f}")
80 print(f"Test MAE: {mae_test:.2f}, R2: {r2_test:.4f}")

```

5 Included Plots

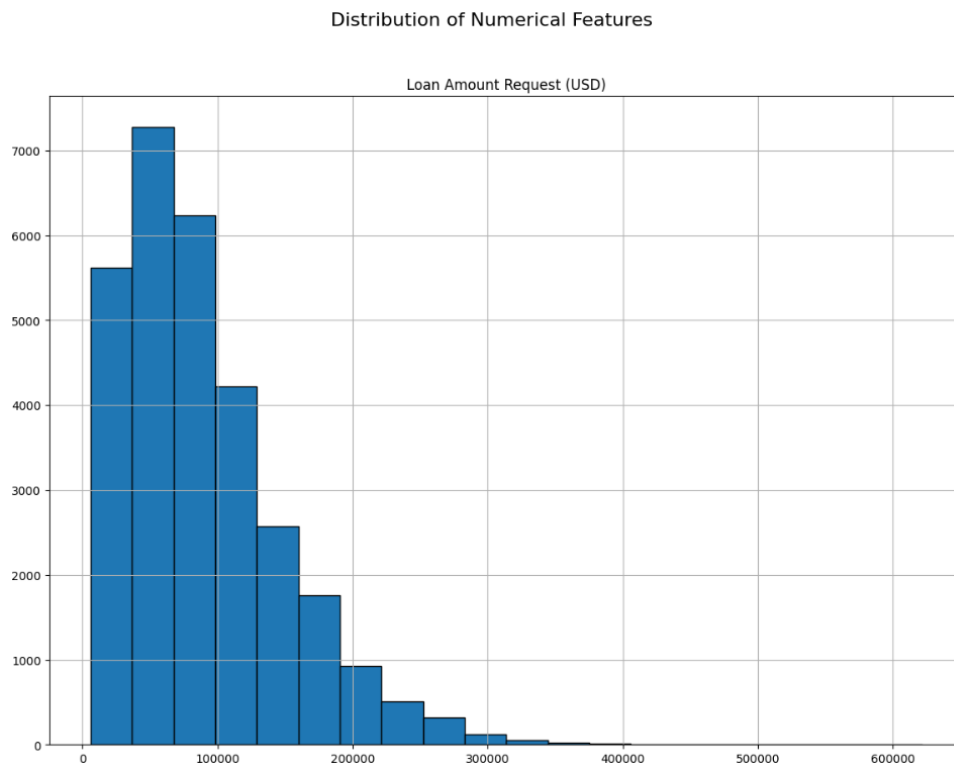


Figure 1: Distribution of Loan Amount Request.

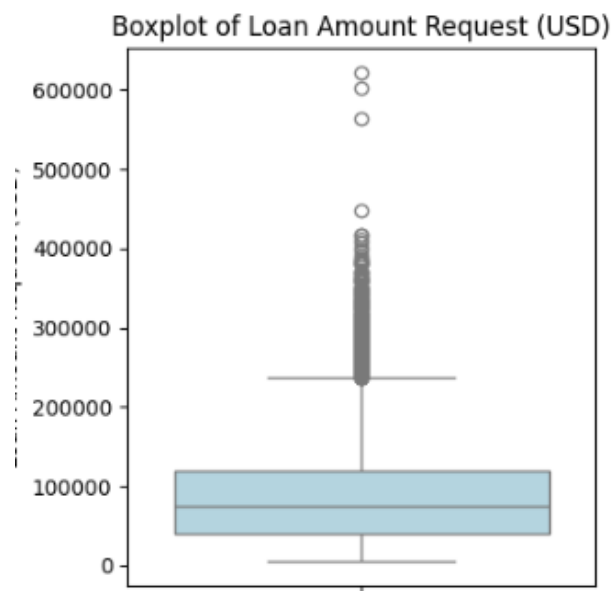


Figure 2: Boxplot of Loan Amount Request, showing outliers.

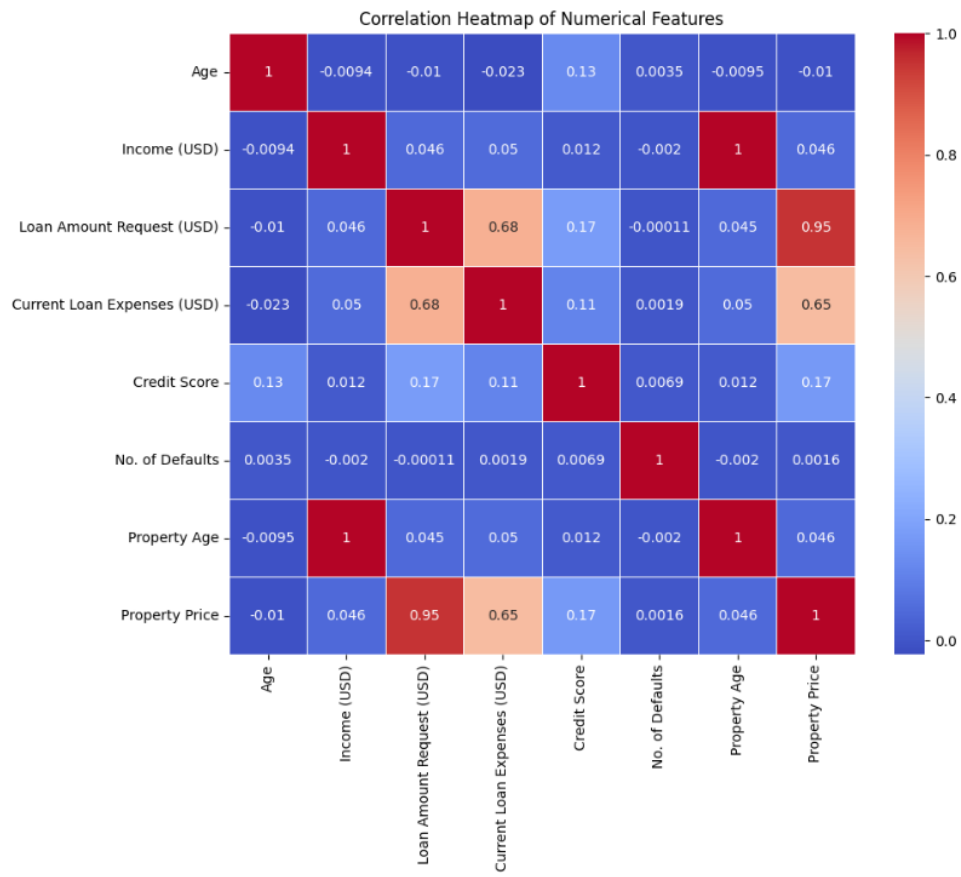


Figure 3: Correlation Heatmap of numerical features.

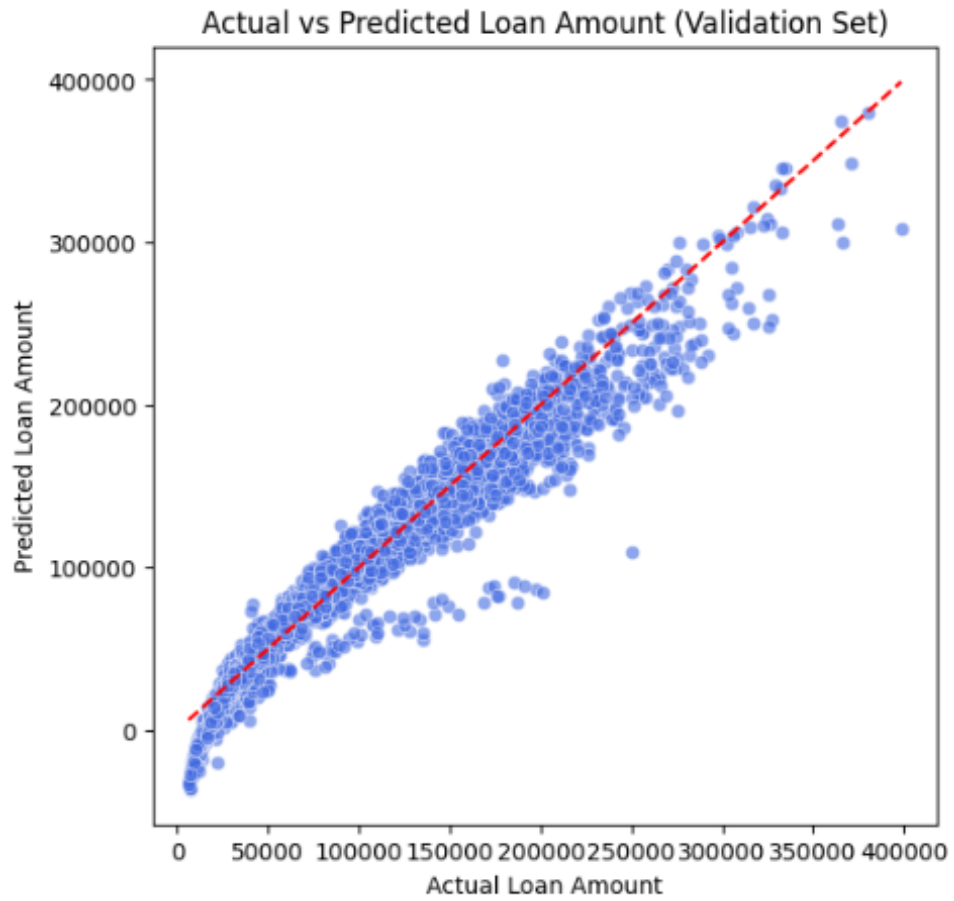


Figure 4: Actual vs Predicted Loan Amount (Validation Set).

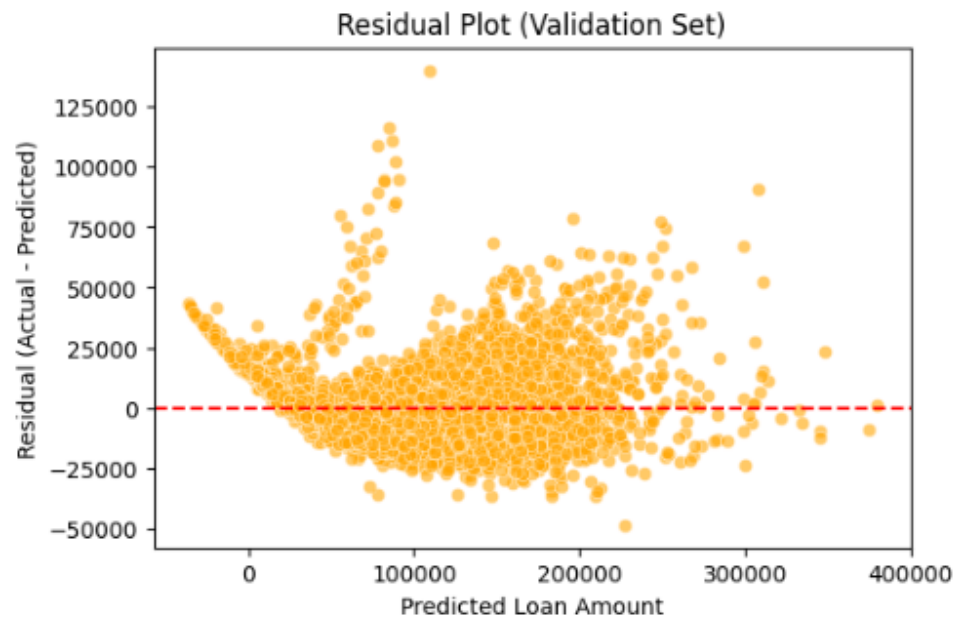


Figure 5: Residual Plot for Validation Set.

6 Cross-Validation Results

The model was evaluated using 5-fold cross-validation. The table below summarizes the fold-wise performance along with the mean values across folds.

Fold	MAE	MSE	RMSE	R2
1	9970.74	2.13×10^8	14600.02	0.9407
2	10032.56	2.09×10^8	14453.88	0.9389
3	10293.25	9.34×10^8	30555.89	0.7338
4	9729.49	1.90×10^8	13768.64	0.9468
5	10023.68	2.05×10^8	14311.39	0.9431
Mean	10009.94	3.50×10^8	17537.97	0.9007

Table 1: 5-Fold Cross-Validation Results for Loan Amount Prediction Model

7 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 2: Summary of Results for Loan Amount Prediction

8 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.

9 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.