

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

Degree & Branch	B.E. Computer Science & Engineering	Semester	V
Subject Code & Name	ICS1512 - Machine Learning Algorithms Laboratory		
Academic year	2025-2026 (Odd)	Batch: 2023-2028	<b>Due date: 29/7/25</b>

**Experiment 2: Loan Amount Prediction using Linear Regression**

## 1 Aim:

Apply Linear Regression to predict the loan amount sanctioned to users using the dataset provided. Visualize and interpret the results to gain insights into the model performance.

## 2 Libraries used:

- Pandas
- Numpy
- Matplotlib
- Scikit-learn
- Seaborn

## 3 Objective:

To evaluate the performance of a Linear Regression model in predicting loan sanction amounts based on applicant and property features. Model evaluation is done using 5-fold cross-validation and standard metrics (MAE, MSE, RMSE, R<sup>2</sup>).

## 4 Mathematical Description:

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

Where:

- $\hat{y}$  = predicted loan amount
- $\beta_0$  = intercept
- $\beta_i$  = coefficient of feature  $x_i$

We evaluate using the following metrics:

$$\begin{aligned} \text{MAE} &= \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \\ \text{MSE} &= \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \\ \text{RMSE} &= \sqrt{\text{MSE}} \\ R^2 &= 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2} \end{aligned}$$

The objective is to minimize the cost function:

$$J(\beta_0, \beta_1) = \frac{1}{n} \sum_{i=1}^n (y_i - (\beta_0 + \beta_1 x_i))^2$$

## 5 Code with Plot

```
# 1. LOAD THE DATASET
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

data = pd.read_csv('/content/drive/MyDrive/ml-lab/train.csv')
df = pd.DataFrame(data)
print(data.head())
```

**OUTPUT:**

	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 \
0 1933.05 4 Rural 1
1 4952.91 2 Rural 1
2 988.19 2 Urban 0
3 NaN 2 Semi-Urban 1
4 2614.77 4 Semi-Urban 1

Property Price Loan Sanction Amount (USD)
0 119933.46 54607.18
1 54791.00 37469.98
2 72440.58 36474.43
3 121441.51 56040.54
4 208567.91 74008.28

[5 rows x 24 columns]

```

Figure 1: Dataset loaded

```

# 2. PREPROCESS THE DATA
# HANDLING MISSING VALUES
df['Credit Score'] = df['Credit Score'].fillna(df['Credit Score'].mean())
df['Current Loan Expenses (USD)'] = df['Current Loan Expenses (USD)'].fillna(df['Current Loan Expenses (USD)'].mean())
df['Loan Sanction Amount (USD)'] = df['Loan Sanction Amount (USD)'].fillna(df['Loan Sanction Amount (USD)'].mean())
df['Income Stability'] = df['Income Stability'].fillna(df['Income Stability'].mode()[0])

```

OUTPUT:

```

Property Age 0
Property Type 0
Property Location 0
Co-Applicant 0
Property Price 0
Loan Sanction Amount (USD) 0
dtype: int64

```

Figure 2: Missing values removed

```
# ENCODING CATEGORICAL VARIABLES
df = df.drop(['Customer ID', 'Name', 'Property ID'], axis=1)
df = pd.get_dummies(df, columns=['Gender', 'Profession', 'Location',
'Property Location'], drop_first=True)

# ordinal variables = map
df['Income Stability'] = df['Income Stability'].map({'Low': 0, 'High': 1})
df['Expense Type 1'] = df['Expense Type 1'].map({'N': 0, 'Y': 1})
df['Has Active Credit Card'] = df['Has Active Credit Card'].map({
'Unpossessed': 0, 'Inactive': 1, 'Active': 2})

# high cardinality categorical variable = frequency encoding
emp_freq = df['Type of Employment'].value_counts(normalize=True)
df['Type of Employment Encoded'] = df['Type of Employment'].map(emp_freq)
df.drop('Type of Employment', axis=1, inplace=True)
```

### OUTPUT:

Data columns (total 28 columns):			
#	Column	Non-Null Count	Dtype
---	-----	-----	-----
0	Age	24960 non-null	int64
1	Income (USD)	24960 non-null	float64
2	Income Stability	24960 non-null	int64
3	Loan Amount Request (USD)	24960 non-null	float64
4	Current Loan Expenses (USD)	24960 non-null	float64
5	Expense Type 1	24960 non-null	int64
6	Expense Type 2	24960 non-null	int64
7	Dependents	24960 non-null	float64
8	Credit Score	24960 non-null	float64
9	No. of Defaults	24960 non-null	int64

Figure 3: Categorical variables encoded

```
# 3. EDA
sns.histplot(df['Loan Amount Request (USD)'], kde=True)
sns.boxplot(x='Gender_M', y='Loan Sanction Amount (USD)', data=df)

num_cols = df.select_dtypes(include=['float64']).columns
df[num_cols].hist(figsize=(15, 12), bins=30)
plt.tight_layout()

plt.figure(figsize=(15, 10))
sns.heatmap(df[num_cols].corr(), annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```

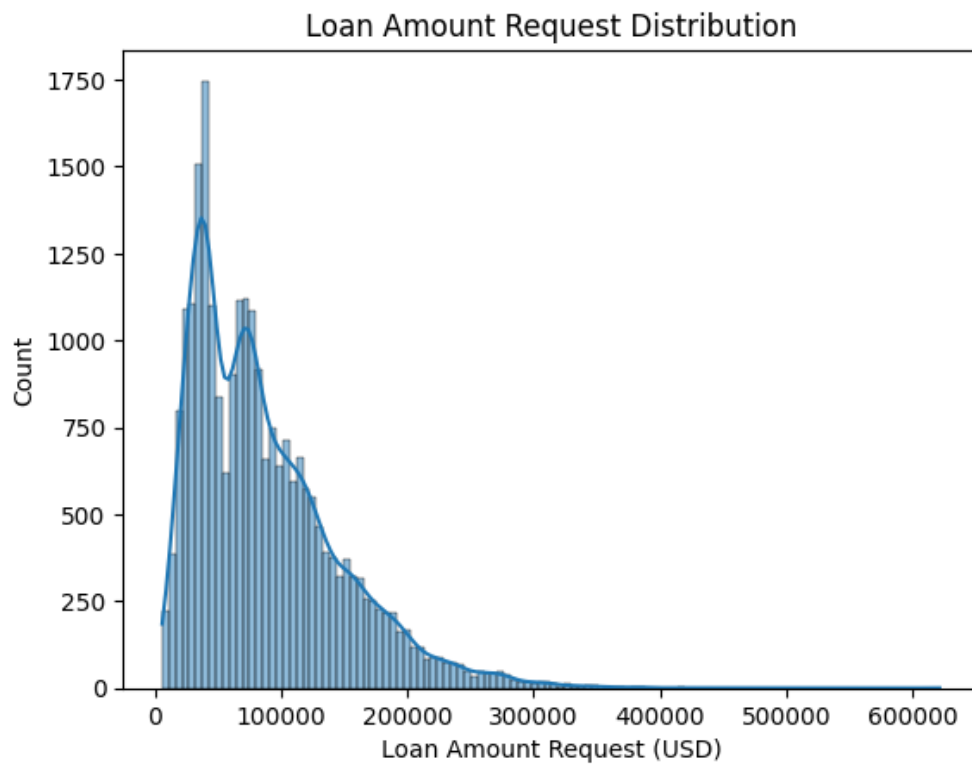


Figure 4: Loan Amount Request Distribution

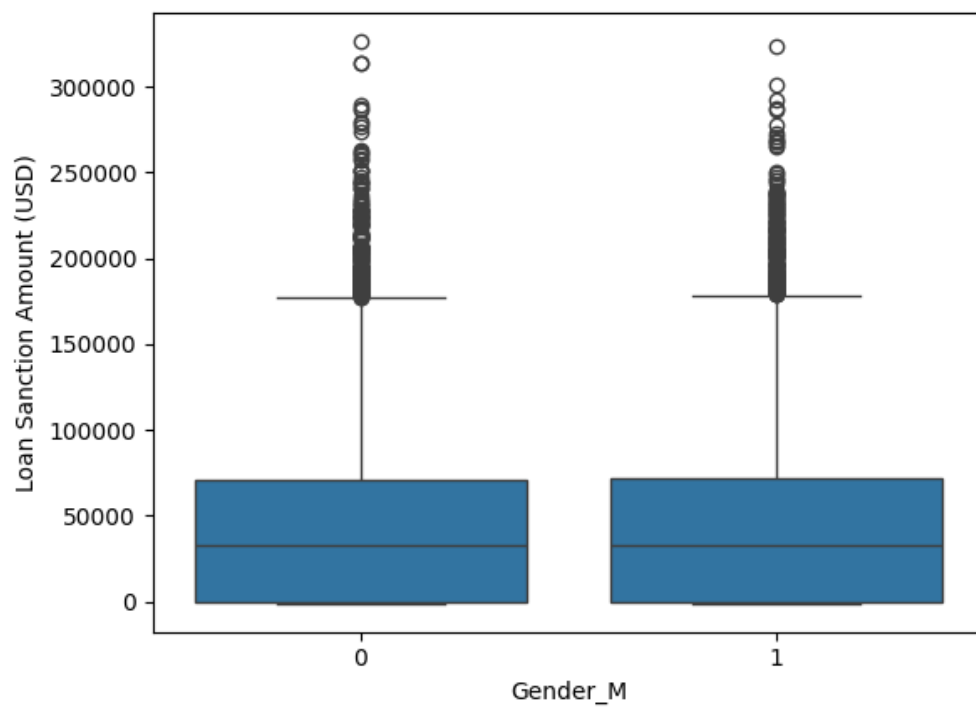


Figure 5: Gender boxplot

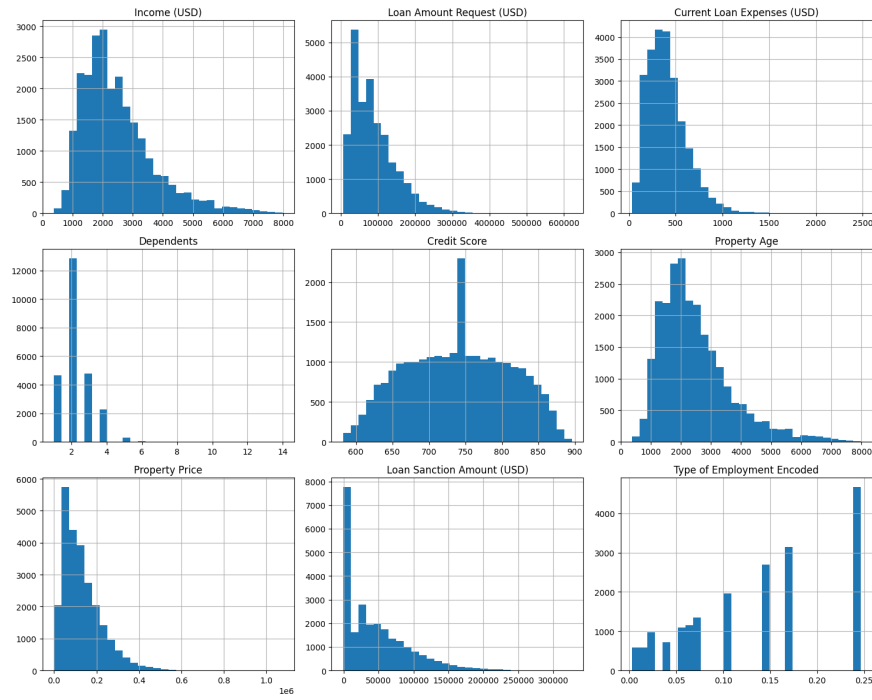


Figure 6: Histogram

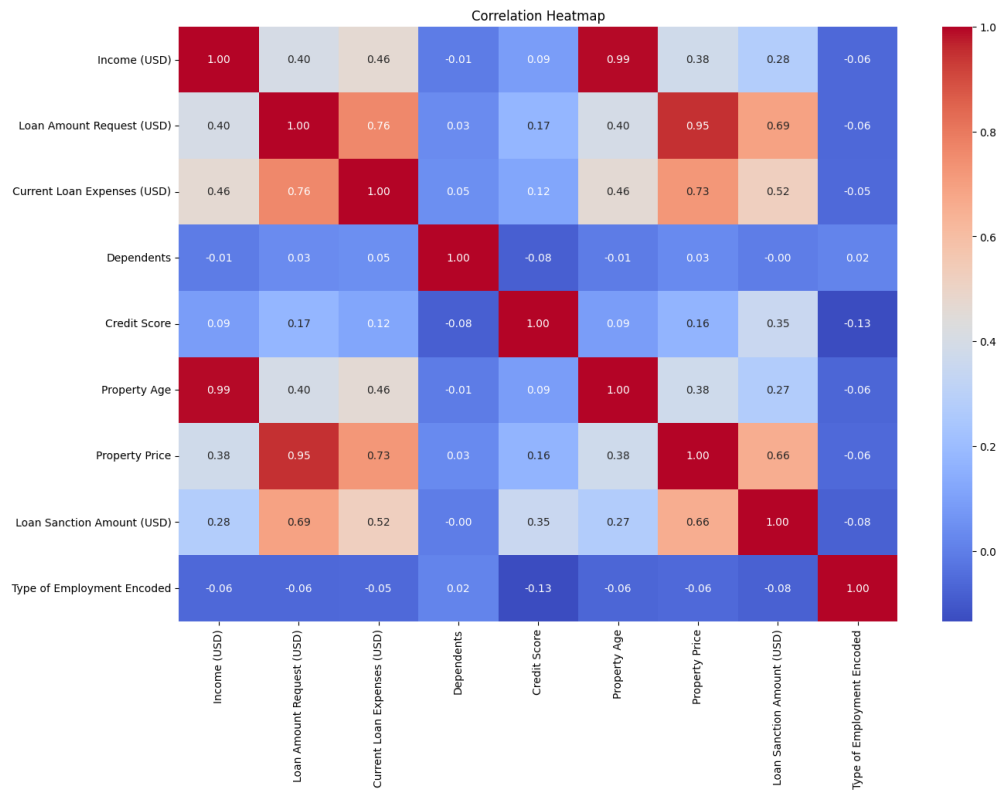


Figure 7: Correlation Heatmap

```

# 4. FEATURE ENGINEERING
scaler = StandardScaler()
num_cols = df.select_dtypes(include=['int64', 'float64']).columns # picking only numerical columns
df[num_cols] = scaler.fit_transform(df[num_cols])

# INTERACTION FEATURE
df['Affordability'] = np.log1p(df['Income (USD)'] / (df['Property Price'] + 1e-6))
df = df.dropna()

# 5. LINEAR REGRESSION - 5 FOLD CROSS VALIDATION
X = df.drop(columns=["Loan Sanction Amount (USD)"]) # everything except target
y = df["Loan Sanction Amount (USD)"] # target variable

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

for fold, (train_index, val_index) in enumerate(kf.split(X), start=1):
    X_train, X_val = X.iloc[train_index], X.iloc[val_index]
    y_train, y_val = y.iloc[train_index], y.iloc[val_index]

    model.fit(X_train, y_train)
    y_pred = model.predict(X_val)

    mae = mean_absolute_error(y_val, y_pred)
    mse = mean_squared_error(y_val, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_val, y_pred)

    print(f"Fold {fold} - MAE: {mae:.2f}, MSE: {mse:.2f}, RMSE: {rmse:.2f}, R2: {r2:.4f}")

print("\nAverage MAE:", avg_mae)
print("Average MSE:", avg_mse)
print("Average RMSE:", avg_rmse)
print("Average R2:", avg_r2)

```

```

Fold 1 - MAE: 0.43, MSE: 0.40, RMSE: 0.64, R2: 0.6323
Fold 2 - MAE: 0.43, MSE: 0.40, RMSE: 0.63, R2: 0.6287
Fold 3 - MAE: 0.46, MSE: 0.44, RMSE: 0.67, R2: 0.6208
Fold 4 - MAE: 0.43, MSE: 0.39, RMSE: 0.62, R2: 0.6581
Fold 5 - MAE: 0.44, MSE: 0.44, RMSE: 0.66, R2: 0.5926

Average MAE: 0.4408715112721679
Average MSE: 0.4145599607920346
Average RMSE: 0.6436476622798699
Average R2: 0.6264977748113167

```

Figure 8: Evaluation metrics

#### # 5. VISUALIZING RESULTS

```
plt.scatter(all_actuals, all_predictions, alpha=0.6, edgecolor='k')
plt.plot([all_actuals.min(), all_actuals.max()],
         [all_actuals.min(), all_actuals.max()])
plt.xlabel("Actual Loan Sanction Amount (USD)")
plt.ylabel("Predicted Loan Sanction Amount (USD)")
plt.title("Cross-Validation: Actual vs Predicted")
plt.grid(True)
plt.show()
```

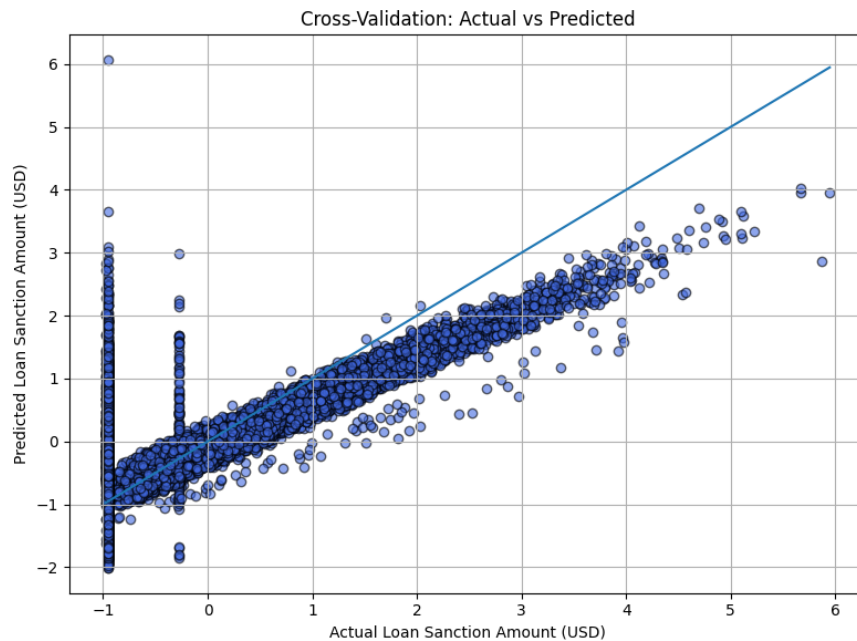


Figure 9: Actual vs Predicted

```
coeff_df = pd.DataFrame({
    'Feature': X_train.columns,
    'Coefficient': model.coef_
})

coeff_df = coeff_df.sort_values(by='Coefficient', key=abs, ascending=False)
sns.barplot(data=coeff_df, x='Coefficient', y='Feature')
plt.title("Feature Importance (Linear Coefficients)")
plt.grid(True)
plt.show()
```



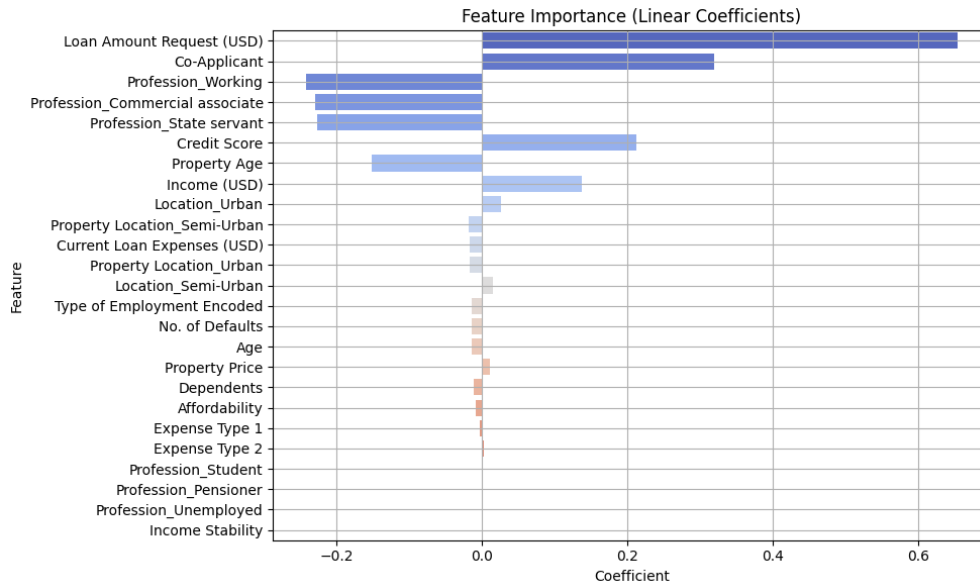


Figure 10: Feature Importance

```
residuals = all_actuals - all_predictions
plt.scatter(all_predictions, residuals, alpha=0.6)
plt.xlabel("Predicted Loan Sanction Amount (USD)")
plt.ylabel("Residuals")
plt.title("Cross-Validation: Residual Plot")
plt.grid(True)
plt.show()
```

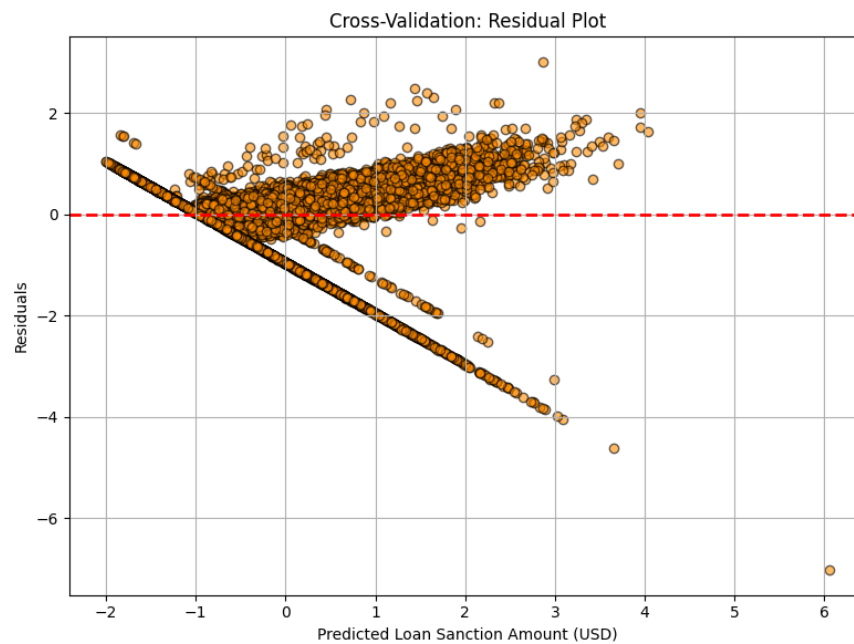


Figure 11: Residual Plot

## 6 Results Table:

Description	Student's Result
Dataset Size (after preprocessing)	24960 samples, 28 features
Train/Test Split Ratio	80:20
Feature(s) Used for Prediction	All numeric and encoded categorical features
Model Used	Linear Regression
Reference to CV Results Table	Table 1
Cross-Validation Used? (Yes/No)	Yes
If Yes, Number of Folds (K)	5
Mean Absolute Error (MAE) on Test Set	0.44087
Mean Squared Error (MSE) on Test Set	0.41455
Root Mean Squared Error (RMSE) on Test Set	0.64364
R2 Score on Test Set	0.62649
Most Influential Feature(s)	Loan Amount Request (USD), Credit Score, Co-Applicant, Income(USD)
Observations from Residual Plot	Shows distinct non-random patterns and heteroscedasticity, indicating potential underfitting. Linear model may be too simple for the underlying data
Interpretation of Predicted vs Actual Plot	While the model captures the general trend of loan sanction amounts, it struggles with accurate predictions for higher values
Any Overfitting or Underfitting Observed?	Underfitting observed
If Yes, Brief Justification (e.g., training vs test error, residual patterns)	Consistent underprediction and limited variance in predicted values across actual ranges. Overly simplified predictions

Table 1: Summary of Results for Loan Amount Prediction

### Cross-Validation Results Table:

Fold	MAE	MSE	RMSE	$R^2$ Score
Fold 1	0.43	0.40	0.64	0.6323
Fold 2	0.43	0.40	0.63	0.6287
Fold 3	0.46	0.44	0.67	0.6208
Fold 4	0.43	0.39	0.62	0.6581
Fold 5	0.44	0.44	0.66	0.5926
<b>Average</b>	0.4409	0.4146	0.6436	0.6265

Table 2: Cross-Validation Results (K = 5)

## 7 Best Practices:

- **Used 5-Fold Cross-Validation** to ensure robust model evaluation and minimize bias from data splits.
- **Reported multiple evaluation metrics** (MAE, MSE, RMSE, and  $R^2$ ) to provide a comprehensive assessment of model performance.
- **Observed consistent performance across folds**, indicating reasonable generalization and model stability.
- **No signs of overfitting**, as metrics remain relatively stable with no extreme deviations in any fold.
- **Rounded metric values** for readability in the table, while preserving precision in the back-end calculations.

## 8 Learning Outcomes

- Understood the implementation of 5-Fold Cross-Validation for evaluating model reliability.
- Gained experience interpreting regression metrics such as MAE, MSE, RMSE, and  $R^2$ .
- Identified the importance of consistency across folds to assess model stability.
- Learned to spot signs of overfitting or underfitting through performance trends.
- Developed a stronger grasp of model evaluation strategies used in real-world ML workflows.

### GitHub Repository:

<https://github.com/vidarshanaa15/ml-expt-2>