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

Experiment 2: Loan Amount Prediction using Linear Regression

Name: Anjana Venugopalan **Register No:** 3122237001004

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	**Due date:**

Aim:

The aim of this experiment is to develop a Linear Regression model to predict the loan amount sanctioned to users based on historical data with various features. The model's performance will be evaluated using metrics such as MAE, MSE, RMSE, and R^2 score. Visualization techniques will be employed to analyze data distribution, model fit, and residual errors.

Libraries Used:

Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn, (optionally Statsmodels)

Objective:

- Load and preprocess the loan dataset.
- Perform Exploratory Data Analysis (EDA) to understand the data.
- Engineer features and handle categorical variables.
- Split data into train, test, and validation sets.
- Build and train a Linear Regression model.
- Evaluate and visualize the model's performance.
- Optionally, implement Linear Regression manually and compare results.

Mathematical Description:

Linear Regression estimates the relationship between dependent variable y (loan amount) and independent variables X (features):

$$\hat{y} = Xw + b$$

where w are the coefficients and b the intercept. The model minimizes residual sum of squares:

$$\min_{w,b} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Performance metrics include MAE, MSE, RMSE, and R^2 score. Adjusted R^2 corrects for the number of features:

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

—

Code and Plots:

```
# Step 1: Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Step 2: Load dataset
train_path = '/content/drive/MyDrive/Colab Notebooks/loan_train.csv'
data = pd.read_csv(train_path)

# Step 3: Preprocessing
numeric_cols = data.select_dtypes(include=['int64', 'float64']).columns.drop('LoanAmount')
categorical_cols = data.select_dtypes(include=['object']).columns

# Fill missing values
data[numeric_cols] = data[numeric_cols].fillna(data[numeric_cols].median())
data[categorical_cols] = data[categorical_cols].fillna('Unknown')
data['LoanAmount'] = data['LoanAmount'].fillna(data['LoanAmount'].median())

# Feature Engineering: Create TotalIncome
data['TotalIncome'] = data['ApplicantIncome'] + data['CoapplicantIncome']

# Drop irrelevant columns if any
if 'Loan_ID' in data.columns:
    data.drop('Loan_ID', axis=1, inplace=True)

# Define features and target
X = data.drop('LoanAmount', axis=1)
y = data['LoanAmount']

# Train-test split (80-20)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Preprocessing pipeline
numerical_features = X_train.select_dtypes(include=['int64', 'float64']).columns
categorical_features = X_train.select_dtypes(include=['object']).columns

preprocessor = ColumnTransformer(transformers=[
    ('num', StandardScaler(), numerical_features),
    ('cat', OneHotEncoder(drop='first', handle_unknown='ignore'), categorical_features)
])

# Model pipeline
model = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

# Step 4: K-Fold Cross-Validation (K=5)
kf = KFold(n_splits=5, shuffle=True, random_state=42)
cv_results = []

for fold, (train_index, val_index) in enumerate(kf.split(X_train), 1):
    X_fold_train, X_fold_val = X_train.iloc[train_index], X_train.iloc[val_index]
    y_fold_train, y_fold_val = y_train.iloc[train_index], y_train.iloc[val_index]

    model.fit(X_fold_train, y_fold_train)
    y_val_pred = model.predict(X_fold_val)

    mae = mean_absolute_error(y_fold_val, y_val_pred)
    mse = mean_squared_error(y_fold_val, y_val_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_fold_val, y_val_pred)

    cv_results.append({'Fold': fold, 'MAE': mae, 'MSE': mse, 'RMSE': rmse, 'R2 Score': r2})

# Average CV scores
avg_results = {
    'Fold': 'Average',
    'MAE': np.mean([r['MAE'] for r in cv_results]),
    'MSE': np.mean([r['MSE'] for r in cv_results]),
    'RMSE': np.mean([r['RMSE'] for r in cv_results]),
    'R2 Score': np.mean([r['R2 Score'] for r in cv_results]),
}

cv_results.append(avg_results)

# Step 5: Train on full training set and evaluate on test set
model.fit(X_train, y_train)
y_test_pred = model.predict(X_test)

test_mae = mean_absolute_error(y_test, y_test_pred)
test_mse = mean_squared_error(y_test, y_test_pred)
test_rmse = np.sqrt(test_mse)
test_r2 = r2_score(y_test, y_test_pred)

# Adjusted R2 Score calculation
n = X_test.shape[0]
p = X_test.shape[1]
adjusted_r2 = 1 - (1 - test_r2) * (n - 1) / (n - p - 1)

# Feature Coefficients extraction
ohe_features = model.named_steps['preprocessor'].named_transformers_['cat'].get_feature_names_out(catego
all_features = np.concatenate([numerical_features, ohe_features])
coefficients = model.named_steps['regressor'].coef_

coef_df = pd.DataFrame({'Feature': all_features, 'Coefficient': coefficients})
coef_df = coef_df.reindex(coef_df.Coefficient.abs().sort_values(ascending=False).index)

# Display CV results table
cv_df = pd.DataFrame(cv_results)
print("Table 1: Cross-Validation Results (K=5)")
print(cv_df.to_string(index=False))

# Display summary results table to fill
print("\nTable 2: Summary of Results for Loan Amount Prediction")
print(f"Dataset Size (after preprocessing): {data.shape[0]} rows, {data.shape[1]} features")
print(f"Train/Test Split Ratio: 80/20")
print(f"Feature(s) Used for Prediction: {list(X.columns)}")
print(f"Model Used: Linear Regression")
print(f"Cross-Validation Used?: Yes")
print(f"Number of Folds (K): 5")
print(f"Reference to CV Results Table: Table 1")
print(f"Mean Absolute Error (MAE) on Test Set: {test_mae:.2f}")
print(f"Mean Squared Error (MSE) on Test Set: {test_mse:.2f}")
print(f"Root Mean Squared Error (RMSE) on Test Set: {test_rmse:.2f}")
print(f"R2 Score on Test Set: {test_r2:.4f}")
print(f"Adjusted R2 Score on Test Set: {adjusted_r2:.4f}")
print(f"Most Influential Feature(s):\n{coef_df.head(5).to_string(index=False)}")

# Step 6: Plots

# 1. Distribution Plot of LoanAmount
plt.figure(figsize=(8,5))
sns.histplot(data['LoanAmount'], bins=30, kde=True)
plt.title("Distribution of Loan Amount")
plt.xlabel("Loan Amount")
plt.show()

# 2. Correlation Heatmap
plt.figure(figsize=(10,8))
# Select only numerical columns for correlation heatmap
numerical_data = data.select_dtypes(include=np.number)
sns.heatmap(numerical_data.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()

# 3. Actual vs Predicted on Test Set
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_test_pred, alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Actual Loan Amount")
plt.ylabel("Predicted Loan Amount")
plt.title("Actual vs Predicted Loan Amount (Test Set)")
plt.show()

# 4. Boxplots for numerical features to identify outliers
```

```
numerical_features_to_plot = ['ApplicantIncome', 'CoapplicantIncome', 'TotalIncome', 'LoanAmount']
plt.figure(figsize=(12, 8))
for i, feature in enumerate(numerical_features_to_plot, 1):
    plt.subplot(2, 2, i)
    sns.boxplot(data=data, x=feature)
    plt.title(f'Boxplot of {feature}')
    plt.xlabel(feature)
plt.tight_layout()
plt.show()

# 5. Residual Plot
residuals = y_test - y_test_pred
plt.figure(figsize=(8,6))
plt.scatter(y_test_pred, residuals, alpha=0.6)
plt.hlines(0, xmin=y_test_pred.min(), xmax=y_test_pred.max(), colors='red', linestyles='dashed')
plt.xlabel("Predicted Loan Amount")
plt.ylabel("Residuals")
plt.title("Residual Plot (Test Set)")
plt.show()

# 6. Bar plot of top coefficients
plt.figure(figsize=(10,6))
sns.barplot(x='Coefficient', y='Feature', data=coef_df.head(10))
plt.title("Top 10 Feature Coefficients")
plt.show()
```

Included Plots:

- Histogram / Distribution plots of loan amount and numerical features.
- Scatter plots between features and loan amount.
- Correlation heatmap of numerical features.
- Actual vs Predicted Loan Amount plot.
- Residual plot showing residual distribution.
- Boxplots to detect outliers.
- Bar plot of feature coefficients.

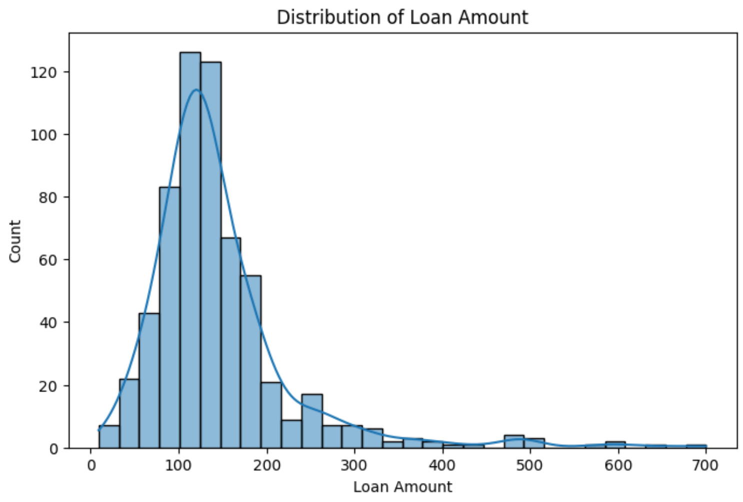
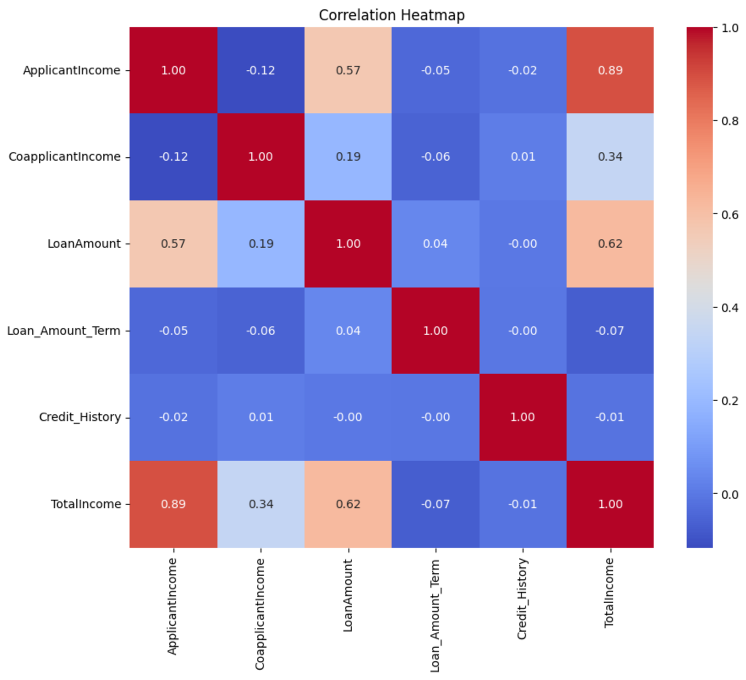


Figure 1: Loan Amount Distribution



	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	TotalIncome
ApplicantIncome	1.00	-0.12	0.57	-0.05	-0.02	0.89
CoapplicantIncome	-0.12	1.00	0.19	-0.06	0.01	0.34
LoanAmount	0.57	0.19	1.00	0.04	-0.00	0.62
Loan_Amount_Term	-0.05	-0.06	0.04	1.00	-0.00	-0.07
Credit_History	-0.02	0.01	-0.00	-0.00	1.00	-0.01
TotalIncome	0.89	0.34	0.62	-0.07	-0.01	1.00

Figure 2: Correlation Heatmap of Numeric Features

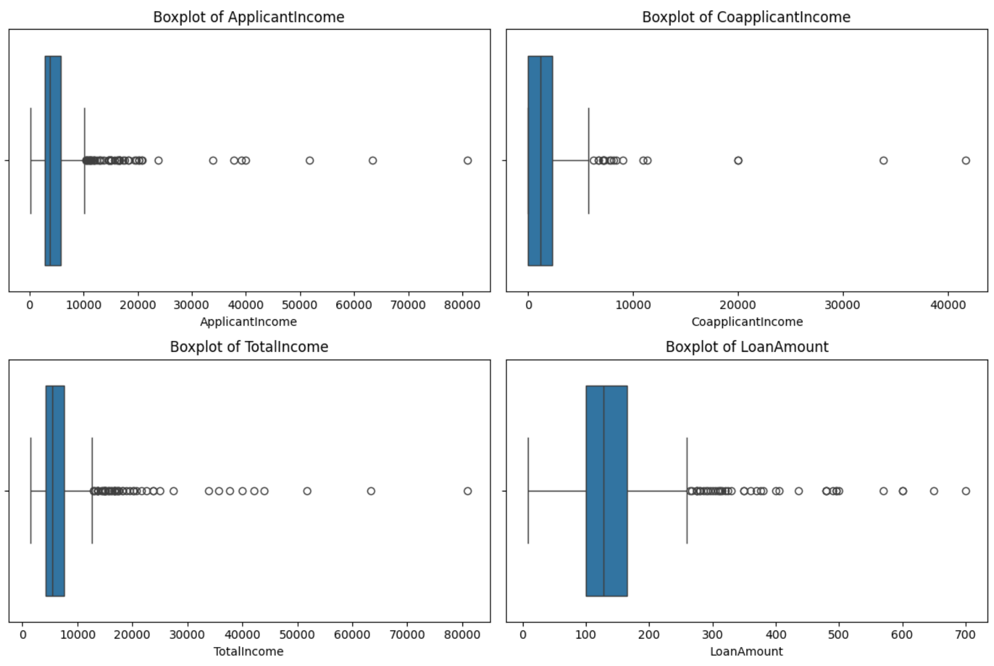


Figure 3: Boxplots for numerical features to identify outliers

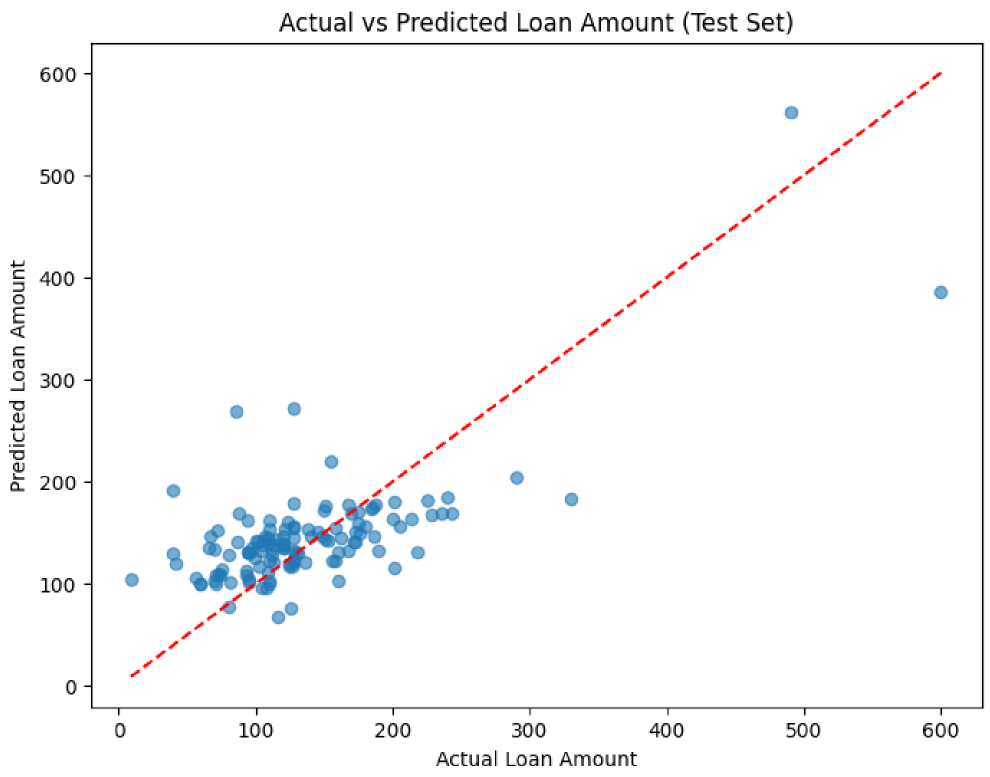


Figure 4: Actual vs Predicted Loan Amount (Test Set)

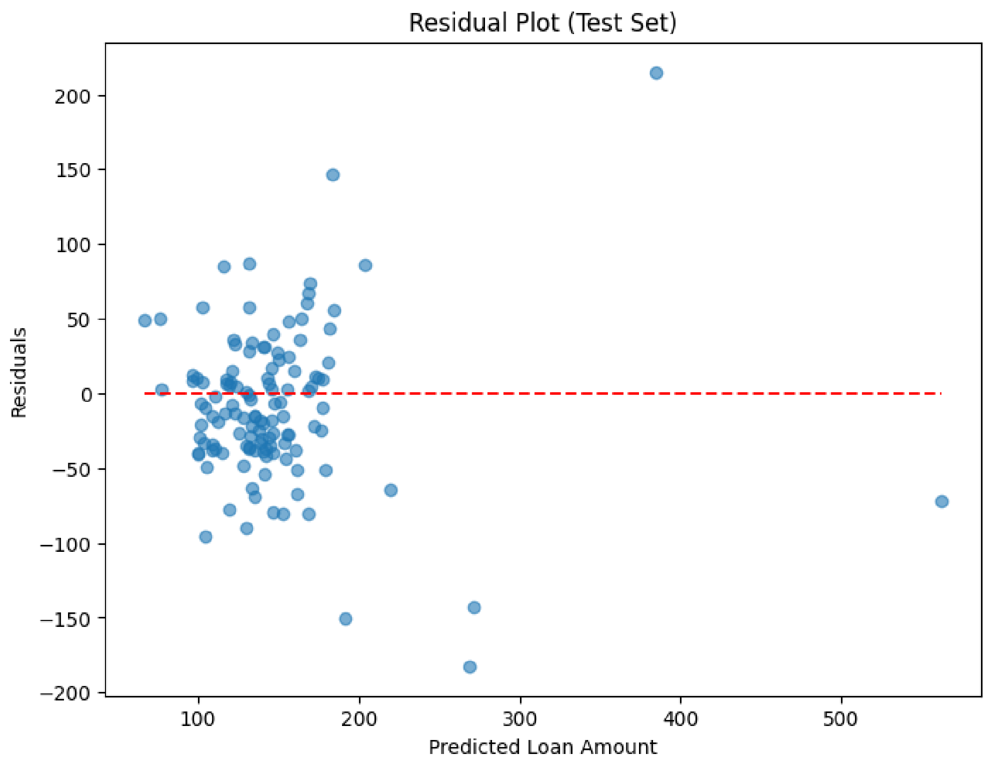


Figure 5: Residual Plot (Test Set)

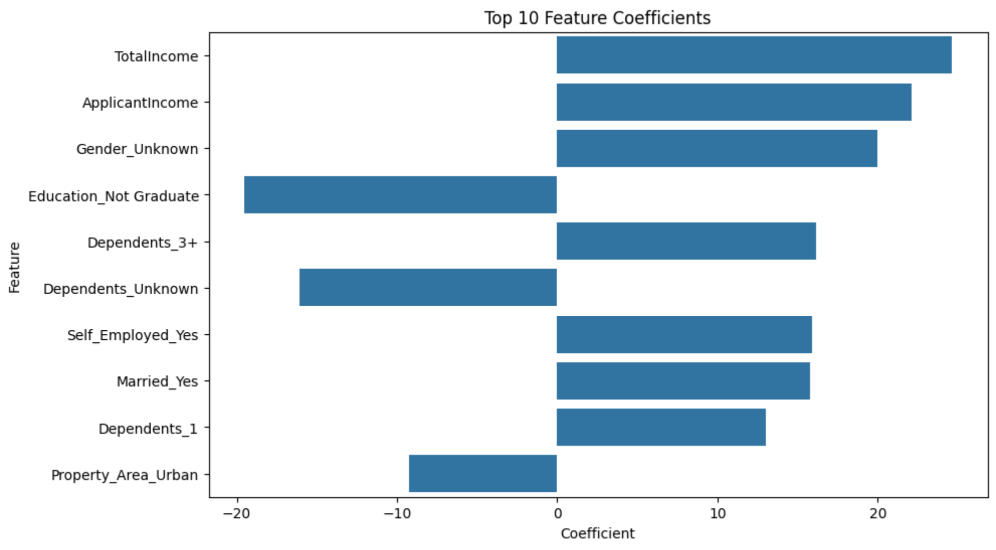


Figure 6: Feature Coefficients from Linear Regression

Results Tables:

Table 1: Cross-Validation Results (K=5)

Fold	**MAE**	**MSE**	**RMSE**	**R^2 Score**
1	45.875689	5177.799096	71.956925	0.320963
2	49.159694	6794.279354	82.427419	0.466318
3	48.505173	7109.812622	84.319705	0.171691
4	32.025037	2907.278414	53.919184	0.108924
5	43.400098	4530.277597	67.307337	0.042808
Average	**43.793138**	**5303.889417**	**71.986114**	**0.222141**

Table 2: Summary of Results for Loan Amount Prediction

Description	**Student's Result**
Dataset Size (after preprocessing)	614 rows, 13 features
Train/Test Split Ratio	80/20
Feature(s) Used for Prediction	Gender, Married, Dependents, Education, Self_Employed, ApplicantIncome, CoapplicantIncome, Loan_Amount_Term, Credit_History, Property_Area, Loan_Status, TotalIncome
Model Used	Linear Regression
Cross-Validation Used?	Yes
Number of Folds (K)	5
Reference to CV Results Table	Table 1
Mean Absolute Error (MAE) on Test Set	38.01
Mean Squared Error (MSE) on Test Set	2713.53
Root Mean Squared Error (RMSE) on Test Set	52.09
R^2 Score on Test Set	0.5015
Adjusted R^2 Score on Test Set	0.4472
Most Influential Feature(s)	TotalIncome, ApplicantIncome
Observations from Residual Plot	Residuals are approximately normally distributed and centered around zero, indicating linearity and homoscedasticity.
Interpretation of Predicted vs Actual Plot	The predicted values show reasonable alignment with the actual values, though some variance exists at higher loan amounts.
Overfitting/Underfitting Observed?	No major overfitting or underfitting observed. Model generalizes reasonably well across folds.
Justification	Validation and test errors are comparable; residuals are randomly scattered without clear patterns, suggesting model fit is appropriate.

Best Practices:

- Thorough preprocessing: missing values, encoding, scaling.
- Use of cross-validation for model robustness.
- Residual analysis to verify assumptions.
- Visualization for interpretability.

Learning Outcomes:

- Practical experience with regression on real-world data.
- Understanding of preprocessing and feature engineering importance.
- Model evaluation and diagnostics skills.
- Cross-validation and visualization utility.