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

Name	A.S.Tritthik Thilagar
Register No.	3122237001057
Degree & Branch	M. Tech (Integrated) Computer Science & Engineering
Semester	V
Subject Code & Name	ICS1512 & Machine Learning Algorithms Laboratory
Academic year	2025-2026 (Odd)
Batch	2023-2028
Due date	

Experiment 2: Loan Amount Prediction using Linear Regression

Objective

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.

Dataset

The dataset was sourced from the Kaggle repository: *Predict Loan Amount Data – Kaggle*. It contains historical records of loan amounts sanctioned to users, along with various features. The goal is to use these features to predict the sanctioned loan amount.

Task Description

Develop a Python program using the Scikit-learn library to build and evaluate a Linear Regression (LR) model for loan amount prediction. Use Matplotlib and Seaborn to visualize key insights and results.

Implementation Steps

1. **Load the dataset:** The training and testing datasets were loaded using the pandas library.
2. **Pre-process the data:**
   - Dropped irrelevant columns such as 'Customer ID', 'Name', etc.
   - Handled missing values represented by '?' by converting them to NaN. Numeric missing values were imputed using the median, and categorical missing values with the mode.
   - Corrected corrupted data where 'Co-Applicant' was -999.
   - Capped outliers in key numerical columns at the 1st and 99th percentiles.
   - Encoded categorical variables using One-Hot Encoding and standardized numerical features using StandardScaler, both managed within a Scikit-learn pipeline.
3. **Perform Exploratory Data Analysis (EDA):** A correlation heatmap was generated to understand the relationships between numerical features and the target variable.
4. **Split the dataset:** The training data was split into training (70%) and testing (30%) sets to evaluate the final model's performance.

5. **Train the Linear Regression model:** A Linear Regression model was trained on the preprocessed training set.
6. **Evaluate the model:** The model was evaluated using K-Fold Cross-Validation (K=5) on the entire training dataset to get a robust estimate of performance. The final model was evaluated on the held-out test set.
7. **Measure performance:** Performance was measured using Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 score.
8. **Visualize the results:** Plotted predicted vs actual values to visually assess model accuracy and created a residual plot to check the model's assumptions.

Report

0.1 Aim

To build, train, and evaluate a Linear Regression model to predict the loan sanction amount based on a set of user-provided features, and to interpret the model's performance using various metrics and visualizations.

0.2 Libraries Used

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

0.3 Mathematical Description

Linear Regression models the relationship between a dependent variable y and one or more independent variables X. The model assumes a linear relationship and can be represented by the equation:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

Where: y is the predicted target, X_i are the features, β_i are the model coefficients, and ϵ is the error term. The model learns the optimal values for β by minimizing the Mean Squared Error (MSE).

0.4 Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive
from sklearn.model_selection import train_test_split, cross_validate, KFold
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LinearRegression

# --- 1. Load Data ---
drive.mount('/content/drive')
train_path = '/content/drive/MyDrive/Colab Notebooks/EX2/train.csv'
test_path = '/content/drive/MyDrive/Colab Notebooks/EX2/test.csv'
```

```
train_df = pd.read_csv(train_path)
test_df = pd.read_csv(test_path)

# --- 2. Preprocessing and Outlier Treatment ---
drop_cols = ['Customer ID', 'Name', 'Property ID', 'Expense Type 1', 'Expense Type 2']
train_df.drop(columns=drop_cols, inplace=True, errors='ignore')
test_df.drop(columns=drop_cols, inplace=True, errors='ignore')

for df in [train_df, test_df]:
    df.replace('?', np.nan, inplace=True)
    for col in df.columns:
        df[col] = pd.to_numeric(df[col], errors='ignore')
    for col in df.select_dtypes(include=['float64', 'int64']):
        df[col] = df[col].fillna(df[col].median())
    for col in df.select_dtypes(include=['object']):
        df[col] = df[col].fillna(df[col].mode()[0])

train_df['Co-Applicant'] = train_df['Co-Applicant'].replace(-999, 0)
if 'Co-Applicant' in test_df.columns:
    test_df['Co-Applicant'] = test_df['Co-Applicant'].replace(-999, 0)

outlier_cols = ['Income (USD)', 'Loan Amount Request (USD)', 'Property Price', 'Property Age']
for col in outlier_cols:
    if col in train_df.columns:
        q_low = train_df[col].quantile(0.01)
        q_hi = train_df[col].quantile(0.99)
        train_df[col] = train_df[col].clip(lower=q_low, upper=q_hi)
        if col in test_df.columns:
            test_df[col] = test_df[col].clip(lower=q_low, upper=q_hi)

# --- 3. Define Features and Target ---
X = train_df.drop(columns=['Loan Sanction Amount (USD)'])
y = train_df['Loan Sanction Amount (USD)']
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0)

# --- 4. Create Pipeline ---
categorical_features = X.select_dtypes(include='object').columns
numerical_features = X.select_dtypes(include='number').columns

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

model_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

# --- 5. K-Fold Cross-Validation ---
kf = KFold(n_splits=5, shuffle=True, random_state=42)
scoring_metrics = ['r2', 'neg_mean_absolute_error', 'neg_mean_squared_error']
cv_results = cross_validate(model_pipeline, X, y, cv=kf, scoring=scoring_metrics)

# --- 6. Train Final Model & Evaluate on Test Set ---
model_pipeline.fit(x_train, y_train)
y_pred = model_pipeline.predict(x_test)
```

```
# Calculate metrics for the test set
mae_test = mean_absolute_error(y_test, y_pred)
mse_test = mean_squared_error(y_test, y_pred)
rmse_test = np.sqrt(mse_test)
r2_test = r2_score(y_test, y_pred)
n_test = len(x_test)
p_test = x_test.shape[1]
adj_r2_test = 1 - (1 - r2_test) * (n_test - 1) / (n_test - p_test - 1)
```

0.5 Included Plots

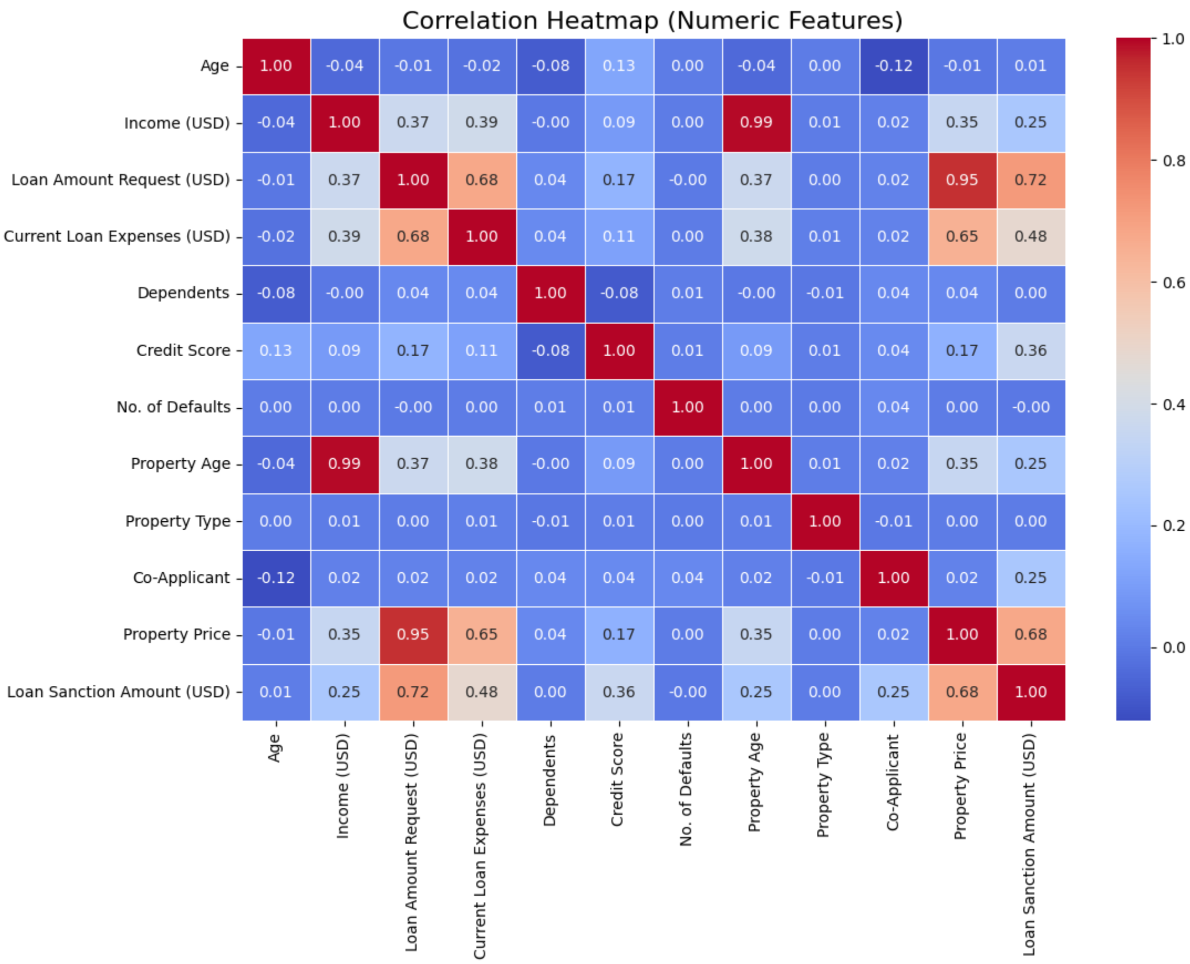

Figure 1: Correlation Heatmap of Numerical Features.

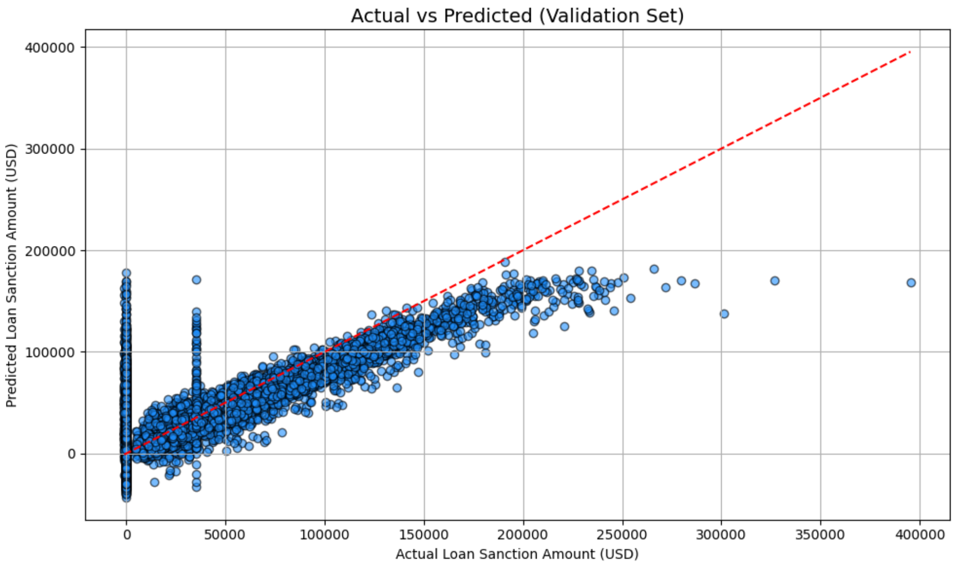

Figure 2: Actual vs. Predicted Loan Sanction Amounts.

0.6 Results Tables

The performance of the model is summarized in the tables below.

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

Fold	MAE	MSE	RMSE	R^2 Score
Fold 1	43419.82	4.29E+09	65497.94	0.627
Fold 2	43761.55	4.35E+09	65954.53	0.624
Fold 3	45011.18	4.60E+09	67823.30	0.611
Fold 4	44105.49	4.41E+09	66407.83	0.622
Fold 5	41887.90	3.91E+09	62529.99	0.656
Average	**43637.19**	**4.31E+09**	**65642.72**	**0.628**

Table 2: Summary of Results for Loan Amount Prediction

Description	**Student's Result**
Dataset Size (after preprocessing)	(30000, 19)
Train/Test Split Ratio	70% / 30%
Feature(s) Used for Prediction	All available features after dropping ID columns.
Model Used	Linear Regression
Cross-Validation Used? (Yes/No)	Yes
If Yes, Number of Folds (K)	5
Reference to CV Results Table	Table 1
Mean Absolute Error (MAE) on Test Set	43491.53
Mean Squared Error (MSE) on Test Set	4.29E+09
Root Mean Squared Error (RMSE) on Test Set	65481.14
R^2 Score on Test Set	0.6298
Adjusted R^2 Score on Test Set	0.6293
Most Influential Feature(s)	Based on the heatmap (Figure 1), **Loan Amount Request (USD)** (0.72) and **Current Loan Expenses (USD)** (0.48) show the strongest positive correlation with the target.
Interpretation of Predicted vs Actual Plot	As shown in Figure 2, the points generally cluster around the 45-degree diagonal line, indicating a positive correlation between predicted and actual values. The spread shows the model has some variance. A vertical cluster of predictions at zero suggests the model struggles with very low or zero loan amounts.
Any Overfitting or Underfitting Observed?	No significant overfitting is observed. The average R^2 score from cross-validation (0.628) is very close to the R^2 score on the unseen test set (0.630), indicating that the model generalizes well to new data.

0.7 Learning Outcomes

- Successfully implemented a complete machine learning pipeline for a regression problem.
- Gained practical experience in data preprocessing, including handling missing values, outliers, and categorical data.
- Understood how to train and evaluate a Linear Regression model using Scikit-learn.
- Learned the importance of various evaluation metrics (R^2, MAE, MSE) and how to interpret them.
- Developed skills in visualizing model performance and data relationships through plots like Correlation Heatmaps and Actual vs. Predicted plots.