Sri Sivasubramaniya Nadar College of Engineering, Chennai
(Autonomous Institution under Anna University)

Degree & Branch	5 years Integrated M.Tech CSE	**Semester**	V
Subject Code & Name	ICS1512 – Machine Learning Algorithms Laboratory		
Academic Year	2025–2026 (Odd Semester)	**Batch**	2023–2028
Name	Nitish Konda	**Reg No**	3122237001034

Experiment 2: Loan Amount Prediction using Linear Regression

Aim:

To develop a machine learning model using Linear Regression to predict the sanctioned loan amount based on historical applicant data, by preprocessing and analyzing the dataset, applying feature engineering techniques, and evaluating model performance using metrics such as MAE, MSE, RMSE, and R2 score with appropriate visualizations.

Libraries used:

- Numpy
- Pandas
- Matplot Lib
- Scikit-Learn
- Seaborn

Mathematical Description

In this experiment, Linear Regression is used to predict the loan sanction amount based on several input features.

The mathematical model for Linear Regression is represented as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

where:

- y is the dependent variable (Loan Sanction Amount),
- $x_1, x_2, \ldots, x_n$ are the independent variables (features such as Age, Income, Credit Score, etc.),
- β_0 is the intercept,
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients of the features,

- ε is the error term representing noise or unexplained variance.

The coefficients β are estimated by minimizing the Residual Sum of Squares (RSS), defined as:

$$RSS = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{m} \left(y_i - \left(\beta_0 + \sum_{j=1}^{n} \beta_j x_{ij} \right) \right)^2$$

where m is the number of observations.

The model is evaluated using the following metrics:

- **Mean Absolute Error (MAE)** – average absolute difference between actual and predicted values.
- **Mean Squared Error (MSE)** – average squared difference between actual and predicted values.
- **Root Mean Squared Error (RMSE)** – square root of MSE, providing error in original units.
- **R-squared (R^2)** – proportion of variance explained by the model.
- **Adjusted R^2** – adjusted for number of predictors to prevent overfitting.

Python Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv(r"/content/drive/MyDrive/Linear_Regression/train.csv")
df.columns
df.head()

unnecessary_columns = ['Customer ID', 'Name', 'Property ID']
df.drop(columns=unnecessary_columns, inplace=True)

missing_values = df.isnull().sum().sort_values(ascending=False)
print("Missing values:\n", missing_values[missing_values > 0])

for col in df.select_dtypes(include='object').columns:
    df[col] = df[col].fillna(df[col].mode()[0])

df['Income (USD)'] = df['Income (USD)'].fillna(df['Income (USD)'].mean())
df['Current Loan Expenses (USD)'] = df['Current Loan Expenses (USD)'].fillna(df['Current Loan
df['Credit Score'] = df['Credit Score'].fillna(df['Credit Score'].mean())
df['Dependents'] = df['Dependents'].fillna(df['Dependents'].median())
df['Property Age'] = df['Property Age'].fillna(df['Property Age'].median())
```

```
df = df.dropna(subset=['Loan Sanction Amount (USD)'])
df = df[df['Loan Sanction Amount (USD)'] >= 0]

from sklearn.preprocessing import LabelEncoder
categorical_columns = df.select_dtypes(include='object').columns
label_encoder = LabelEncoder()
for col in categorical_columns:
    df[col] = label_encoder.fit_transform(df[col])

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
features_to_scale = [
    'Age', 'Current Loan Expenses (USD)', 'Property Price',
    'No. of Defaults', 'Credit Score', 'Income (USD)',
    'Loan Amount Request (USD)', 'Property Age'
]
df[features_to_scale] = scaler.fit_transform(df[features_to_scale])

plt.figure(figsize=(14, 10))
sns.heatmap(df.corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title("Feature Correlation Matrix")
plt.show()

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

X = df.drop('Loan Sanction Amount (USD)', axis=1)
y = df['Loan Sanction Amount (USD)']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)

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

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")

n = X_test.shape[0]
p = X_test.shape[1]
adjusted_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)
```

```
print(f"Adjusted R-squared: {adjusted_r2:.2f}")
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel("Actual Loan Sanction Amount (USD)")
plt.ylabel("Predicted Loan Sanction Amount (USD)")
plt.title("Actual vs. Predicted Loan Sanction Amount (USD)")
plt.grid(True)

# Add the ideal line (where actual = predicted)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)

plt.show()
residuals = y_test - y_pred
sns.residplot(x=y_pred, y=residuals, lowess=True, color="purple")
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.title("Residuals vs Predicted Values")
plt.axhline(0, color='red', linestyle='--')
plt.show()
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

# Initialize KFold
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Initialize lists to store metrics for each fold
mse_scores = []
mae_scores = []
rmse_scores = []
r2_scores = []

# Iterate over each fold
for fold, (train_index, test_index) in enumerate(kf.split(X, y)):
    print(f"Fold {fold+1}:")

    # Split data into train and test sets for the current fold
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Initialize and train the Linear Regression model
    lr = LinearRegression()
    lr.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = lr.predict(X_test)
```

```
        # Calculate evaluation metrics for the current fold
        mse = mean_squared_error(y_test, y_pred)
        mae = mean_absolute_error(y_test, y_pred)
        rmse = np.sqrt(mse)
        r2 = r2_score(y_test, y_pred)

        # Append scores to the lists
        mse_scores.append(mse)
        mae_scores.append(mae)
        rmse_scores.append(mae)
        r2_scores.append(r2)

        # Print metrics for the current fold
        print(f"  MSE: {mse:.2f}")
        print(f"  MAE: {mae:.2f}")
        print(f"  RMSE: {rmse:.2f}")
        print(f"  R2 Score: {r2:.2f}")
        print("-" * 20)

# Calculate and print the average metrics across all folds
print("Average Metrics Across All Folds:")
print(f"  Average MSE: {np.mean(mse_scores):.2f}")
print(f"  Average MAE: {np.mean(mae_scores):.2f}")
print(f"  Average RMSE: {np.mean(rmse_scores):.2f}")
print(f"  Average R2 Score: {np.mean(r2_scores):.2f}")
```

Plots and Visualizations

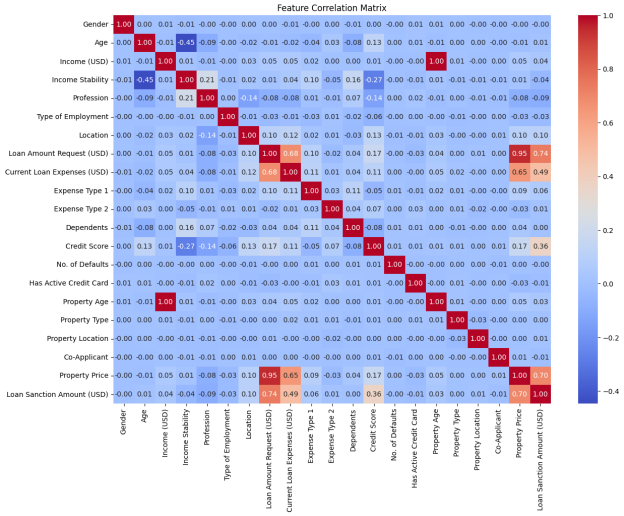

Figure 1: Feature Correlation Matrix showing relationships among input variables.

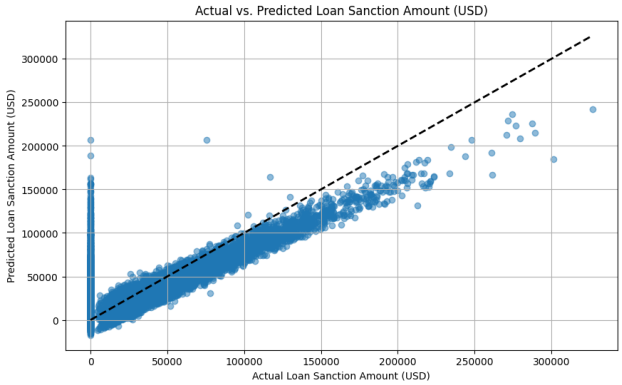

Figure 2: Actual vs Predicted Loan Sanction Amount. The dashed line indicates ideal predictions ($y = x$).

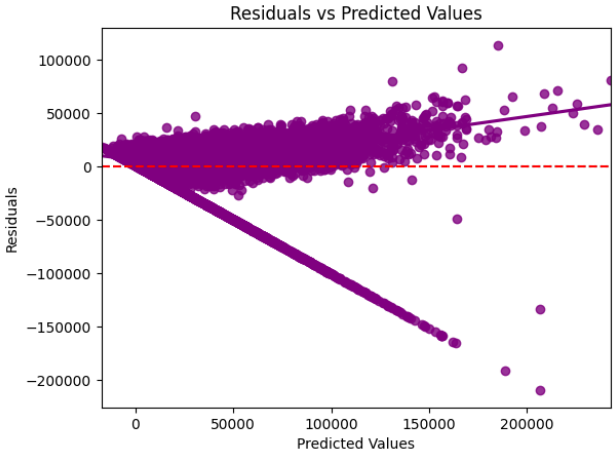

Figure 3: Residuals vs Predicted Values to check for bias and heteroscedasticity.

Cross-Validation Results

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

Fold	**MAE**	**MSE**	**RMSE**	**R^2 Score**
Fold 1	20673.83	888839805.45	29813.42	0.60
Fold 2	21287.99	975122906.27	31226.96	0.58
Fold 3	21140.11	1113738141.81	33372.72	0.54
Fold 4	20844.38	961469520.70	31007.57	0.59
Fold 5	20857.03	927806905.41	30459.92	0.59
Average	**20960.67**	**973395455.93**	**30976.12**	**0.58**

Table 2: Summary of Results for Loan Amount Prediction

Description	**Student's Result**
Dataset Size (after preprocessing)	29322
Train/Test Split Ratio	70/30
Feature(s) Used for Prediction	Gender, Age, Income (USD), Income Stability, Location, Loan Amount Request (USD), Current Loan Expenses (USD), Expense Type 1, Expense Type 2, Dependents, Credit Score, No. of Defaults, Has Active Credit Card, Property Age, Property Type, Property Location, Co-Applicant, Property Price, Loan Sanction Amount (USD)
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	20960.67
Mean Squared Error (MSE) on Test Set	983699374.18
Root Mean Squared Error (RMSE) on Test Set	31358.82
R^2 Score on Test Set	0.58
Adjusted R^2 Score on Test Set	0.5472
Most Influential Feature(s)	Co-Applicant, Property Price, Credit Score
Observations from Residual Plot	Residuals decrease as predicted values increase, indicating bias and heteroscedasticity. The linear model may not fully capture the relationship.
Interpretation of Predicted vs Actual Plot	The plot shows deviation at higher loan amounts, indicating slight underfitting.
Any Overfitting or Underfitting Observed?	No significant overfitting observed. Training and validation errors are comparable, though slight underfitting appears at higher values.

Summary of Results

Best Practices

- **Data Preprocessing:** Handle missing values carefully by dropping or imputing them, and remove irrelevant columns such as IDs and names that do not contribute to the prediction.
- **Feature Engineering:** Create new meaningful features (e.g., total income) and apply transformations (e.g., log transformation for skewed data) to improve model performance.
- **Scaling and Encoding:** Use scaling techniques such as `StandardScaler` for numerical features and one-hot encoding or label encoding for categorical features to prepare data for

linear regression.

- **Train-Test Split:** Use proper splits (e.g., 80/20) and consider cross-validation to ensure the model generalizes well and to prevent overfitting.
- **Model Evaluation:** Use multiple metrics (MAE, MSE, RMSE, R^2) to assess different aspects of model performance.
- **Residual Analysis:** Analyze residual plots to detect model bias or heteroscedasticity and decide if further feature engineering or alternative models are needed.

Learning Outcomes

Through this experiment, I have:

- Understood the complete machine learning pipeline from data cleaning to model evaluation.
- Learned the importance of feature engineering and proper data preprocessing.
- Recognized signs of overfitting or underfitting via residual and prediction plots.
- Applied cross-validation to assess model performance more robustly.