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	**Due date: 30.07.2025**

Experiment 2: Loan Amount Prediction using Linear Regression

1 Aim

To predict the sanctioned loan amount using Linear Regression based on the historical loan data provided.

2 Libraries Used

- `pandas`
- `numpy`
- `matplotlib`
- `seaborn`
- `sklearn`

3 Objective

To apply Linear Regression for estimating loan amounts, perform EDA, and visualize the model performance using appropriate metrics and plots.

4 Mathematical Description

1. Linear Regression

Linear Regression models the relationship between a dependent variable y and independent variables $x_1, x_2, \ldots, x_n$ by fitting a linear equation of the form:

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

Where:

- $\hat{y}$ is the predicted output
- w_0 is the intercept (bias term)
- $w_1, w_2, \ldots, w_n$ are the coefficients (weights) for the respective features

The coefficients are chosen to minimize the **Mean Squared Error (MSE)** between actual values y_i and predicted values $\hat{y}_i$:

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

—

2. Ridge Regression (L2 Regularization)

Ridge Regression modifies the linear regression cost function by adding a penalty on the square of the coefficients:

$$\text{Cost} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{n} w_j^2$$

Where:

- λ is the regularization parameter that controls the penalty strength
- w_j are the model coefficients (excluding the intercept)

This helps in reducing overfitting and multicollinearity by shrinking the coefficient values.

—

3. Lasso Regression (L1 Regularization)

Lasso Regression adds a penalty on the absolute value of the coefficients:

$$\text{Cost} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{n} |w_j|$$

Unlike Ridge, Lasso can shrink some coefficients exactly to zero, effectively performing feature selection.

—

4. Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

MSE penalizes larger errors more significantly than smaller ones due to squaring. It is commonly used to evaluate regression models.

—

5. Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|$$

MAE gives the average absolute difference between actual and predicted values. It is less sensitive to outliers than MSE.

—

6. Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$$

RMSE is the square root of MSE and has the same unit as the target variable. It is useful when interpreting the error in the original scale.

—

7. Coefficient of Determination (R^2 Score)

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{m}(y_i - \bar{y})^2}$$

Where:

- $\bar{y}$ is the mean of actual values
- The numerator is the residual sum of squares
- The denominator is the total sum of squares

R^2 represents the proportion of variance in the dependent variable that is predictable from the independent variables.

—

8. Adjusted R^2 Score

$$\text{Adjusted } R^2 = 1 - \left(1 - R^2\right) \cdot \frac{m-1}{m-n-1}$$

Where:

- m is the number of observations
- n is the number of predictors (features)

Adjusted R^2 accounts for the number of predictors and helps prevent overestimating the goodness of fit when adding more features.

5 Code with Plot

```
from google.colab import drive
drive.mount('/content/drive')

# Import Libraries
import kagglehub
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import Ridge, Lasso
from sklearn.preprocessing import LabelEncoder

# Download dataset
path = kagglehub.dataset_download("phileinsophos/predict-loan-amount-data")
print("Path to dataset files:", path)

train = pd.read_csv(path + "/train.csv")
test = pd.read_csv(path + "/test.csv")

# Drop Unnecessary Columns
unnecessary_columns = [’Customer ID’, ’Name’, ’Property ID’]
train.drop(columns=unnecessary_columns, inplace=True)

# Check for missing values
missing_values = train.isnull().sum().sort_values(ascending=False)
print("Missing values:\n", missing_values[missing_values > 0])

# Drop rows with too many missing values or fill them appropriately
train.fillna({
    ’Credit Score’: train[’Credit Score’].median(),
    ’Property Age’: train[’Property Age’].median(),
    ’Income (USD)’: train[’Income (USD)’].median(),
    ’Loan Amount Request (USD)’: train[’Loan Amount Request (USD)’].median(),
    ’Current Loan Expenses (USD)’: 0,
    ’No. of Defaults’: 0,
    ’Dependents’: 0,
}, inplace=True)

train = train[˜train[’Loan Sanction Amount (USD)’].isna()]
train = train[train[’Loan Sanction Amount (USD)’] > 0]

# For categorical: fill with mode
for col in train.select_dtypes(include=’object’).columns:
    train[col] = train[col].fillna(train[col].mode()[0])

categorical_cols = train.select_dtypes(include=’object’).columns

label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    train[col] = le.fit_transform(train[col])
    label_encoders[col] = le
```

```
features_to_scale = [
    ’Age’, ’Current Loan Expenses (USD)’, ’Property Price’,
    ’No. of Defaults’, ’Credit Score’, ’Income (USD)’,
    ’Loan Amount Request (USD)’, ’Property Age’
]

scaler = StandardScaler()
train[features_to_scale] = scaler.fit_transform(train[features_to_scale])

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

# Target variable distribution
sns.histplot(train[’Loan Sanction Amount (USD)’], kde=True)
plt.title("Distribution of Loan Sanction Amount")
plt.show()

X = train.drop(columns=[’Loan Sanction Amount (USD)’])
y = train[’Loan Sanction Amount (USD)’]

# Bin target variable for stratified sampling (5 bins for 5 folds)
y_binned = pd.qcut(y, q=5, labels=False)

skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

mae_scores = []
mse_scores = []
rmse_scores = []
r2_scores = []
adj_r2_scores = []

fold = 1
results = []

def adjusted_r2(r2, n, k):
    """Calculates adjusted R² score."""
    return 1 - (1 - r2) * (n - 1) / (n - k - 1)

saved = False

# Perform Stratified K-Fold Cross-Validation
for train_idx, val_idx in skf.split(X, y_binned):
    X_train_fold, X_val_fold = X.iloc[train_idx], X.iloc[val_idx]
    y_train_fold, y_val_fold = y.iloc[train_idx], y.iloc[val_idx]
    if not saved:
        X_train = X_train_fold
        y_train = y_train_fold
        X_val = X_val_fold
        y_val = y_val_fold
        saved = True

    model = LinearRegression()
    model.fit(X_train_fold, y_train_fold)
    y_pred = model.predict(X_val_fold)

    # Calculate performance metrics
    mae = mean_absolute_error(y_val_fold, y_pred)
    mse = mean_squared_error(y_val_fold, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_val_fold, y_pred)

    # Calculate Adjusted R²
    n = X_val_fold.shape[0]  # Number of samples in validation set
    k = X_val_fold.shape[1]  # Number of features
    adj_r2 = adjusted_r2(r2, n, k)

    # Store the results for each fold
    mae_scores.append(mae)
    mse_scores.append(mse)
    rmse_scores.append(rmse)
    r2_scores.append(r2)
    adj_r2_scores.append(adj_r2)

    results.append([f"Fold {fold}", mae, mse, rmse, r2, adj_r2])
    fold += 1

# Add average row
results.append(["Average", np.mean(mae_scores), np.mean(mse_scores),
                np.mean(rmse_scores), np.mean(r2_scores), np.mean(adj_r2_scores)])

# Convert to DataFrame for display
cv_results_df = pd.DataFrame(results, columns=["Fold", "MAE", "MSE", "RMSE", "R² Score", "Adjus
print(cv_results_df)

# 1. Histogram / Distribution Plot
sns.histplot(train['Loan Sanction Amount (USD)'], kde=True)
plt.title("Distribution of Loan Sanction Amount")
plt.show()

# 2. Scatter Plots
fig, axs = plt.subplots(1, 2, figsize=(14, 6))
sns.scatterplot(data=train, x='Income (USD)', y='Loan Sanction Amount (USD)', ax=axs[0])
sns.scatterplot(data=train, x='Credit Score', y='Loan Sanction Amount (USD)', ax=axs[1])
axs[0].set_title("Income vs Loan Sanction")
axs[1].set_title("Credit Score vs Loan Sanction")
plt.tight_layout()
plt.show()

# 3. Correlation Heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(train.corr(), annot=True, cmap='coolwarm')
plt.title("Feature Correlation Heatmap")
plt.show()

# 4. Actual vs Predicted
model.fit(X_train, y_train)
y_pred = model.predict(X_val)
plt.figure(figsize=(8, 6))
plt.scatter(y_val, y_pred, alpha=0.6)
plt.xlabel("Actual Loan Amount")
plt.ylabel("Predicted Loan Amount")
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'r--')
plt.title("Actual vs Predicted")
plt.show()

# 5. Residual Plot
residuals = y_val - 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()

# 6. Boxplots (e.g., for income and loan)
fig, axs = plt.subplots(1, 2, figsize=(14, 6))
sns.boxplot(y=train['Income (USD)'], ax=axs[0])
sns.boxplot(y=train['Loan Sanction Amount (USD)'], ax=axs[1])
axs[0].set_title("Income Boxplot")
axs[1].set_title("Loan Sanction Amount Boxplot")
plt.tight_layout()
plt.show()

# 7. Bar Plot of Feature Coefficients
coeffs = pd.Series(model.coef_, index=X.columns)
coeffs.sort_values().plot(kind='barh', figsize=(10, 8))
plt.title("Feature Coefficients")
plt.show()
```

6 Included Plots

6.1 Histogram / Distribution Plot of Loan Amount

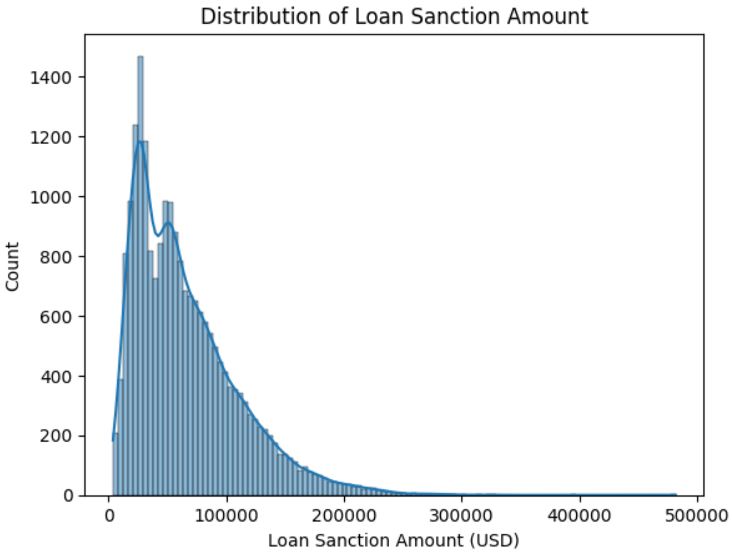

Figure 1: Distribution of Loan Amounts

Interpretation: The histogram shows that the loan amount distribution is right-skewed, with most values concentrated near the starting $100,000.

6.2 Scatter Plot of Income vs Loan Amount

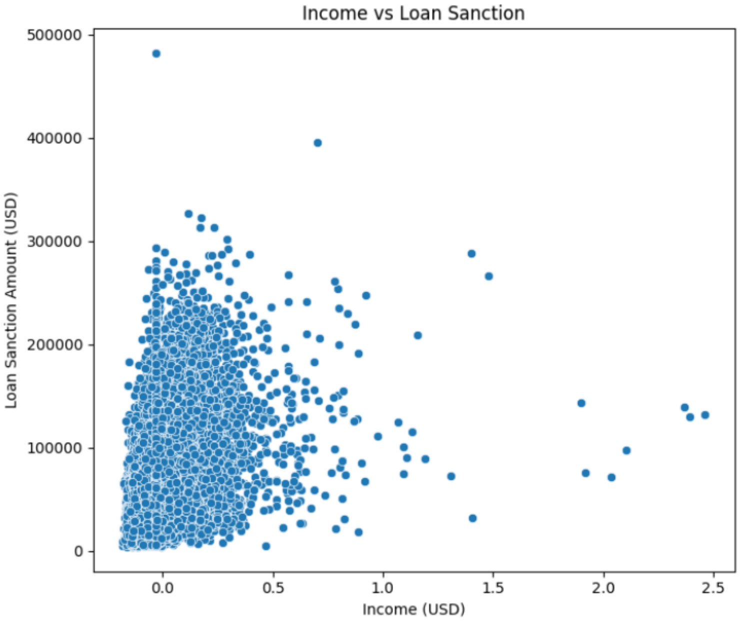

Figure 2: Scatter Plot of Applicant Income vs Loan Amount

Interpretation: There appears to be a weak positive correlation between income and loan amount. Higher income does not always guarantee higher loan amounts, suggesting other features influence the sanctioning process.

6.3 Correlation Heatmap of Features

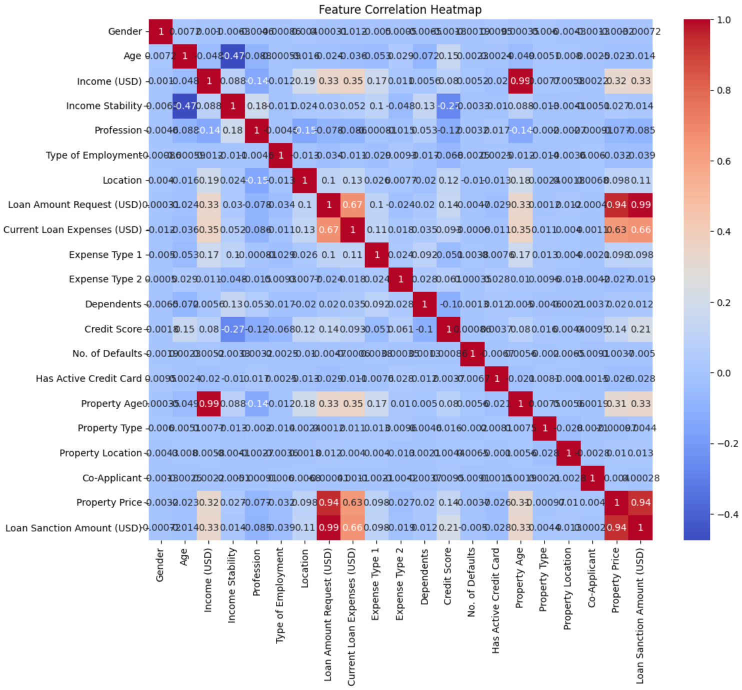

Figure 3: Correlation Heatmap of Dataset Features

Interpretation: The heatmap indicates that loan amount has moderate correlation with some of the features like requested loan amount, income etc.

6.4 Actual vs Predicted Loan Amount

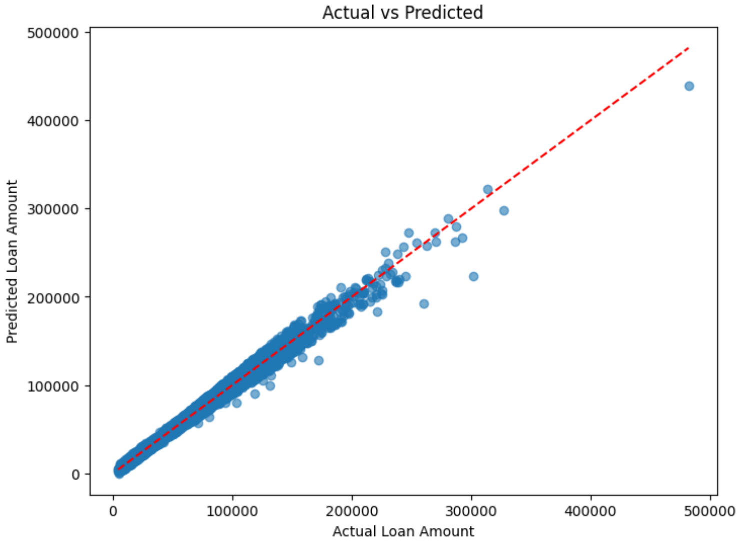

Figure 4: Actual vs Predicted Loan Amount

Interpretation: The points lie close to the diagonal, indicating good prediction performance.

6.5 Residual Plot

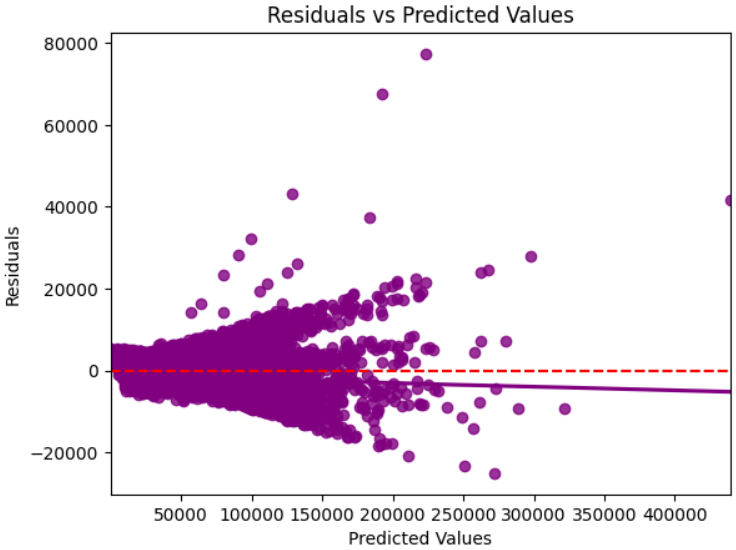

Figure 5: Residuals vs Predicted Values

Interpretation: The residuals appear randomly scattered around zero, which validates the assumptions of linear regression and indicates a decent model fit without systematic error.

6.6 Boxplot of Income

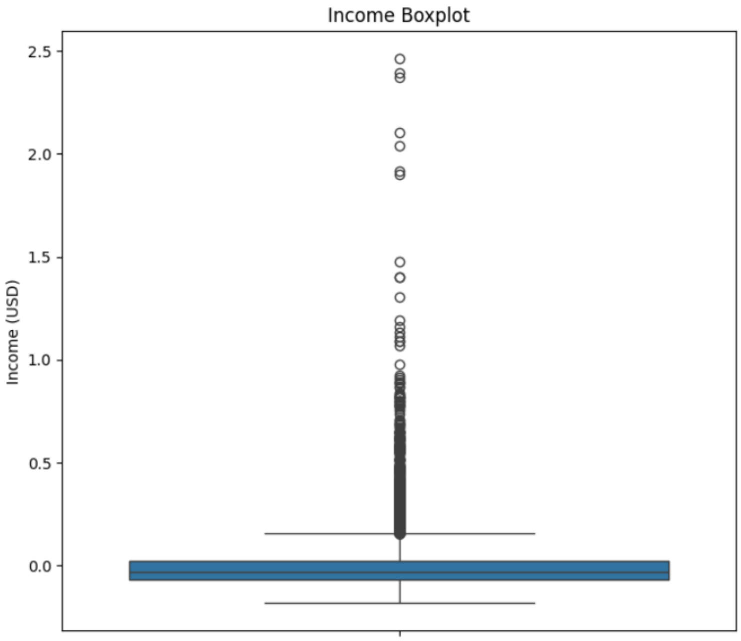

Figure 6: Boxplot of Applicant Income and Loan Amount

Interpretation: The plot show the presence of outliers. It has a tighter interquartile range, suggesting most loans fall within a narrow band.

6.7 Bar Plot of Feature Coefficients

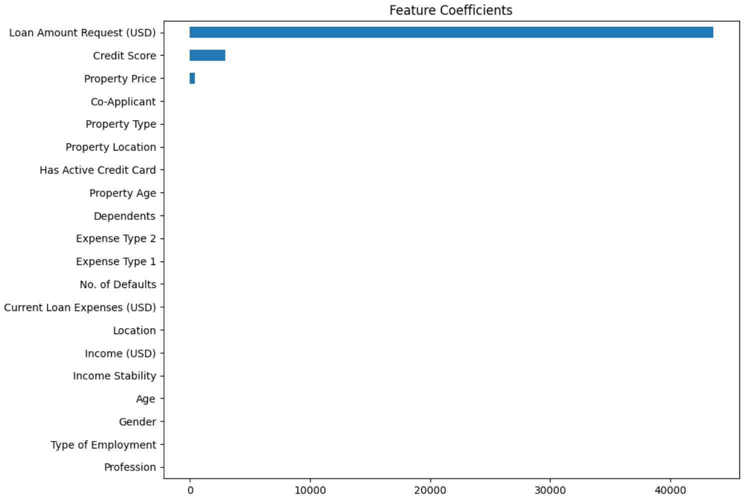

Figure 7: Bar Plot of Feature Coefficients

Interpretation: This plot reveals the influence of each feature in the linear regression model. requested loan amount, credit score, property price seem to contribute the most to the loan amount prediction.

7 Results Tables

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

Fold	MAE	MSE	RMSE	R^2 Score	Adjusted R^2 Score
Fold 1	3579.903348	2.799154×10^7	5290.702960	0.986616	0.986563
Fold 2	3594.835461	2.819211×10^7	5309.624612	0.985864	0.985793
Fold 3	3596.358311	2.678876×10^7	5175.785575	0.986304	0.986245
Fold 4	3630.981504	2.844975×10^7	5333.830562	0.985877	0.985821
Fold 5	3686.128595	2.948840×10^7	5430.322687	0.985282	0.985233
Average	3619.643444	2.818211×10^7	5308.053279	0.985989	0.985931

Table 2: Summary of Results

Description	Student’s Result
Dataset Size (after preprocessing)	21457
Train/Test Split Ratio	80:20 (K = 5 in K-Fold)
Features Used	’Gender’, ’Age’, ’Income (USD)’, ’Income Stability’, ’Profession’, ’Type of Employment’, ’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
Number of Folds	5
MAE (Test Set)	3619.643444
MSE (Test Set)	2.818211×10^7
RMSE (Test Set)	5308.053279
R^2 Score (Test Set)	0.985997
Adjusted R^2 Score	0.985931
Most Influential Feature(s)	Loan Amount Requested (USD), Credit Score, Property Price
Observations from Residual Plot	Residual spread increases as predicted income increases, which implies heteroscedasticity (i.e. variance is not constant).
Overfitting/Underfitting?	Good Fit
Justification (if any)	Very similar metrics for both train and test set.

8 Best Practices

- Always handle missing values, check data quality and remove any outliers.

- Normalize or scale features before training.
- Use residual plots to check assumptions of linearity and homoscedasticity.
- Evaluate the model using multiple metrics.
- Use cross-validation for better generalization estimates.

9 Learning Outcomes

- Understood the full pipeline of building a regression model.
- Practiced data cleaning, preprocessing, and feature encoding.
- Visualized data and regression results effectively.
- Learned to interpret and explain model performance using standard metrics.
- Compared predicted vs actual outcomes and derived insights.