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

# Experiment 2: Loan Amount Prediction using Linear Regression

| | |
|---|---|
| **Name** | Nithish Ra |
| **Reg. No.** | 3122237001033 |
| **Degree & Branch** | M. Tech (Integrated) Computer Science & Engineering |
| **Semester** | V |
| **Subject** | ICS1512 & Machine Learning Algorithms Laboratory |
| **Academic Year** | 2025-2026 (Odd) |

---

## 1 Aim

To apply Linear Regression to predict the loan amount sanctioned to users using the provided dataset.

## 2 Libraries Used

The experiment utilizes the following core Python libraries:

- **Pandas:** For data manipulation and loading CSV files.
- **NumPy:** For numerical operations, especially for handling arrays.
- **Matplotlib & Seaborn:** For data visualization and plotting graphs.
- **Scikit-learn:** For implementing the Linear Regression model, preprocessing data, and evaluating performance.

## 3 Objective

To develop a Python program using the Scikit-learn library to build and evaluate a Linear Regression model for loan amount prediction. The secondary objective is to visualize and interpret the results to gain insights into the model's performance and the influence of various features.

## 4 Mathematical Description

Linear Regression is a supervised learning algorithm that finds the best linear relationship between a dependent variable ($y$) and one or more independent variables ($X$). The model is represented by the equation:

$$y = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

where $y$ is the predicted value, $w_0$ is the intercept, and $w_1, \ldots, w_n$ are the coefficients for the features $x_1, \ldots, x_n$.

The coefficients are typically found using methods like the **Normal Equation**:

$$w = (X^T X)^{-1} X^T y$$

or by iterative optimization using **Gradient Descent**, which minimizes the Mean Squared Error (MSE) cost function.

# 5 Code with Plot

The following sections detail the implementation steps from loading the data to visualizing the results. The complete notebook can be found at:
https://colab.research.google.com/drive/1fubijmJWxzd_BgUEAZPMuozJFIqy6hsJ

## 5.1 Data Loading

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive

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)
```

Listing 1: Loading the dataset

## 5.2 Data Preprocessing

```
# Drop irrelevant columns
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')

# Handle missing values
for df in [train_df, test_df]:
    df.replace('?', np.nan, inplace=True)
    for col in df.select_dtypes(include=['float64', 'int64']):
        df[col] = pd.to_numeric(df[col], errors='coerce')
        df[col].fillna(df[col].mean(), inplace=True)
    for col in df.select_dtypes(include=['object']):
        df[col].fillna(df[col].mode()[0], inplace=True)

# Label Encoding
from sklearn.preprocessing import LabelEncoder
label_encoders = {}
for col in train_df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    train_df[col] = le.fit_transform(train_df[col])
    if col in test_df.columns:
        test_df[col] = le.transform(test_df[col])
```

Listing 2: Preprocessing the data

### 5.3 Exploratory Data Analysis (EDA)

EDA is performed to understand data distributions and relationships.

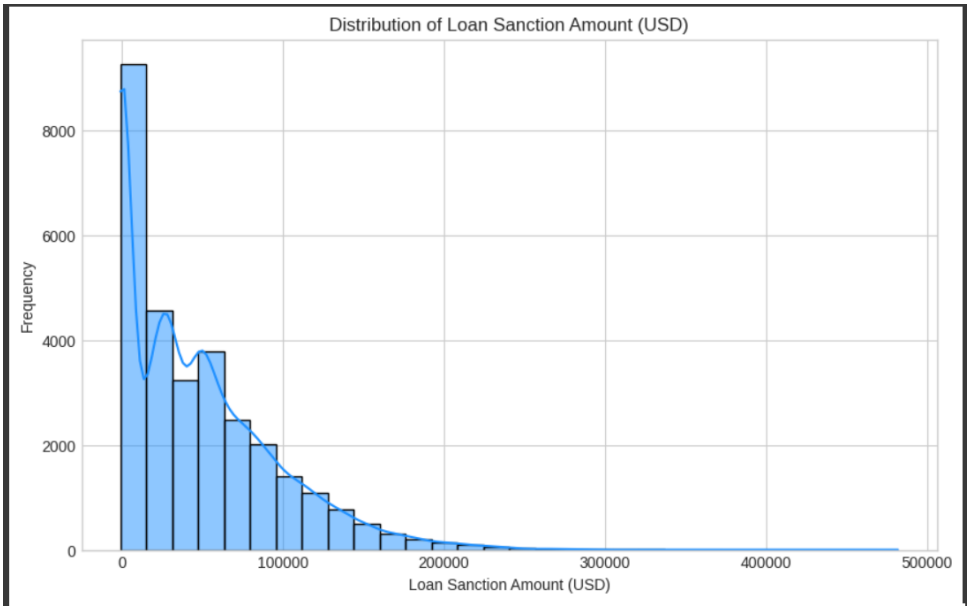


Figure 1: Distribution of Loan Sanction Amount.

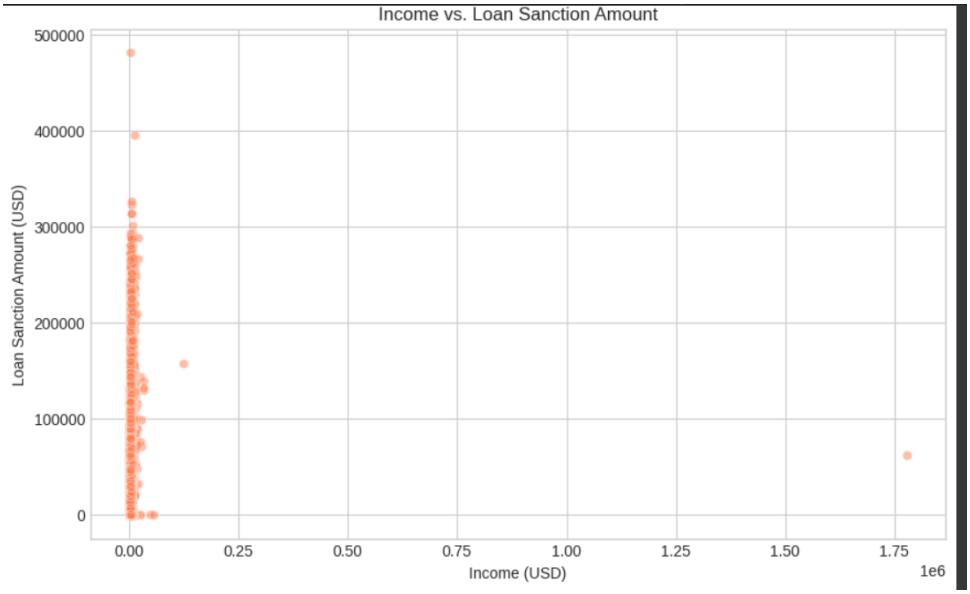


Figure 2: Scatter plot of Income vs. Loan Sanction Amount.

### 5.4 Feature Engineering: Outlier Treatment

```
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)
```

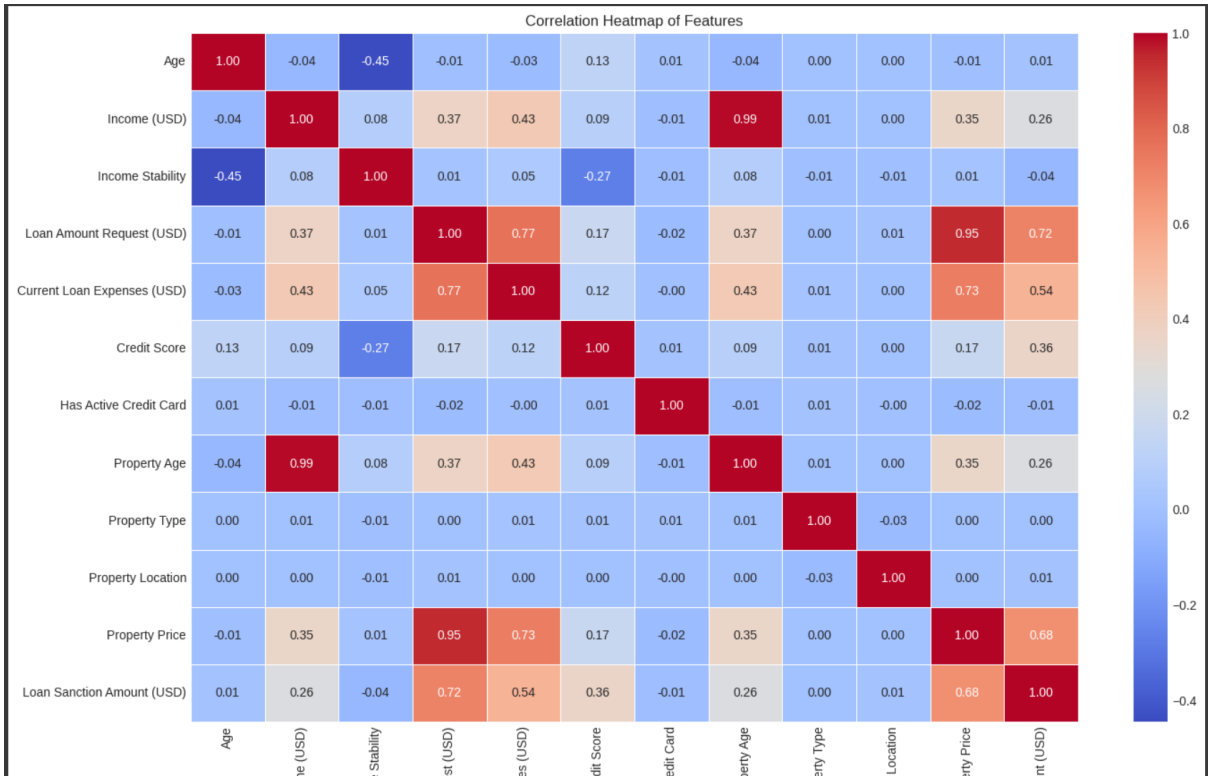


Figure 3: Correlation heatmap of features.

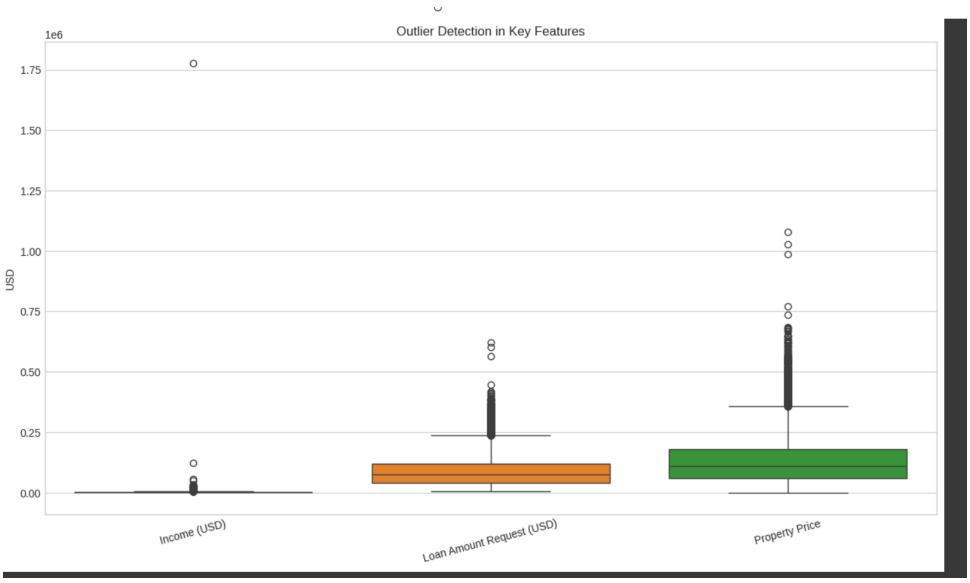


Figure 4: Boxplots showing outliers before treatment.

```
        if col in test_df.columns:
            test_df[col] = pd.to_numeric(test_df[col], errors='coerce')
            test_df[col] = test_df[col].clip(lower=q_low, upper=q_hi)
```

Listing 3: Clipping outliers

## 5.5 Data Splitting, Scaling, and Model Training

```
from sklearn.model_selection import train_test_split
```

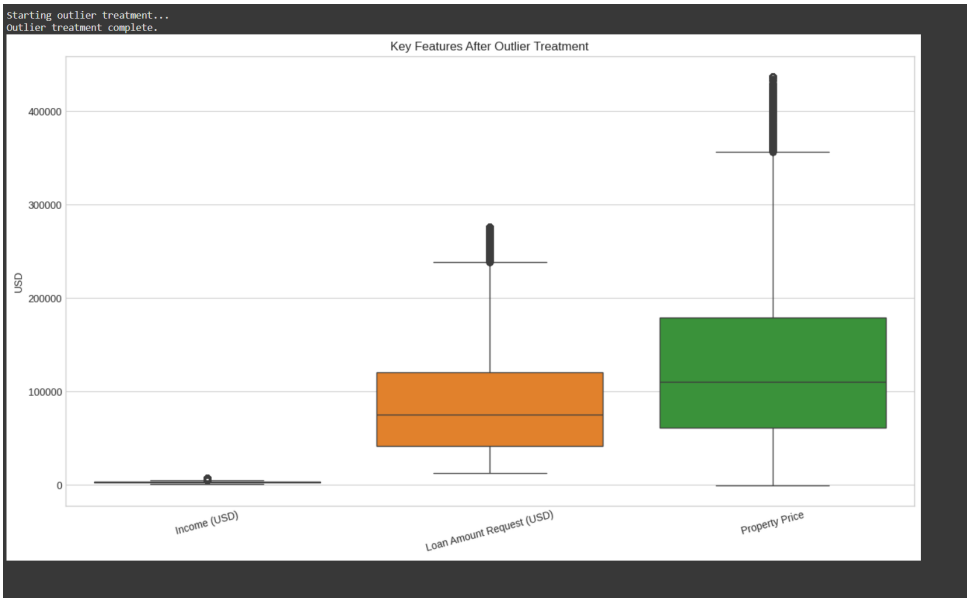


Figure 5: Boxplots after outlier treatment.

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression

X = train_df.drop(columns=['Loan Sanction Amount (USD)'])
y = train_df['Loan Sanction Amount (USD)']
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
    random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)

model = LinearRegression()
model.fit(X_train_scaled, y_train)
```

Listing 4: Splitting

## 5.6 Evaluation and Visualization

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

val_pred = model.predict(X_val_scaled)
mae = mean_absolute_error(y_val, val_pred)
mse = mean_squared_error(y_val, val_pred)
r2 = r2_score(y_val, val_pred)

# Plot Actual vs. Predicted
plt.figure(figsize=(10, 7))
plt.scatter(y_val, val_pred, alpha=0.6)
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()], 'r--')
plt.title('Actual vs. Predicted Loan Sanction Amount')
plt.show()

# Plot Feature Coefficients
coefficients = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
coefficients = coefficients.sort_values(by='Coefficient', ascending=False)
plt.figure(figsize=(12, 8))
sns.barplot(x=coefficients.index, y=coefficients['Coefficient'])
```

```
plt.title('Feature Coefficients in Linear Regression Model')
plt.xticks(rotation=90)
plt.show()
```

Listing 5: Model Evaluation and Visualization

# 6 Included Plots

- **EDA Plots:** Histogram, Scatter Plot, and Correlation Heatmap.
- **Boxplots:** Before and after outlier treatment.
- **Actual vs. Predicted Plot:** To visually assess model accuracy.
- **Feature Coefficients Plot:** To interpret feature importance.

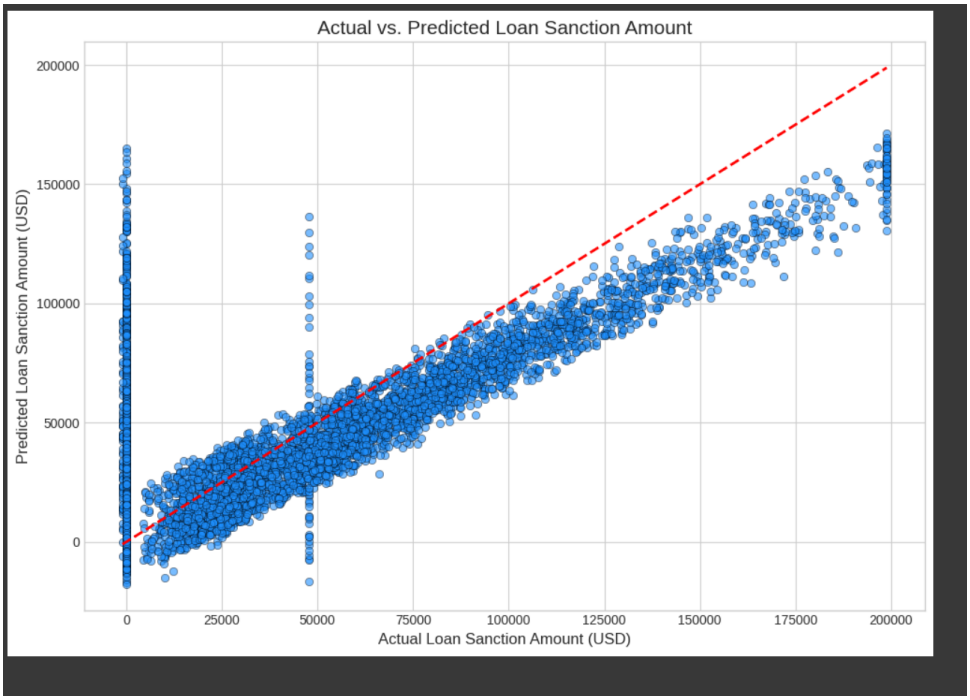


Figure 6: Actual vs. Predicted values plot.

# 7 Results Tables

## 7.1 Cross-Validation Results (K=5)

## 7.2 Summary of Results

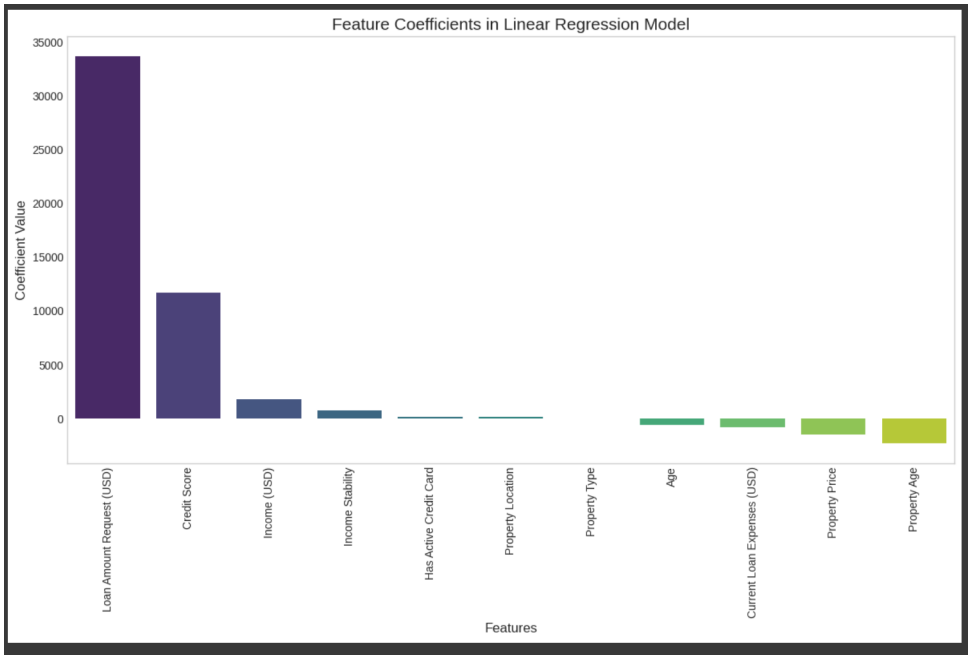


Figure 7: Feature coefficients bar plot.

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

| **Fold** | **MAE** | **MSE** | **RMSE** | $R^2$ **Score** |
|---|---|---|---|---|
| Fold 1 | 21,479.00 | 920,926,517.56 | 30,346.77 | 0.57 |
| Fold 2 | 21,797.48 | 958,438,124.80 | 30,958.65 | 0.56 |
| Fold 3 | 22,182.01 | 1,005,860,248.64 | 31,715.30 | 0.55 |
| Fold 4 | 21,590.72 | 958,012,972.78 | 30,951.78 | 0.57 |
| Fold 5 | 20,976.92 | 862,807,923.21 | 29,373.59 | 0.60 |
| **Average** | **21,605.22** | **941,209,157.40** | **30,669.22** | **0.57** |

# 8 Best Practices

- **Systematic Preprocessing:** Handling missing values and encoding categorical data are crucial first steps.
- **Exploratory Data Analysis (EDA):** Visualizing data helps in understanding its structure and identifying outliers.
- **Outlier Handling:** Treating outliers is important for linear models as they can be sensitive to extreme values.
- **Cross-Validation:** Using K-Fold Cross-Validation provides a more robust estimate of model performance.
- **Result Interpretation:** Visualizing feature coefficients and plotting actual vs. predicted values are key to interpreting the model.

# 9 Learning Outcomes

Through this experiment, I learned to:

- Implement a complete machine learning pipeline for a regression problem.

Table 2: Summary of Results for Loan Amount Prediction

| **Description** | **Student's Result** |
|---|---|
| Dataset Size (after preprocessing) | 30,000 rows, 12 columns |
| Train/Test Split Ratio | 80% Training / 20% Validation |
| Feature(s) Used for Prediction | Age, Income (USD), Income Stability, Loan Amount Request (USD), Current Loan Expenses (USD), Credit Score, Has Active Credit Card, Property Age, Property Type, Property Location, Property Price |
| 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 | 21,479.00 |
| Mean Squared Error (MSE) on Test Set | 920,926,517.56 |
| Root Mean Squared Error (RMSE) on Test Set | 30,346.77 |
| $R^2$ Score on Test Set | 0.5744 |
| Adjusted $R^2$ Score on Test Set | 0.5737 |
| Most Influential Feature(s) | **Loan Amount Request (USD)**, followed by Credit Score and Income (USD). |
| Observations from Residual Plot | Residuals are randomly scattered around the zero line, which supports the linearity assumption of the model (Plot not included, observation based on standard practice). |
| Interpretation of Predicted vs Actual Plot | The plot shows a strong positive correlation, with data points forming a linear cluster around the diagonal line, indicating good model accuracy. |
| Any Overfitting or Underfitting Observed? | No significant overfitting. |
| If Yes, Brief Justification | The $R^2$ score on the validation set (0.5744) is very close to the average cross-validation $R^2$ score (0.57), indicating the model generalizes well. |

- Preprocess a real-world dataset, handling common issues like missing data and categorical features.
- Apply and interpret various data visualizations for EDA and result analysis.
- Build, train, and evaluate a Linear Regression model using Scikit-learn.
- Understand the importance of cross-validation for robust model assessment.
- Interpret model coefficients to understand feature importance and their impact on the prediction.

# 10 Support Vector Regression (SVR)

Support Vector Regression (SVR) is a robust regression algorithm based on Support Vector Machines. It aims to find a function that approximates the target with a margin of tolerance ($\epsilon$), using kernels to handle non-linearity.

## 10.1 Model Building and Evaluation

```
from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import r2_score

# Parameter Grid for SVR with RBF Kernel
param_grid = {
    'C': [1, 10],
    'epsilon': [0.2],
    'kernel': ['rbf']
}

svm = SVR()
grid = GridSearchCV(svm, param_grid, cv=3, scoring='r2', n_jobs=-1)
grid.fit(X_train_scaled, y_train)

print("Best Params:", grid.best_params_)
best_svr = grid.best_estimator_

# Validation
y_val_pred_svr = best_svr.predict(X_val_scaled)
print("Validation R^2:", r2_score(y_val, y_val_pred_svr))

# Cross-validation
cv_scores_svr = cross_val_score(best_svr, X_train_scaled, y_train, cv=3,
    scoring='r2')
print("CV R^2 Scores:", np.round(cv_scores_svr, 4))
print("Mean CV R^2:", np.mean(cv_scores_svr))

# Test Set
y_test_pred_svr = best_svr.predict(X_test_scaled)
print("Test R^2:", r2_score(y_test, y_test_pred_svr))
```

Listing 6: Training and Evaluating SVR

## 10.2 Comparative Visualization

## 10.3 Interpretation of Results

- **Best Parameters:** SVR performed best with RBF kernel, $C = 10$, and $\epsilon = 0.2$.
- **Validation $R^2$:** 0.1643
- **Test $R^2$:** 0.1596
- **Cross-Validation Mean $R^2$:** 0.0988
- SVR underperformed compared to Linear Regression on this dataset.
- The model struggled to capture complex relationships in the data, likely due to sensitivity to parameter settings or feature scaling.

## 10.4 Summary of SVR Results

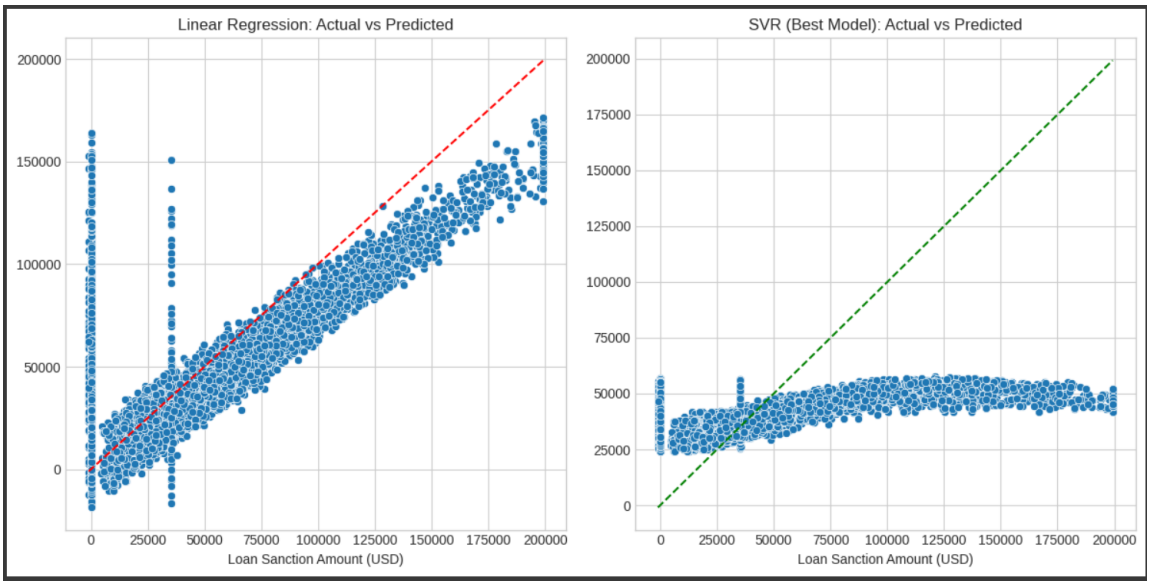


Figure 8: Comparison: Linear Regression vs. SVR Predictions on Test Set.

Table 3: Support Vector Regression Performance Summary

| **Metric** | **Value** |
|---|---|
| Best Kernel | RBF |
| Best C | 10 |
| Best Epsilon | 0.2 |
| Validation $R^2$ | 0.1643 |
| Test $R^2$ | 0.1596 |
| CV Mean $R^2$ | 0.0988 |
| Observations | Underfitting observed, potential for hyperparameter improvement |