Sri Sivasubramaniya Nadar College of Engineering, Chennai

(An Autonomous Institution Affiliated to Anna University)

Degree & Branch: Integrated M.Tech. Computer Science & Engineering
Semester: V

Course Code & Title: ICS1512 - Machine Learning Algorithms Laboratory

Academic Year: 2025-2026 (Odd) **Batch:** 2023-2028

Name: Vishwajith L K **Reg. No.:** 3122237001061

Experiment 2: Loan Amount Prediction using Linear Regression

1. Aim

To predict loan amounts using a Linear Regression model by analyzing historical loan data, carrying out exploratory data analysis (EDA), and identifying key factors influencing the sanctioned loan amounts.

2. Libraries Used

- **NumPy**: For numerical operations and matrix manipulations
- **Pandas**: For data cleaning, preprocessing, and analysis
- **Scikit-learn**: For implementing Linear Regression and evaluating metrics
- **Matplotlib & Seaborn**: For data visualization and statistical plots

3. Objectives

- Perform preprocessing and cleaning of loan data
- Conduct EDA to better understand relationships between variables
- Implement a Linear Regression model for predicting loan sanction amounts
- Evaluate model performance using error metrics and residual analysis
- Interpret the results and identify the most influential features

4. Mathematical Description

The Linear Regression model assumes a linear relationship between features and the target variable. It is represented by the equation:

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

Where:

- y is the predicted loan amount
- β_0 is the intercept term
- $\beta_1, \beta_2, \ldots, \beta_n$ are the model coefficients
- $x_1, x_2, \ldots, x_n$ are the feature values
- ϵ is the error term

The coefficients are estimated using the **Ordinary Least Squares (OLS)** method by minimizing the sum of squared residuals:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$$

5. Code Implementation

1. Loading the Dataset

```
import pandas as pd
import numpy as np
import sklearn as sk
from sklearn.linear_model import LinearRegression

training_df = pd.read_csv('/kaggle/input/predict-loan-amount-data/train.csv')
training_df.drop(['Customer ID', 'Name'], axis=1, inplace=True)
training_df.head(10)
```

	Gender	Age	Income (USD)	Income Stability	Profession	Type of Employment	Location	Loan Amount Request (USD)	Current Loan Expenses (USD)	Expense Type 1	...	Credit Score	No. of Defaults	Has Active Credit Card
0	F	56	1933.05	Low	Working	Sales staff	Semi-Urban	72809.58	241.08	N	...	809.44	0	NaN
1	M	32	4952.91	Low	Working	NaN	Semi-Urban	46837.47	495.81	N	...	780.40	0	Unpossessed
2	F	65	988.19	High	Pensioner	NaN	Semi-Urban	45593.04	171.95	N	...	833.15	0	Unpossessed
3	F	65	NaN	High	Pensioner	NaN	Rural	80057.92	298.54	N	...	832.70	1	Unpossessed
4	F	31	2614.77	Low	Working	High skill tech staff	Semi-Urban	113858.89	491.41	N	...	745.55	1	Active
5	F	60	1234.92	Low	State servant	Secretaries	Rural	34434.72	181.48	N	...	684.12	1	Inactive
6	M	43	2361.56	Low	Working	Laborers	Semi-Urban	152561.34	697.67	Y	...	637.29	0	Unpossessed
7	F	45	NaN	Low	State servant	Managers	Semi-Urban	240311.77	807.64	N	...	812.26	0	Active
8	F	38	1296.07	Low	Working	Cooking staff	Rural	35141.99	155.95	N	...	705.29	1	Active
9	M	18	1546.17	Low	Working	Laborers	Rural	42091.29	500.20	N	...	613.24	0	Unpossessed

Figure 1: Dataset

2. Data Preprocessing

```
from sklearn.preprocessing import LabelEncoder
# Handle missing values
for col in training_df.select_dtypes(include='object').columns:
    training_df[col].fillna(training_df[col].mode()[0], inplace=True)

for col in training_df.select_dtypes(include=np.number).columns:
    training_df[col].fillna(training_df[col].mean(), inplace=True)

# Encode categorical variables
label_enc = LabelEncoder()
for col in training_df.select_dtypes(include='object').columns:
    training_df[col] = label_enc.fit_transform(training_df[col])
```

	Gender	Age	Income (USD)	Income Stability	Profession	Type of Employment	Location	Loan Amount Request (USD)	Current Loan Expenses (USD)	Expense Type 1	...	Credit Score	No. of Defaults	Has Active Credit Card
0	0	56	1933.050000	1	7	14	1	72809.58	241.08	0	...	809.44	0	0
1	1	32	4952.910000	1	7	8	1	46837.47	495.81	0	...	780.40	0	2
2	0	65	988.190000	0	3	8	1	45593.04	171.95	0	...	833.15	0	2
3	0	65	2630.574417	0	3	8	0	80057.92	298.54	0	...	832.70	1	2
4	0	31	2614.770000	1	7	6	1	113858.89	491.41	0	...	745.55	1	0
5	0	60	1234.920000	1	4	15	0	34434.72	181.48	0	...	684.12	1	1
6	1	43	2361.560000	1	7	8	1	152561.34	697.67	1	...	637.29	0	2
7	0	45	2630.574417	1	4	10	1	240311.77	807.64	0	...	812.26	0	0
8	0	38	1296.070000	1	7	2	0	35141.99	155.95	0	...	705.29	1	0
9	1	18	1546.170000	1	7	8	0	42091.29	500.20	0	...	613.24	0	2

10 rows × 22 columns

Figure 2: Preprocessed Data

3. Exploratory Data Analysis

```
#Histogram: To understand the distribution of loan amounts
from matplotlib import pyplot as plt
training_df['Loan Amount Request (USD)'].plot(kind='hist', bins=20, title='Loan Amoun
plt.show()
training_df['Loan Sanction Amount (USD)'].plot(kind='hist', bins=20, title='Loan Sanc
plt.show()

#Scatter Plots: To examine the relationship between key features and the loan amount.
training_df.plot(kind='scatter', x='Income (USD)', y='Loan Sanction Amount (USD)',tit
plt.show()
training_df.plot(kind='scatter', x='Credit Score', y='Loan Sanction Amount (USD)',tit
plt.show()

#Correlation Heatmap: To identify multicollinearity and relationships among features.
import seaborn as sns
plt.figure(figsize=(12, 8))
sns.heatmap(training_df.corr(),annot=True, fmt=".2f")
plt.show()
```

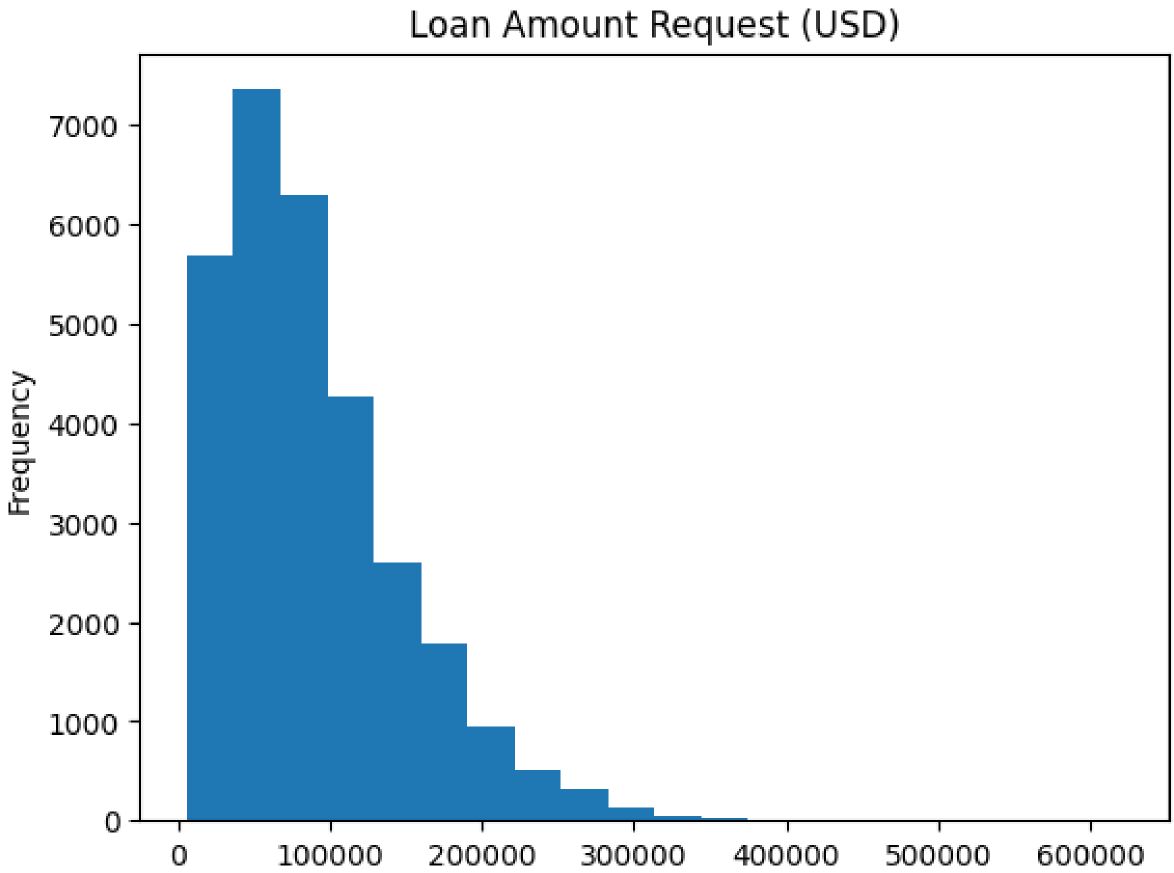

Figure 3: Histogram: Loan Amount Request (USD)

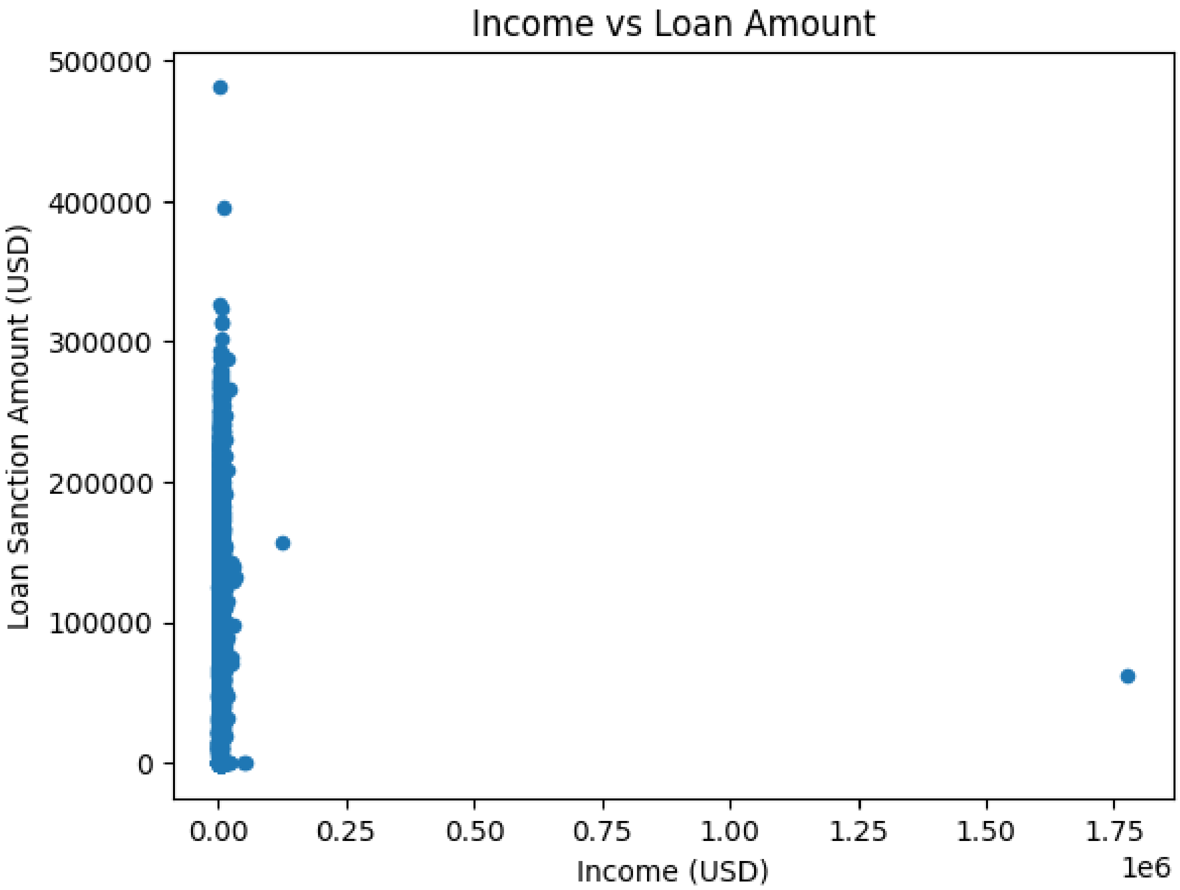

Figure 4: Histogram: Loan Sanction Amount (USD)

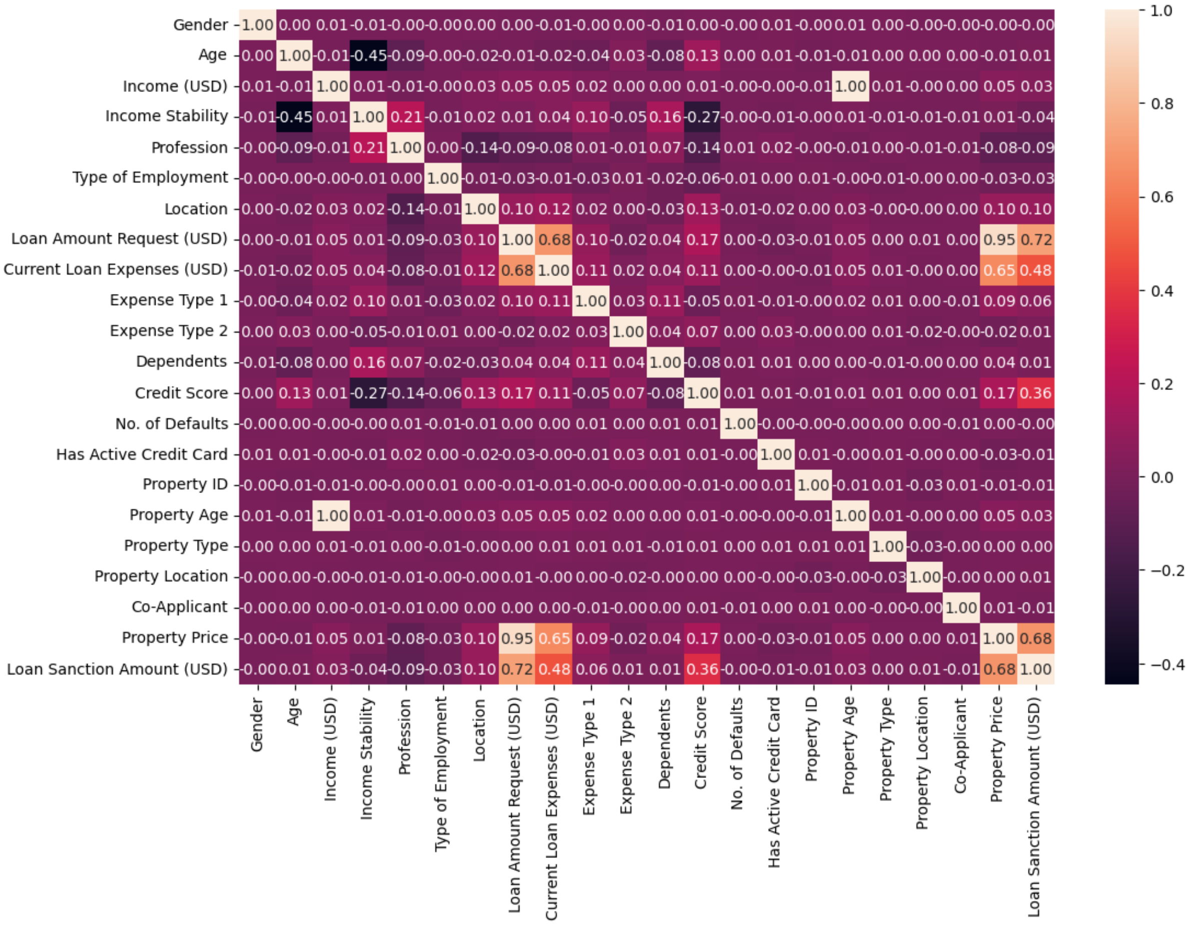

Figure 5: Scatter Plot and Correlation Heatmap

4. Feature Engineering

```
#Boxplots: To identify outliers in numerical features such as income.
training_df['Income (USD)'].plot(kind='box',title='Outliers in Income')
plt.show()

#handling outliers
def detect_outliers_iqr(data):
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return (data < lower_bound) | (data > upper_bound)

outliers_income = detect_outliers_iqr(training_df['Income (USD)'])
training_df.loc[outliers_income, 'Income (USD)'] = training_df['Income (USD)'].median
training_df['Income (USD)'].plot(kind='box',title='Outliers in Income')
plt.show()
```

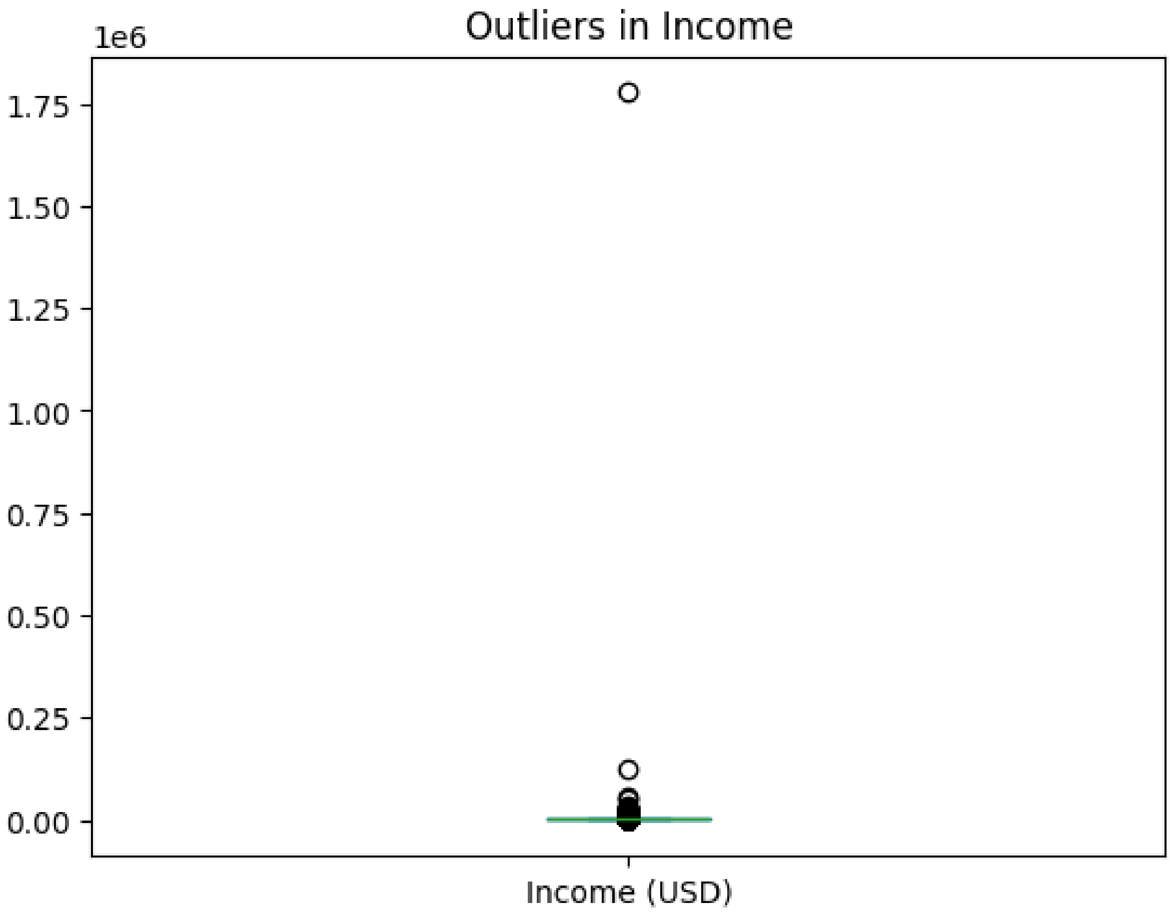

Figure 6: Before Handling Outliers

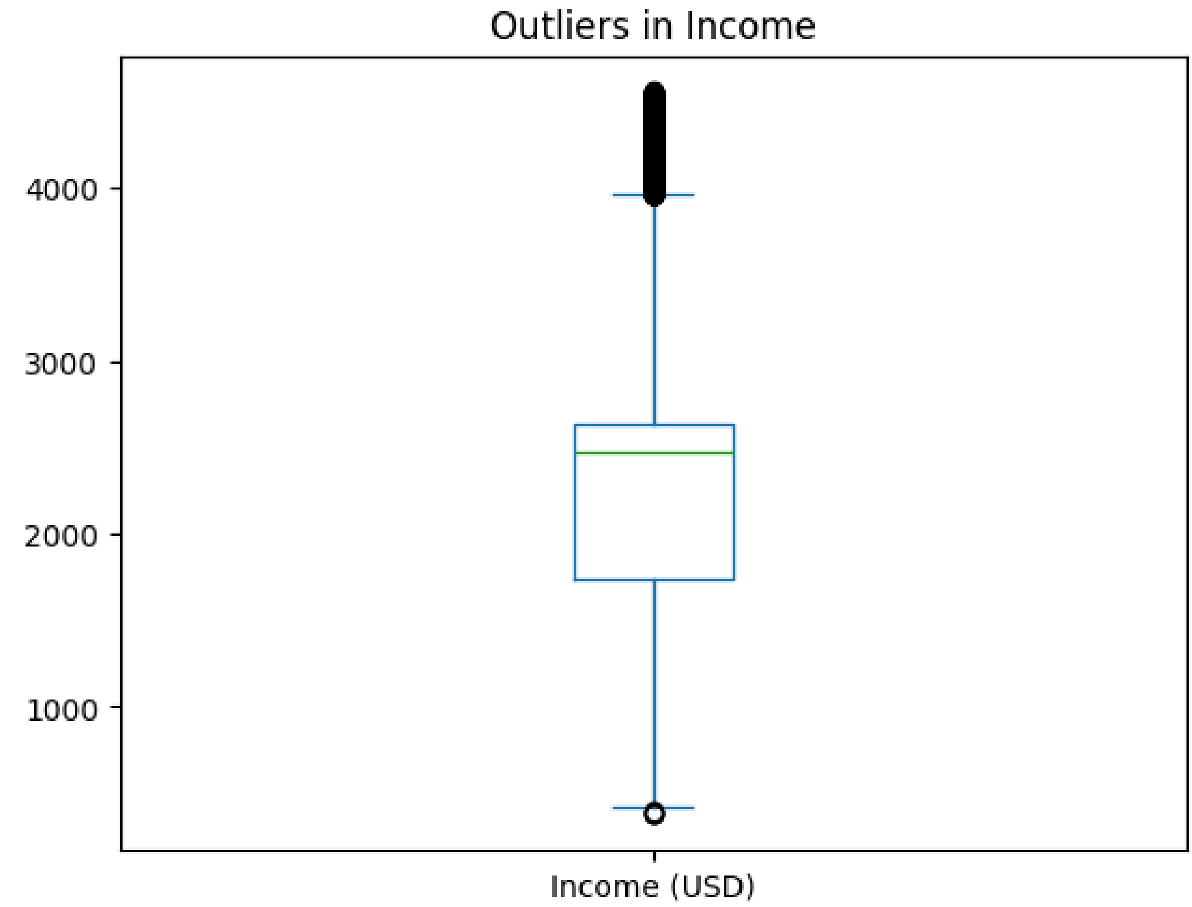

Figure 7: After Handling Outliers

5. Split the dataset

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X = training_df.drop(columns=['Loan Sanction Amount (USD)'])
features = X.columns
Y = training_df['Loan Sanction Amount (USD)']
scaler = StandardScaler()
X = scaler.fit_transform(X)
x_train,x_temp, y_train, y_temp = train_test_split(X,Y,test_size=0.2)
x_val,x_test, y_val, y_test = train_test_split(x_temp,y_temp,test_size=0.5)
```

6. Model Training

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
```

7. Model Evaluation

```
from sklearn.model_selection import KFold, cross_val_score
import numpy as np
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score,root_mean_

MAE = mean_absolute_error(y_true= y_test,y_pred=y_pred)
MSE = mean_squared_error(y_true= y_test,y_pred=y_pred)
r2 = r2_score(y_true= y_test,y_pred=y_pred)
RMSE = root_mean_squared_error(y_true= y_test,y_pred=y_pred)
adj_r2 = 1 - (1 - r2) * ((len(y_test) - 1) / (len(y_test) - X.shape[1] - 1))

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

mae_scores = []
mse_scores = []
r2_scores = []

print("K-Fold Cross Validation Results:\n")
fold = 1
for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = Y.iloc[train_index], Y.iloc[test_index]

    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

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

    print(f"Fold {fold}: MAE={mae:.2f}, MSE={mse:.2f}, RMSE={rmse:.2f}, R2={r2:.2f}")

    mae_scores.append(mae)
    mse_scores.append(mse)
    r2_scores.append(r2)

    fold += 1

print("\nAverage Results:")
print(f"Average MAE: {np.mean(mae_scores):.2f}")
print(f"Average MSE: {np.mean(mse_scores):.2f}")
print(f"Average RMSE: {np.sqrt(np.mean(mse_scores)):.2f}")
print(f"Average R2 Score: {np.mean(r2_scores):.2f}")
```

```
MAE:  21960.48496835262
MSE:  1028921033.3430872
R2 score:  0.5582306411094848
Root MSE:  32076.798988413528
Adjusted R2 score:  0.5551154105733194
```

Figure 8: Evaluation Metrics on Test Set

```
K-Fold Cross Validation Results:

Fold 1: MAE=21721.46, MSE=992759227.36, RMSE=31508.08, R2=0.57
Fold 2: MAE=21872.96, MSE=979657503.58, RMSE=31299.48, R2=0.57
Fold 3: MAE=22358.09, MSE=1064323575.57, RMSE=32623.97, R2=0.54
Fold 4: MAE=21759.61, MSE=993344822.86, RMSE=31517.37, R2=0.58
Fold 5: MAE=20999.10, MSE=879813659.40, RMSE=29661.65, R2=0.61

Average Results:
Average MAE: 21742.25
Average MSE: 981979757.75
Average RMSE: 31336.56
Average R2 Score: 0.57
```

Figure 9: K-Fold Cross Validation Results

8. Visualization of the Results

```
# Predicted vs Actual and residual plots
...
\end\subsection*{8. Visualization of the Results}
\begin{verbatim}
plt.figure(figsize=(6, 6))
sns.scatterplot(x=y_test, y=y_pred)
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("Predicted vs Actual (Test Set)")
plt.grid(True)
plt.show()
```

```
#Residual Plot
residuals = y_test - y_pred
plt.figure(figsize=(6, 4))
sns.histplot(residuals, kde=True)
plt.title("Residuals Distribution")
plt.xlabel("Residual")
plt.grid(True)
plt.show()

coeff_df = pd.DataFrame({
    'Feature': features,
    'Coefficient': model.coef_
}).sort_values(by='Coefficient', key=abs, ascending=False)

plt.figure(figsize=(12, 6))
sns.barplot(data=coeff_df, x='Coefficient', y='Feature')
plt.title("Feature Importance (Model Coefficients)")
plt.grid(True)
plt.show()
```

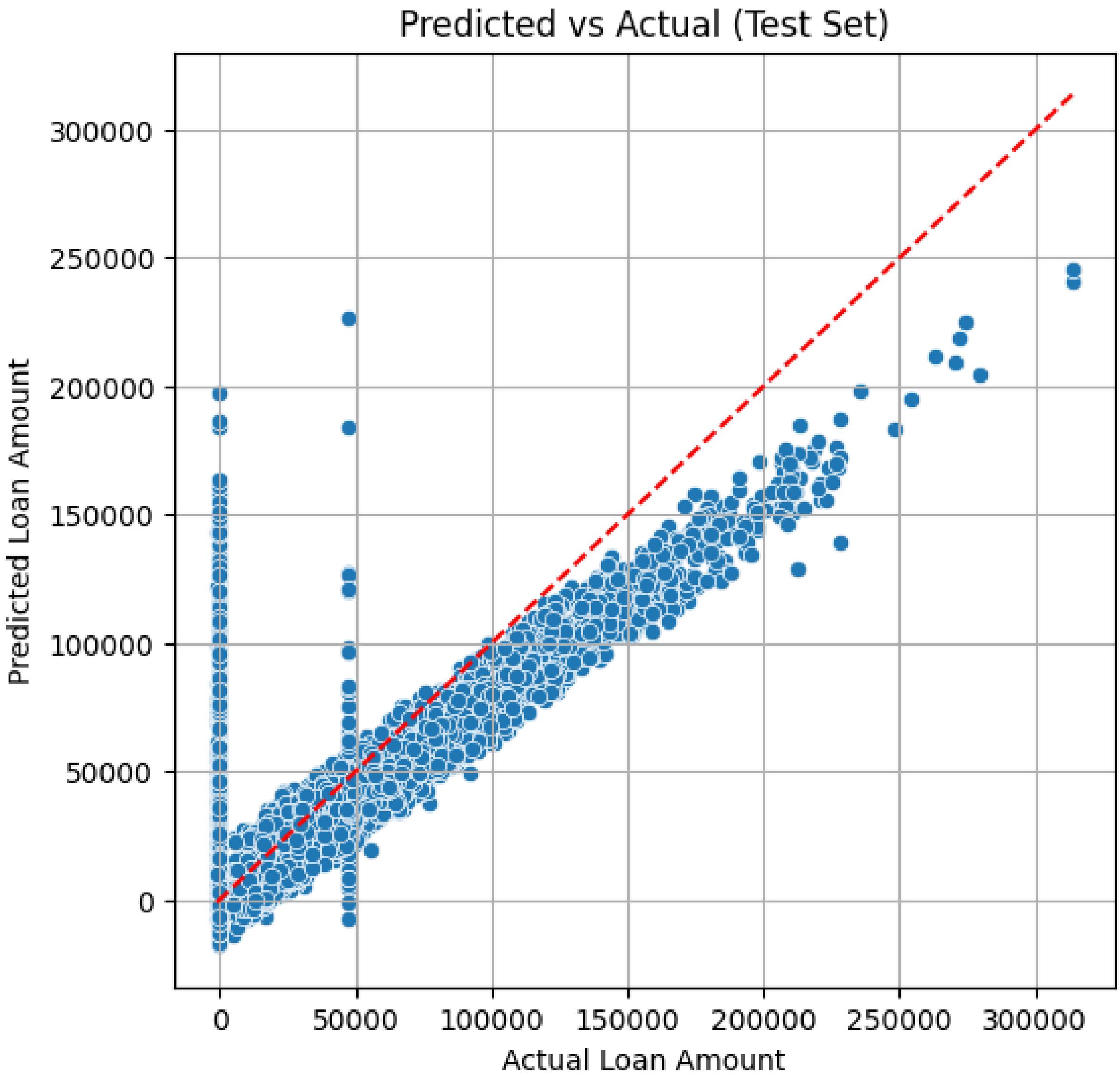

Figure 10: Predicted vs Actual Loan Amount

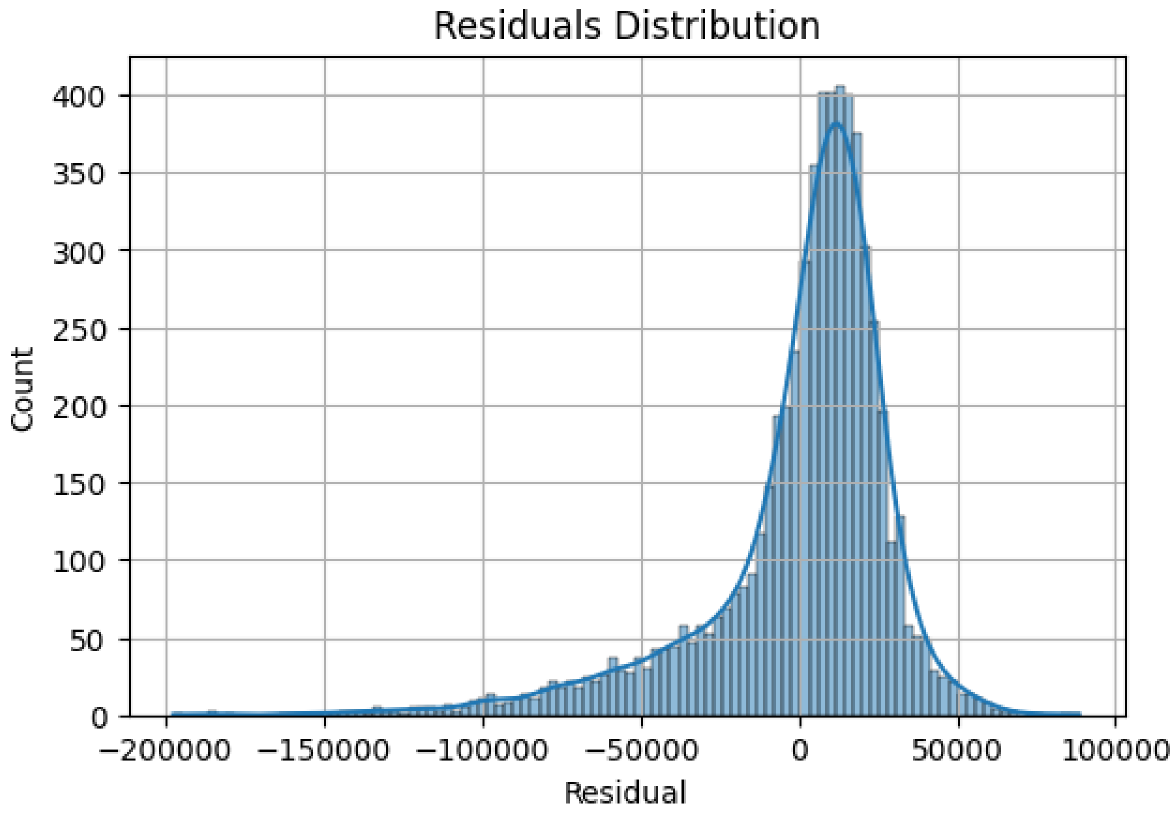

Figure 11: Residual Plot

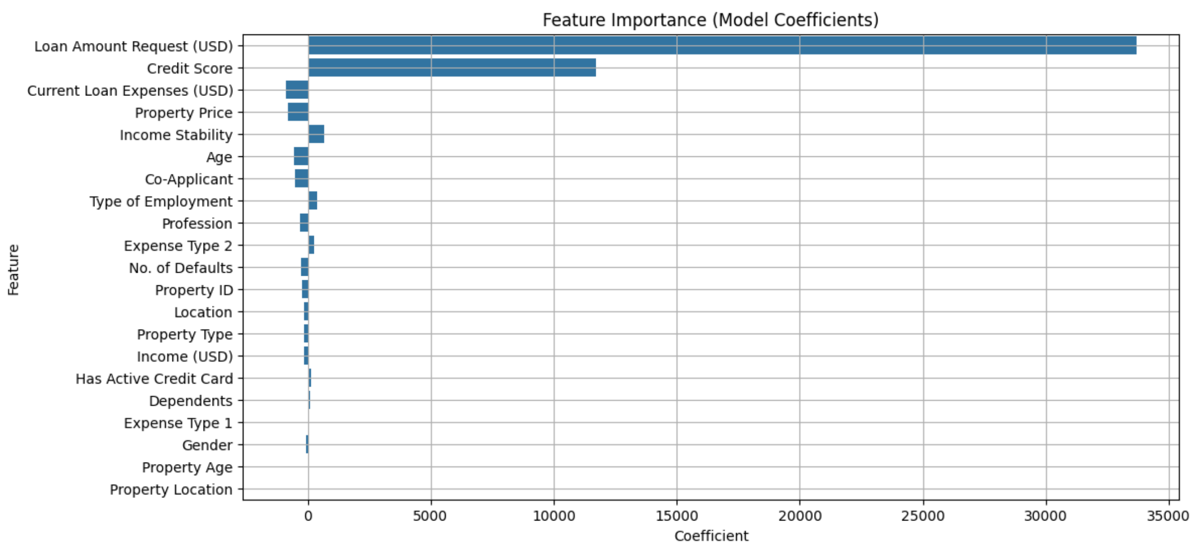

Figure 12: Feature Importance Barplot

6. Results Tables

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

Fold	MAE	MSE	RMSE	R^2 Score
Fold 1	21721.46	9.927592e+07	31508.08	0.57
Fold 2	21872.96	9.796575e+07	31299.48	0.57
Fold 3	22358.09	1.064324e+08	32623.97	0.54
Fold 4	21759.61	9.933448e+07	31517.37	0.58
Fold 5	20999.10	8.798137e+07	29661.65	0.61
Average	**21742.25**	**9.819798e+07**	**31336.56**	**0.57**

7. Best Practices

- Performed thorough data cleaning and preprocessing
- Used train-test split to evaluate model performance
- Implemented feature scaling for better convergence
- Compared multiple evaluation metrics
- Documented all steps and interpretations

Table 2: Summary of Results for Loan Amount Prediction

Description	Student's Result
Dataset Size (after preprocessing)	30,000 rows, 21 columns
Train/Test Split Ratio	80% training, 10% validation, 10% test
Feature(s) Used for Prediction	['Gender', 'Age', 'Income (USD)', 'Income Stability
Model Used	Linear Regression
Cross-Validation Used? (Yes/No)	Yes
If Yes, Number of Folds (K)	5
Reference to CV Results Table	1
Mean Absolute Error (MAE) on Test Set	21960.48
Mean Squared Error (MSE) on Test Set	1028921033.34
Root Mean Squared Error (RMSE) on Test Set	32076.8
R^2 Score on Test Set	0.558
Adjusted R^2 Score on Test Set	0.555
Observations from Residual Plot	Residuals randomly distributed
Interpretation of Predicted vs Actual Plot	Linear relationship
Any Overfitting or Underfitting Observed?	No

9. Learning Outcomes

- Understood how to implement Linear Regression in a real-world financial dataset
- Learned the role of data preprocessing and feature selection in model performance
- Developed skills in EDA, visualization, and interpreting regression results

- Realized the importance of cross-validation and residual analysis for robust models