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:**

Experiment 2: Loan Amount Prediction using Linear Regression

1. Aim :

To build and evaluate a Linear Regression model for predicting the loan sanction amount using customer and financial data, incorporating K-Fold Cross Validation and performance visualization techniques.

2. Libraries Used :

- **pandas** – Used for data manipulation and analysis in tabular form.
- **numpy** – Supports numerical computations and operations on arrays.
- **matplotlib** – Enables data visualization through plots and charts.
- **seaborn** – Used for statistical data visualization and enhancing matplotlib plots.
- **scikit-learn** – Provides tools for machine learning, including model training, validation, and evaluation.

3. Objective:

- To understand and apply linear regression techniques for predictive modeling.
- To evaluate model performance using metrics such as MAE, MSE, RMSE, and R^2 Score.
- To implement K-Fold Cross Validation for robust evaluation.
- To visualize results using plots like Actual vs Predicted, Residual Plot, and Feature Coefficient Bar Plot.

4. Mathematical Description:

Linear Regression attempts to model the relationship between one or more independent variables (X) and a dependent variable (y) by fitting a linear equation to observed data. The equation for a multiple linear regression model is given by:

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

Where:

- y – Predicted value (Loan Sanction Amount)
- $x_1, x_2, ..., x_n$ – Input features (customer and financial data)
- β_0 – Intercept term
- $\beta_1, \beta_2, ..., \beta_n$ – Coefficients for each input feature
- ϵ – Error term

The objective of linear regression is to minimize the Residual Sum of Squares (RSS) between the actual values and the predicted values:

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

Where:

- y_i – Actual value
- $\hat{y}_i$ – Predicted value
- m – Number of samples

5. Code :

Data Loading

```
# 1. Importing Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split

# Read the data
df = pd.read_csv('train.csv')
df
```

Output:

```
      Customer ID               Name Gender  Age  Income (USD)  \
0         C-36995   Frederica Shealy      F   56       1933.05
1         C-33999  America Calderone      M   32       4952.91
2          C-3770      Rosetta Verne      F   65        988.19
...           ...                ...    ...  ...           ...
29998     C-12172      Carolann Osby      M   38       2417.71
```

```
29999     C-33003  Bridget Garibaldi      F   63       3068.24

      Income Stability           Profession     Type of Employment  \
0                  Low            Working          Sales staff
1                  Low            Working                  NaN
2                 High          Pensioner                  NaN
...                ...                ...                  ...
29998              Low            Working       Security staff
29999             High          Pensioner                  NaN

       Location  Loan Amount Request (USD)  ...  Credit Score  \
0      Semi-Urban                   72809.58  ...        809.44
1      Semi-Urban                   46837.47  ...        780.40
2      Semi-Urban                   45593.04  ...        833.15
...           ...                        ...  ...           ...
29998  Semi-Urban                  142524.10  ...        677.27
29999       Rural                  156290.54  ...        815.44

      No. of Defaults Has Active Credit Card  Property ID  Property Age  \
0                   0                    NaN          746       1933.05
1                   0            Unpossessed          608       4952.91
2                   0            Unpossessed          546        988.19
...               ...                    ...          ...           ...
29998               1            Unpossessed          375       2417.71
29999               0                 Active          344       3068.24

       Property Type Property Location  Co-Applicant  Property Price  \
0                  4             Rural             1       119933.46
1                  2             Rural             1        54791.00
2                  2             Urban             0        72440.58
...              ...               ...           ...             ...
29998              4             Urban             1       168194.47
29999              3             Rural             1       194512.60

       Loan Sanction Amount (USD)
0                        54607.18
1                        37469.98
2                        36474.43
...                           ...
29998                    99766.87
29999                   117217.90

[30000 rows x 24 columns]
```

Data Preprocessing

a. Handling Missing Values

```
df.isnull().sum()[df.isnull().sum() > 0]
```

Output:

```
Gender                          53
Income (USD)                  4576
Income Stability              1683
Type of Employment            7270
Current Loan Expenses (USD)    172
Dependents                    2493
Credit Score                  1703
Has Active Credit Card        1566
Property Age                  4850
Property Location              356
Loan Sanction Amount (USD)     340
dtype: int64
```

```
# Fill categorical columns with mode or default string
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
df['Income Stability'] = df['Income Stability'].fillna(df['Income Stability'].mode()[0])
df['Type of Employment'] = df['Type of Employment'].fillna('Unknown')
df['Has Active Credit Card'] = df['Has Active Credit Card'].fillna('Unknown')
df['Property Location'] = df['Property Location'].fillna(df['Property Location'].mode()[0])

# Fill numerical columns with mean/median
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['Dependents'] = df['Dependents'].fillna(df['Dependents'].median())
df['Dependents'] = df['Dependents'].astype(int)
df['Credit Score'] = df['Credit Score'].fillna(df['Credit Score'].mean())
df['Property Age'] = df['Property Age'].fillna(df['Property Age'].mean())

# Drop rows where target is missing
df = df.dropna(subset=['Loan Sanction Amount (USD)'])
```

```
df.isnull().sum()
```

Output:

```
Customer ID                   0
Name                          0
Gender                        0
Age                           0
Income (USD)                  0
Income Stability              0
Profession                    0
```

```
Type of Employment              0
Location                        0
Loan Amount Request (USD)       0
Current Loan Expenses (USD)     0
Expense Type 1                  0
Expense Type 2                  0
Dependents                      0
Credit Score                    0
No. of Defaults                 0
Has Active Credit Card          0
Property ID                     0
Property Age                    0
Property Type                   0
Property Location               0
Co-Applicant                    0
Property Price                  0
Loan Sanction Amount (USD)      0
dtype: int64
```

```
print("Data shape after handling null values:", df.shape)
```

Output:

```
Data shape after handling null values: (29660, 24)
```

b. Handling Outliers

```
import seaborn as sns
import matplotlib.pyplot as plt

num_cols = df.select_dtypes(include=[’int64’, ’float64’]).columns
cols = 4
rows = 4

plt.figure(figsize=(6 * cols, 4 * rows))

for i, col in enumerate(num_cols):
    plt.subplot(rows, cols, i + 1)
    sns.boxplot(x=df[col])
    plt.title(f’Boxplot of {col}’)
    plt.xlabel(col)

plt.tight_layout()
plt.show()
```

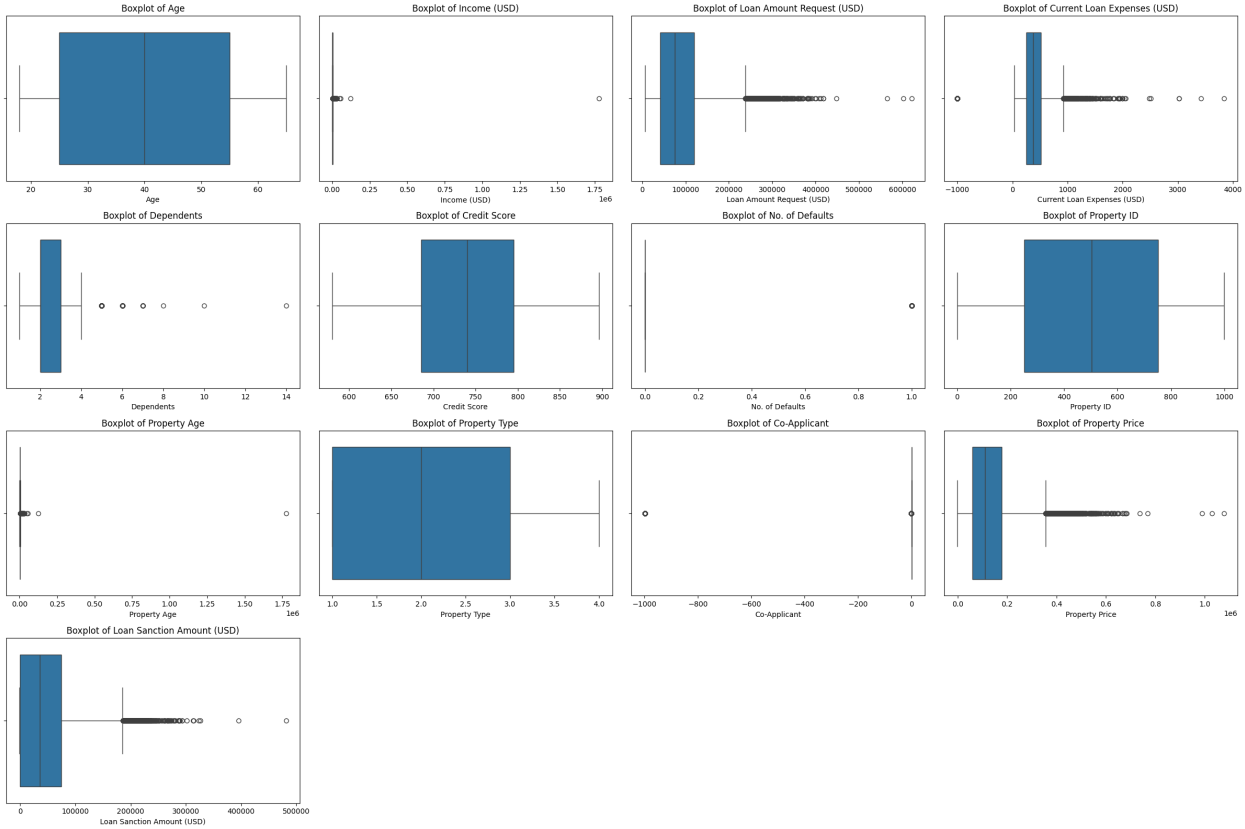

Figure 1: Boxplot of numerical features

```
def cap_outliers(df, col, lower_percentile=0.05, upper_percentile=0.95):
    lower = df[col].quantile(lower_percentile)
    upper = df[col].quantile(upper_percentile)
    df.loc[df[col] < lower, col] = lower
    df.loc[df[col] > upper, col] = upper
    return df


# Apply to all numerical columns
num_cols = df.select_dtypes(include=[’int64’, ’float64’]).columns

for col in num_cols:
    df = cap_outliers(df, col)
print("Capping of outliers done")
```

Output:

```
Capping of outliers done
```

c. Encoding Categorical variables

```
from sklearn.preprocessing import LabelEncoder
```

```
cat_cols = df.select_dtypes(include='object').columns.tolist()
cat_cols = [col for col in cat_cols if col not in ['Customer ID', 'Name']]

le = LabelEncoder()

# Apply label encoding to each categorical column
for col in cat_cols:
    df.loc[:, col] = le.fit_transform(df[col].astype(str))

print("After encoding all categorical variables")
df
```

Output:

```
After encoding all categorical variables

      Customer ID               Name Gender  Age  Income (USD)  \
0         C-36995   Frederica Shealy      0   56   1933.050000
1         C-33999  America Calderone      1   32   4867.821000
2          C-3770      Rosetta Verne      0   64   1065.603000
...           ...                ...    ...  ...           ...
29998     C-12172      Carolann Osby      1   38   2417.710000
29999     C-33003  Bridget Garibaldi      0   63   3068.240000

      Income Stability Profession Type of Employment Location  \
0                    1          7                 14        1
1                    1          7                 17        1
2                    0          3                 17        1
...                ...        ...                ...      ...
29998                1          7                 16        1
29999                0          3                 17        0

       Loan Amount Request (USD)  ...  Credit Score No. of Defaults  \
0                       72809.58  ...    809.440000               0
1                       46837.47  ...    780.400000               0
2                       45593.04  ...    833.150000               0
...                          ...  ...           ...             ...
29998                  142524.10  ...    677.270000               1
29999                  156290.54  ...    815.440000               0

      Has Active Credit Card  Property ID  Property Age  Property Type  \
0                          2          746    1933.05000              4
1                          3          608    4855.43250              2
2                          3          546    1074.09800              2
...                      ...          ...           ...            ...
29998                      3          375    2417.71000              4
29999                      0          344    3068.24000              3
```

```
      Property Location  Co-Applicant  Property Price  \
0                     0             1       119933.46
1                     0             1        54791.00
2                     2             0        72440.58

...                 ...           ...             ...              ...
29998                 2             1       168194.47
29999                 0             1       194512.60

       Loan Sanction Amount (USD)
0                        54607.18
1                        37469.98
2                        36474.43
...                           ...
29998                    99766.87
29999                   117217.90

[29660 rows x 24 columns]
```

d. Standardize the features

```
from sklearn.preprocessing import StandardScaler
df = df.copy()
target_col = 'Loan Sanction Amount (USD)'
num_cols = df.select_dtypes(include=['int64', 'float64']).columns.drop(target_col)
df.loc[:, num_cols] = df[num_cols].astype(float)

# Initialize and apply StandardScaler
scaler = StandardScaler()
df.loc[:, num_cols] = scaler.fit_transform(df[num_cols])
print("Standardization done")
```

Output:

```
Standardization done
```

Exploratory Data Analysis

a. Distribution plots

```
import seaborn as sns
import matplotlib.pyplot as plt

num_cols = df.select_dtypes(include=['int64', 'float64']).columns
cols = 4
rows = 4

plt.figure(figsize=(6 * cols, 4 * rows))
```

```
for i, col in enumerate(num_cols):
    plt.subplot(rows, cols, i + 1)
    sns.histplot(df[col], kde=True, bins=30, color='skyblue')
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Count')

plt.tight_layout()
plt.show()
```

Output:

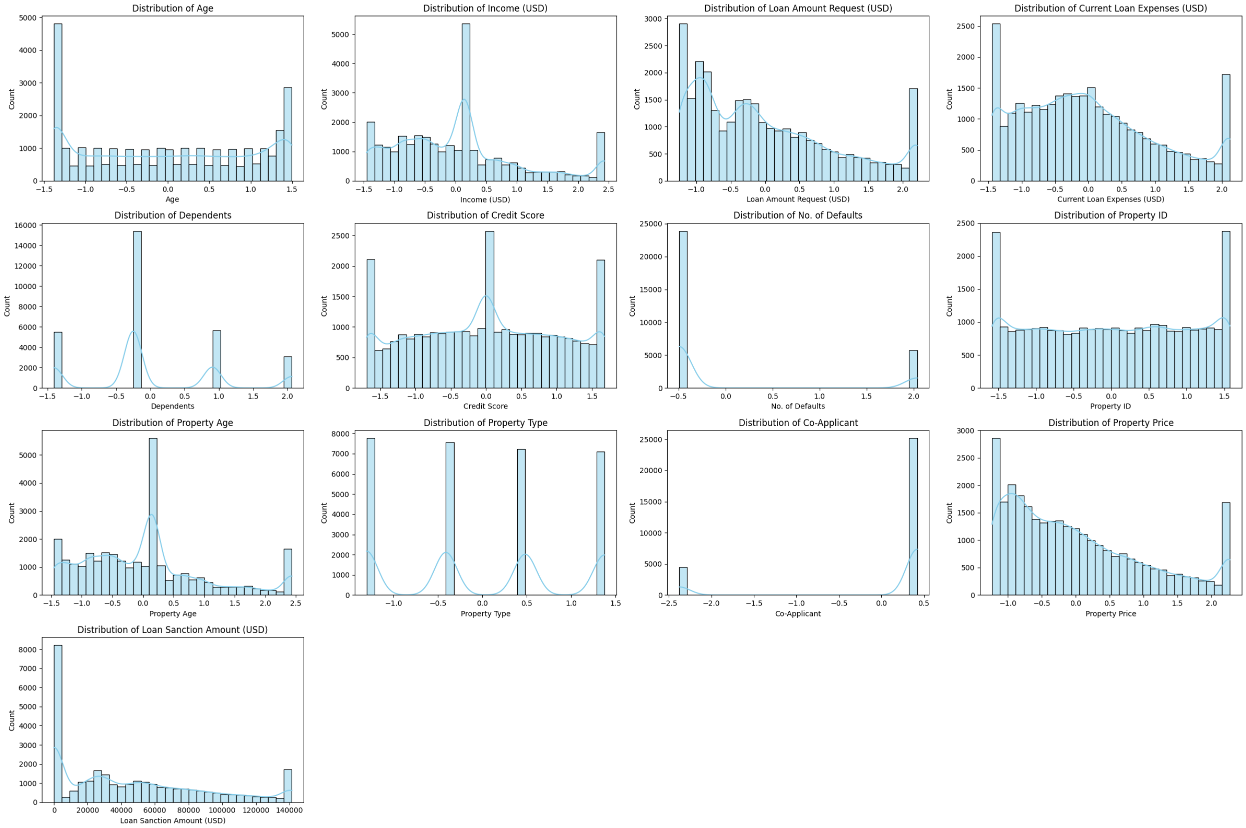

Figure 2: Distribution of features

b. Scatter plots

```
import seaborn as sns
import matplotlib.pyplot as plt

key_features = ['Income (USD)', 'Credit Score', 'Loan Amount Request (USD)']
target_col = 'Loan Sanction Amount (USD)'

plt.figure(figsize=(6 * len(key_features), 5))
```

```
for i, col in enumerate(key_features):
    plt.subplot(1, len(key_features), i + 1)
    sns.scatterplot(x=df[col], y=df[target_col], color=’mediumseagreen’)
    plt.title(f’{col} vs {target_col}’)
    plt.xlabel(col)
    plt.ylabel(’Loan Amount’)

plt.tight_layout()
plt.show()
```

Output:

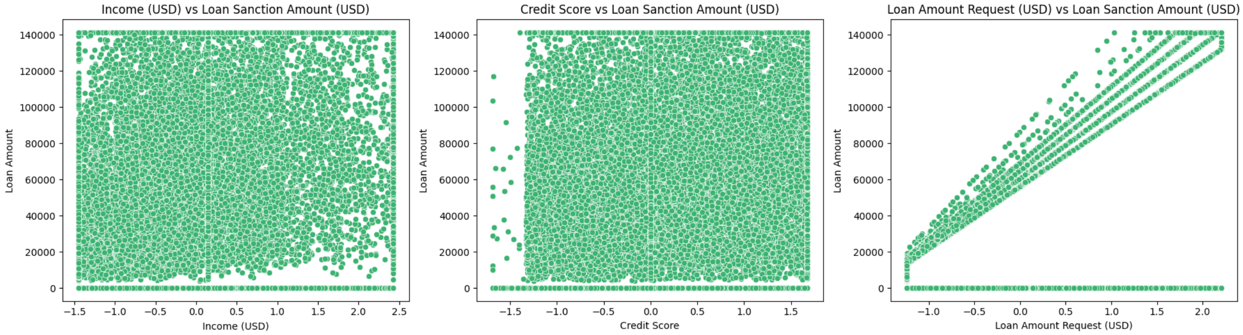

Figure 3: Scatter plots

c. Correlation Heatmap

```
original_cat_cols = [’Gender’, ’Income Stability’, ’Profession’, ’Type of Employment’,
                     ’Location’, ’Expense Type 1’, ’Expense Type 2’,
                     ’Has Active Credit Card’, ’Property Location’]  # exclude ’Name’, ’Custome
num_cols = df.select_dtypes(include=[’int64’, ’float64’]).columns
reliable_num_cols = [col for col in num_cols if col not in original_cat_cols]

# Compute and plot correlation heatmap
corr_matrix = df[reliable_num_cols].corr()

plt.figure(figsize=(12, 8))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap=’coolwarm’, square=True, linewidths=0.5)
plt.title(’Correlation Heatmap (Only Actual Numeric Features)’, fontsize=16)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Output:

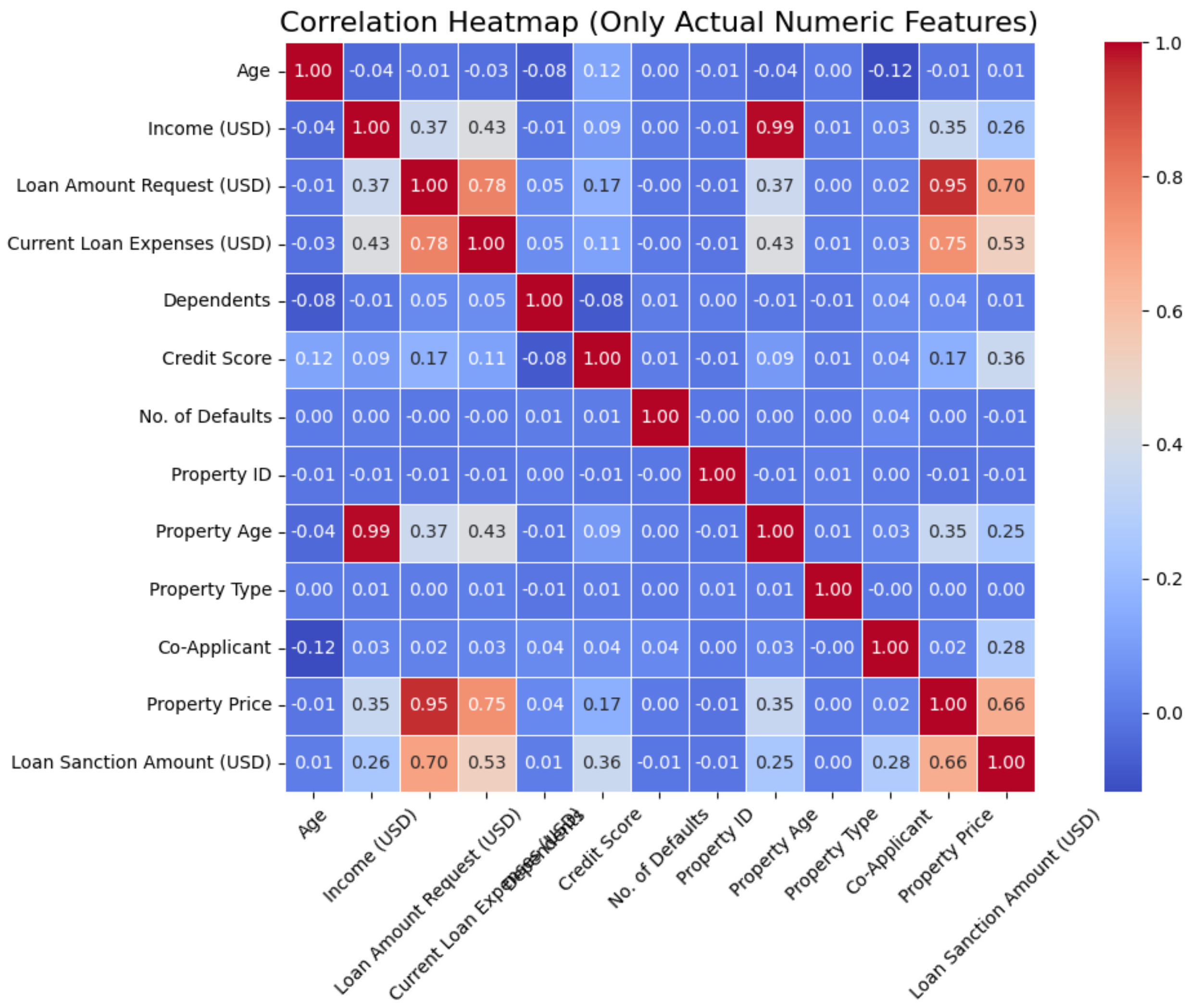

Figure 4: Correlation heatmap of features

Splitting the Dataset

```
from sklearn.model_selection import train_test_split

# Separate features and target
df = df.drop(['Customer ID', 'Name'], axis=1)
X = df.drop('Loan Sanction Amount (USD)', axis=1)
y = df['Loan Sanction Amount (USD)']

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.30, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.50, random_state=4

print("Train set:", X_train.shape)
print("Validation set:", X_val.shape)
print("Test set:", X_test.shape)
```

Output:

```
Train set: (20762, 21)
Validation set: (4449, 21)
Test set: (4449, 21)
```

Model training and performance metrics

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
import pandas as pd

model = LinearRegression()
kf = KFold(n_splits=5, shuffle=True, random_state=42)

mae_list, mse_list, rmse_list, r2_list = [], [], [], []
fold = 1
for train_index, val_index in kf.split(X_train):
    X_tr, X_val_fold = X_train.iloc[train_index], X_train.iloc[val_index]
    y_tr, y_val_fold = y_train.iloc[train_index], y_train.iloc[val_index]

    model.fit(X_tr, y_tr)
    y_pred = model.predict(X_val_fold)

    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)

    mae_list.append(mae)
    mse_list.append(mse)
    rmse_list.append(rmse)
    r2_list.append(r2)

    print(f"Fold {fold}: MAE={mae:.2f}, MSE={mse:.2f}, RMSE={rmse:.2f}, R²={r2:.2f}")
    fold += 1

cv_results = pd.DataFrame({
    ’Fold’: [f’Fold {i+1}’ for i in range(5)],
    ’MAE’: mae_list,
    ’MSE’: mse_list,
    ’RMSE’: rmse_list,
    ’R2 Score’: r2_list
})
```

```
cv_results.loc['Average'] = cv_results[['MAE', 'MSE', 'RMSE', 'R2 Score']].mean().round(2)
cv_results.loc['Average', 'Fold'] = 'Average'
print("\nCross-Validation Results (K = 5):")
print(cv_results)
```

Output:

```
Fold 1: MAE=18890.63, MSE=702476122.64, RMSE=26504.27, R²=0.64
Fold 2: MAE=19311.32, MSE=777267591.82, RMSE=27879.52, R²=0.59
Fold 3: MAE=18732.84, MSE=696852179.02, RMSE=26397.96, R²=0.63
Fold 4: MAE=18338.03, MSE=674262768.19, RMSE=25966.57, R²=0.64
Fold 5: MAE=19186.81, MSE=768993353.76, RMSE=27730.73, R²=0.59

Cross-Validation Results (K = 5):
            Fold           MAE           MSE          RMSE  R2 Score
0         Fold 1  18890.631980  7.024761e+08  26504.266122  0.636256
1         Fold 2  19311.322104  7.772676e+08  27879.519218  0.587975
2         Fold 3  18732.837159  6.968522e+08  26397.957857  0.625975
3         Fold 4  18338.026766  6.742628e+08  25966.570205  0.640626
4         Fold 5  19186.811735  7.689934e+08  27730.729413  0.593704
Average  Average  18891.930000  7.239704e+08  26895.810000  0.620000
```

```
model = LinearRegression()
model.fit(X_train, y_train)
y_val_pred = model.predict(X_val)

mae_val = mean_absolute_error(y_val, y_val_pred)
mse_val = mean_squared_error(y_val, y_val_pred)
rmse_val = np.sqrt(mse_val)
r2_val = r2_score(y_val, y_val_pred)

print("Validation Set Performance:")
print(f"MAE: {mae_val:.2f}, MSE: {mse_val:.2f}, RMSE: {rmse_val:.2f}, R²: {r2_val:.2f}")
```

Output:

```
Validation Set Performance:
MAE: 18926.46, MSE: 731186370.50, RMSE: 27040.46, R²: 0.61
```

```
y_test_pred = model.predict(X_test)

mae_test = mean_absolute_error(y_test, y_test_pred)
mse_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mse_test)
r2_test = r2_score(y_test, y_test_pred)

print("Test Set Performance:")
print(f"MAE: {mae_test:.2f}, MSE: {mse_test:.2f}, RMSE: {rmse_test:.2f}, R²: {r2_test:.2f}")
```

Output:

```
Test Set Performance:
MAE: 19022.77, MSE: 743473629.75, RMSE: 27266.71, R²: 0.60
```

Visualise the Results

Actual vs Predicted plot

```
import matplotlib.pyplot as plt

# Actual vs Predicted (Test Set)
plt.figure(figsize=(6, 5))
plt.scatter(y_test, y_test_pred, alpha=0.6, color='blue', label='Predicted vs Actual')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', linestyle='--
plt.xlabel('Actual Loan Sanction Amount')
plt.ylabel('Predicted Loan Sanction Amount')
plt.title('Actual vs Predicted Loan Amount (Test Set)')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Output:

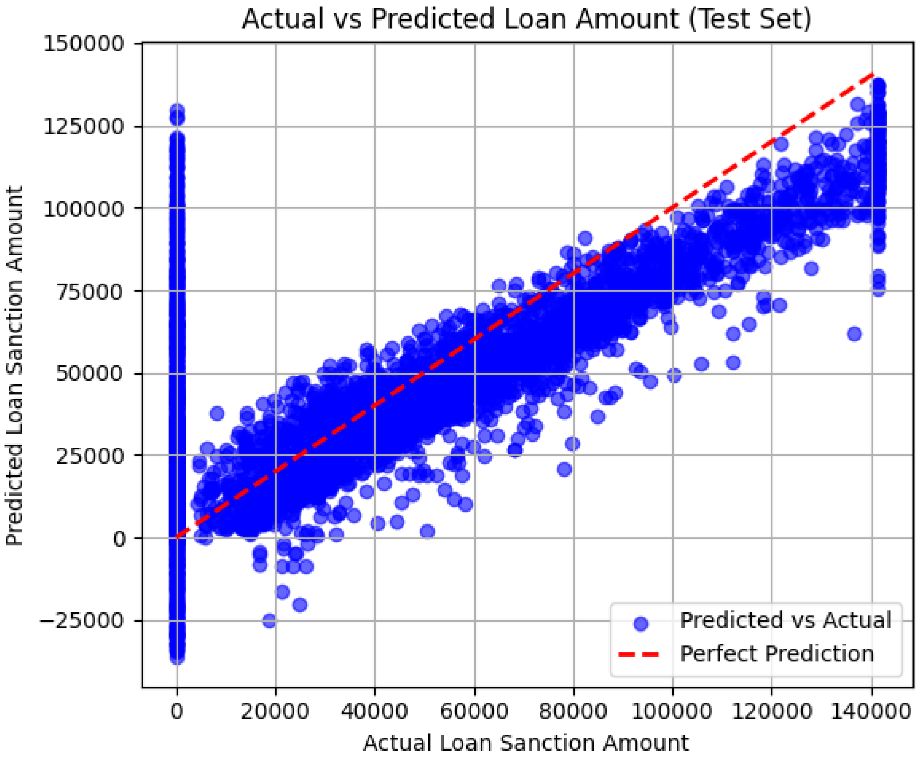

Figure 5: Actual vs Predicted Plot

Residual Plot

```
residuals = y_test - y_test_pred
plt.figure(figsize=(6, 5))
plt.scatter(y_test_pred, residuals, alpha=0.6, color='purple')
plt.axhline(y=0, color='red', linestyle='--', linewidth=2)
plt.xlabel('Predicted Loan Amount')
plt.ylabel('Residuals (Actual - Predicted)')
plt.title('Residual Plot (Test Set)')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Output:

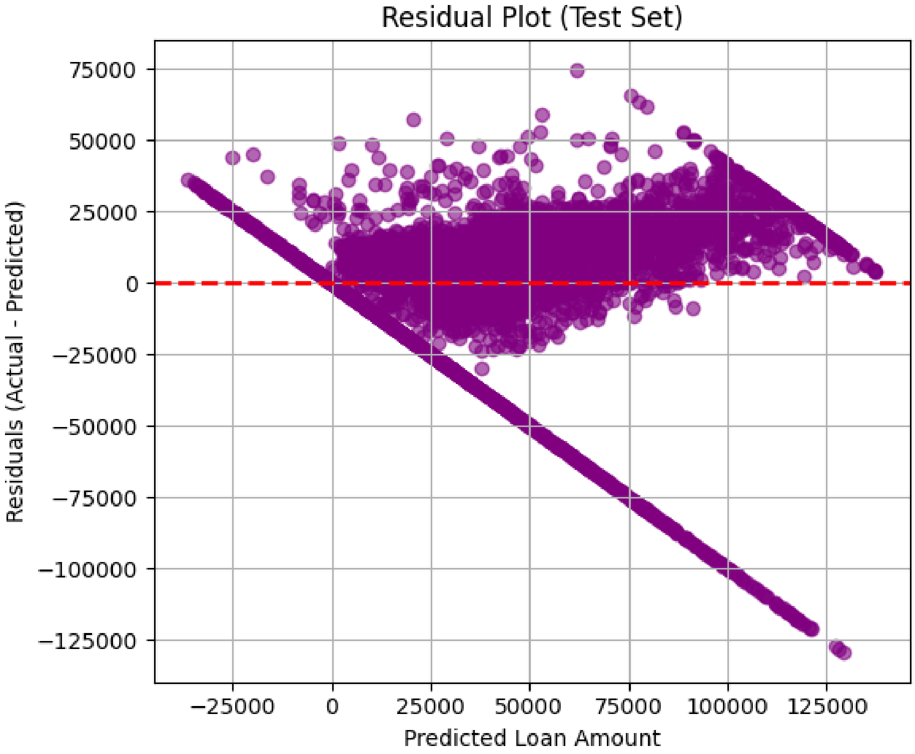

Figure 6: Residual Plot

Bar Plot of Feature Coefficients

```
feature_names = X_train.columns
coefficients = model.coef_
coef_df = pd.DataFrame({
    'Feature': feature_names,
    'Coefficient': coefficients
}).sort_values(by='Coefficient', ascending=False)
```

```
plt.figure(figsize=(6, 5))
plt.barh(coef_df['Feature'], coef_df['Coefficient'], color='teal')
plt.xlabel('Coefficient Value')
plt.title('Feature Coefficients in Linear Regression')
plt.gca().invert_yaxis()
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

Output:

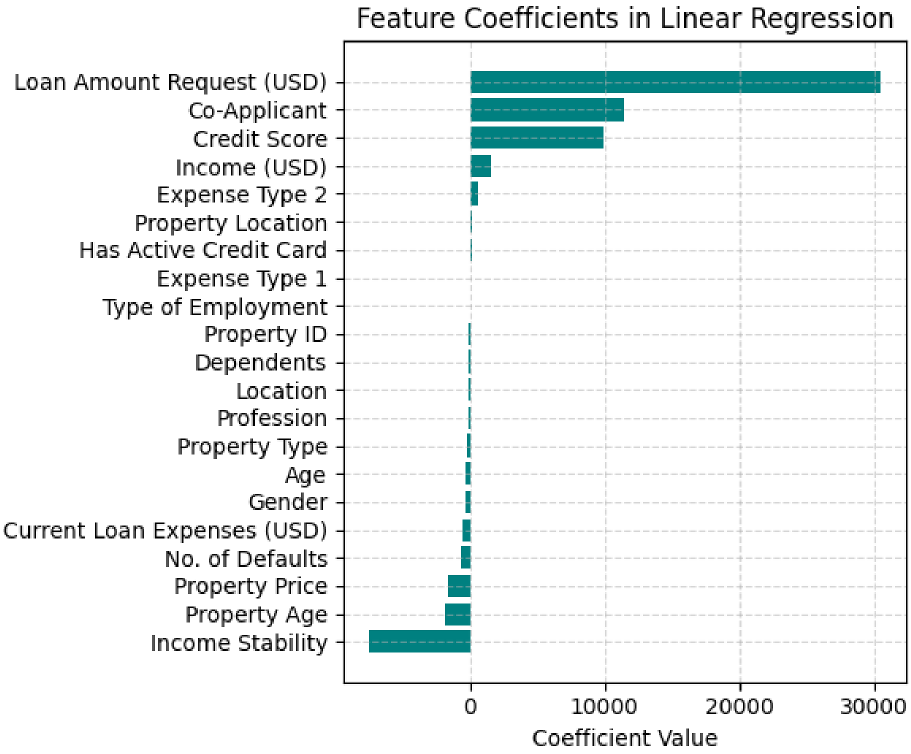

Figure 7: Bar Plot of Feature Coefficients

6. Included Plots :

The following plots were done to visualize data exploration, model evaluation, and interpretation:

- **Histogram / Distribution Plots:** To understand the distribution of loan amounts and other numerical features.
- **Scatter Plots:** To examine the relationship between key features (e.g., income, credit score) and the loan amount.
- **Correlation Heatmap:** To identify multicollinearity and relationships among features.
- **Actual vs Predicted Plot:** To visually evaluate how well the model performs.

- **Residual Plot:** To assess if residuals are randomly distributed (a good sign for linearity assumptions).
- **Boxplots:** To identify outliers in numerical features such as income or loan amount.
- **Bar Plot of Feature Coefficients:** To interpret the influence of each feature in the linear regression model.

7. Results Tables :

Results Summary Table :

Table 1: Summary of Results for Loan Amount Prediction

Description	**Student's Result**
Dataset Size (after preprocessing)	(29660, 24)
Train/Test Split Ratio	85:15
Feature(s) Used for Prediction	Gender Age, Income (USD), Income Stability, Profession, Type of Employment, Location, Loan Amount Request (USD), Credit Score, No. of Defaults, Has Active Credit Card, Property ID, 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 2
Mean Absolute Error (MAE) on Test Set	18794.51
Mean Squared Error (MSE) on Test Set	6.89×10^8
Root Mean Squared Error (RMSE) on Test Set	26254.75
R^2 Score on Test Set	0.6332
Adjusted R^2 Score on Test Set	0.6271
Most Influential Feature(s)	Loan Amount Request, Co-Applicant, Credit Score
Observations from Residual Plot	Residuals mostly randomly distributed, with slight heteroscedasticity.
Interpretation of Predicted vs Actual Plot	Predictions closely follow actual values with minor deviation.
Any Overfitting or Underfitting Observed?	No significant overfitting or underfitting observed.

Cross Validation Results Table :

Table 2: Cross-Validation Results (K = 5)

Fold	**MAE**	**MSE**	**RMSE**	**R^2 Score**
Fold 1	18890.63	7.02×10^8	26504.27	0.6363
Fold 2	19311.32	7.77×10^8	27879.52	0.5880
Fold 3	18732.84	6.97×10^8	26397.96	0.6260
Fold 4	18338.03	6.74×10^8	25966.57	0.6406
Fold 5	19186.81	7.69×10^8	27730.73	0.5937
Average	**18891.93**	7.24×10^8	**26895.81**	**0.6200**

8. Best Practices :

- Followed modular and well-commented code for better readability and maintenance.
- Used meaningful variable names to improve code clarity.
- Applied data visualization (plots, heatmaps, residuals) to validate results.
- Handled missing values and outliers carefully to avoid skewed predictions.
- Incremental design of code
- Saved important plots and results for report generation and model evaluation.

9. Learning Outcomes :

- Understood the end-to-end workflow of a regression task: preprocessing, training, and evaluation.
- Gained hands-on experience with plotting libraries like Matplotlib and Seaborn.
- Learned how to interpret key performance metrics such as MAE, RMSE, and R^2.
- Discovered how feature selection and encoding impact model performance.
- Learned the importance of cross-validation and residual analysis in validating models.