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: Title 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 R^2 score with appropriate visualizations.

Libraries Used:

- pandas
- numpy
- matplotlib.pyplot
- seaborn
- scikit-learn

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 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.

article [utf8]inputenc geometry listings xcolor
margin=1in
codegraygray0.95 darkbluergb0.1,0.1,0.6
mystyle backgroundcolor=codegray, commentstyle=gray, keywordstyle=blue, numberstyle=gray, stringstyle=darkblue, basicstyle=, breakatwhitespace=false, breaklines=true, captionpos=b, keepspaces=true, numbers=left, numbersep=5pt, showspaces=false, showstringspaces=false, showtabs=false, tabsize=2
style=mystyle

Python Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

df = pd.read_csv(r"C:/Users/madhu/Downloads/archive/train.csv")
df.head()
df.describe()
df.isnull().sum()

df["Gender"]=df["Gender"].fillna(df["Gender"].mode()[0])
df["Income (USD)"]=df["Income (USD)"].fillna(df["Income (USD)"].mean())
df["Income Stability"]=df["Income Stability"].fillna(df["Income Stability"].mode()[0])
df["Type of Employment"]=df["Type of Employment"].fillna("unknown")
df["Current Loan Expenses (USD)"]=df["Current Loan Expenses (USD)"].fillna(df["Income (USD)"].m
df["Dependents"]=df["Dependents"].fillna(df["Dependents"].mean())
df["Credit Score"]=df["Credit Score"].fillna(df["Credit Score"].mean())
df["Has Active Credit Card"]=df["Has Active Credit Card"].fillna(df["Has Active Credit Card"].m
df["Property Age"]=df["Property Age"].fillna(df["Property Age"].mean())
```

```
df["Property Location"]=df["Property Location"].fillna(df["Property Location"].mode()[0])
df.dropna(subset=["Loan Sanction Amount (USD)"], inplace=True)

df.drop(columns=["Customer ID", "Name", "Property ID","Type of Employment","Profession"], inpla

df["Gender"] = df["Gender"].map({"M": 1, "F": 0})
df["Expense Type 1"] = df["Expense Type 1"].map({"Y": 1, "N": 0})
df["Expense Type 2"] = df["Expense Type 2"].map({"Y": 1, "N": 0})
df["Has Active Credit Card"] = df["Has Active Credit Card"].map({
    "Active": 1,
    "Inactive": 0,
    "Unpossessed": -1
})
df["Income Stability"] = df["Income Stability"].map({"Low": 0, "High": 1})
df["Property Location"] = df["Property Location"].map({"Rural": 0, "Semi-Urban": 1, "Urban": 2
df["Location"] = df["Location"].map({"Rural": 0, "Semi-Urban": 1, "Urban": 2})

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

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)

from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="mean")
x_train = imputer.fit_transform(x_train)
x_test = imputer.transform(x_test)

model = LinearRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

from sklearn.metrics import mean_squared_error, r2_score
r2_error = r2_score(y_test, y_pred)
print(r2_error)
print(df.columns)

from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler

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

mse_list = []
r2_list = []
rmse = []
```

```
for fold, (train_idx, test_idx) in enumerate(kf.split(x), 1):
    X_train, X_test = x.iloc[train_idx], x.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    rms = np.sqrt(mse)

    mse_list.append(mse)
    r2_list.append(r2)
    rmse.append(rms)

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

print(f"\nAverage MSE: {np.mean(mse_list):.2f}")
print(f"Average R2: {np.mean(r2_list):.2f}")
print(f"Average RMSE: {np.mean(rmse):.2f}")

plt.figure(figsize=(10, 5))
plt.plot(range(1, k+1), mse_list, marker='o', label='MSE per Fold')
plt.plot(range(1, k+1), r2_list, marker='s', label='R² per Fold')
plt.title("K-Fold Validation Performance")
plt.xlabel("Fold")
plt.ylabel("Score")
plt.legend()
plt.grid(True)
plt.show()

import seaborn as sns
sns.set(style="whitegrid")

categorical_cols = ['Gender', 'Income Stability', 'Location', 'Expense Type 1', 'Expense Type 2
                    'Has Active Credit Card', 'Property Type', 'Property Location', 'Co-Applica

for col in categorical_cols:
    plt.figure(figsize=(8, 4))
    sns.countplot(data=df, x=col, palette="Set2")
    plt.title(f"Count Plot of {col}")
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

```
target = 'Loan Sanction Amount (USD)'
numerical_cols = ['Age', 'Income (USD)', 'Loan Amount Request (USD)', 'Current Loan Expenses (U
                  'Dependents', 'Credit Score', 'No. of Defaults', 'Property Age', 'Property Pr

for col in ['Income (USD)', 'Credit Score', 'Property Price', 'Loan Amount Request (USD)']:
    plt.figure(figsize=(6, 4))
    sns.scatterplot(x=col, y=target, data=df)
    plt.title(f"{target} vs {col}")
    plt.tight_layout()
    plt.show()

plt.figure(figsize=(10, 8))
corr = df[numerical_cols + [target]].corr()
sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.tight_layout()
plt.show()

for col in ['Income (USD)', 'Loan Amount Request (USD)', target]:
    plt.figure(figsize=(6, 4))
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot of {col}")
    plt.tight_layout()
    plt.show()

plt.figure(figsize=(6, 4))
sns.scatterplot(x=y_test, y=y_pred)
plt.xlabel("Actual Loan Amount")
plt.ylabel("Predicted Loan Amount")
plt.title("Actual vs Predicted")
plt.plot([y.min(), y.max()], [y.min(), y.max()], color='red', linestyle='--')
plt.tight_layout()
plt.show()

residuals = y_test - y_pred
plt.figure(figsize=(6, 4))
sns.scatterplot(x=y_pred, y=residuals)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.tight_layout()
plt.show()

coefficients = pd.Series(model.coef_, index=x.columns).sort_values()
plt.figure(figsize=(10, 8))
coefficients.plot(kind="bar")
```

```
plt.title("Feature Importance (Linear Coefficients)")
plt.tight_layout()
plt.show()

print("R2 Score:", r2_score(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))

# Final Fold Loop Example (alternate)
kf = KFold(n_splits=5, shuffle=True, random_state=42)
model = LinearRegression()
mse_scores = []
r2_scores = []

for fold, (train_idx, test_idx) in enumerate(kf.split(x), 1):
    X_train, X_test = x.values[train_idx], x.values[test_idx]
    y_train, y_test = y.values[train_idx], y.values[test_idx]

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

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

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

print(f"\nAverage MSE: {np.mean(mse_scores):.2f}")
print(f"Average R²: {np.mean(r2_scores):.2f}")

plt.figure(figsize=(10, 5))
plt.plot(range(1, k+1), mse_scores, marker='o', label='MSE')
plt.plot(range(1, k+1), r2_scores, marker='s', label='R²')
plt.xlabel("Fold")
plt.ylabel("Score")
plt.title("K-Fold Cross Validation Performance")
plt.legend()
plt.grid(True)
plt.show()
```

graphicx float article [margin=1in]geometry booktabs longtable caption array

```
Fold 1: MSE = 1017292280.07, R2 = 0.55, RMSE = 31895.02
Fold 2: MSE = 974737027.27, R2 = 0.57, RMSE = 31220.78
Fold 3: MSE = 1012251604.48, R2 = 0.56, RMSE = 31815.90
Fold 4: MSE = 919543050.51, R2 = 0.62, RMSE = 30323.97
Fold 5: MSE = 994672908.56, R2 = 0.58, RMSE = 31538.44

Average MSE: 983699374.18
Average R2: 0.58
Average RMSE: 31358.82
```

Figure 1: Evaluation Metrics: MSE, R^2, and RMSE across 5 folds

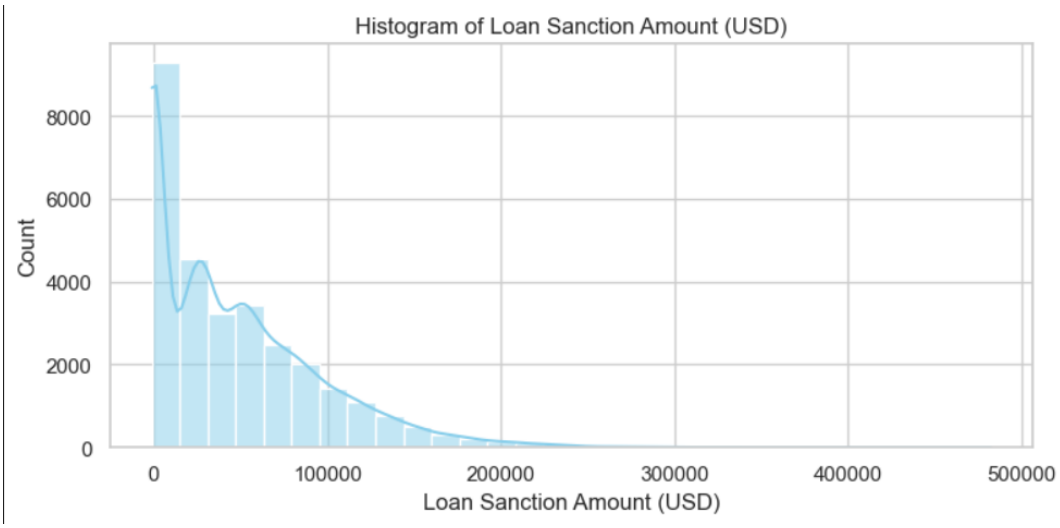

Figure 2: Histogram of Loan Sanction Amount (USD)

Cross-Validation Results Table

If you have used K-Fold Cross-Validation (e.g., with $K = 5$), report the evaluation metrics for each fold in the table below.

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

Fold	MAE	MSE	RMSE	R^2 Score
Fold 1	–	1017292280.07	31895.02	0.55
Fold 2	–	974737027.27	31220.78	0.57
Fold 3	–	1012251604.48	31815.90	0.56
Fold 4	–	919543050.51	30323.97	0.62
Fold 5	–	994672908.56	31538.44	0.58
Average				

Results Summary Table

Students are expected to fill in the following table based on their model's performance and visualizations.

article [margin=1in]geometry enumitem titlesec

8. Best Practices

- **Data Preprocessing:** Handle missing values carefully (drop or impute), and remove irrelevant columns such as IDs and names that do not contribute to 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 (e.g., `StandardScaler`) for numerical features and one-hot 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.

9. Learning Outcomes

Through this experiment, I have:

- Understood the full ML 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.
- Used cross-validation for robust model performance assessment.

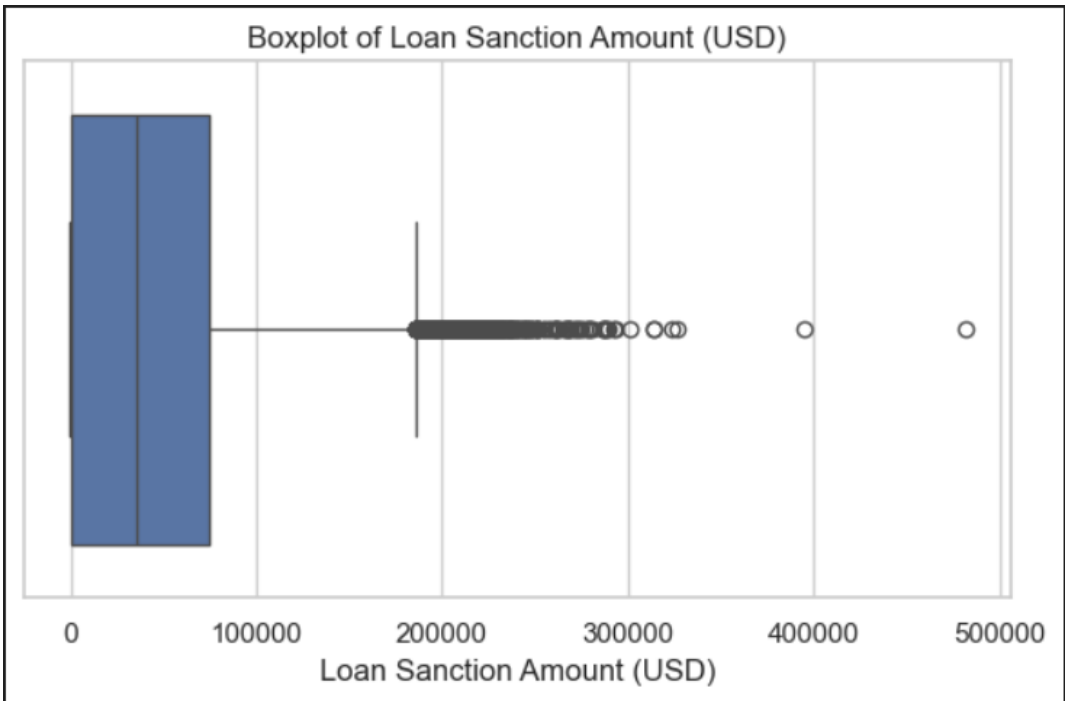

Figure 3: Boxplot of features

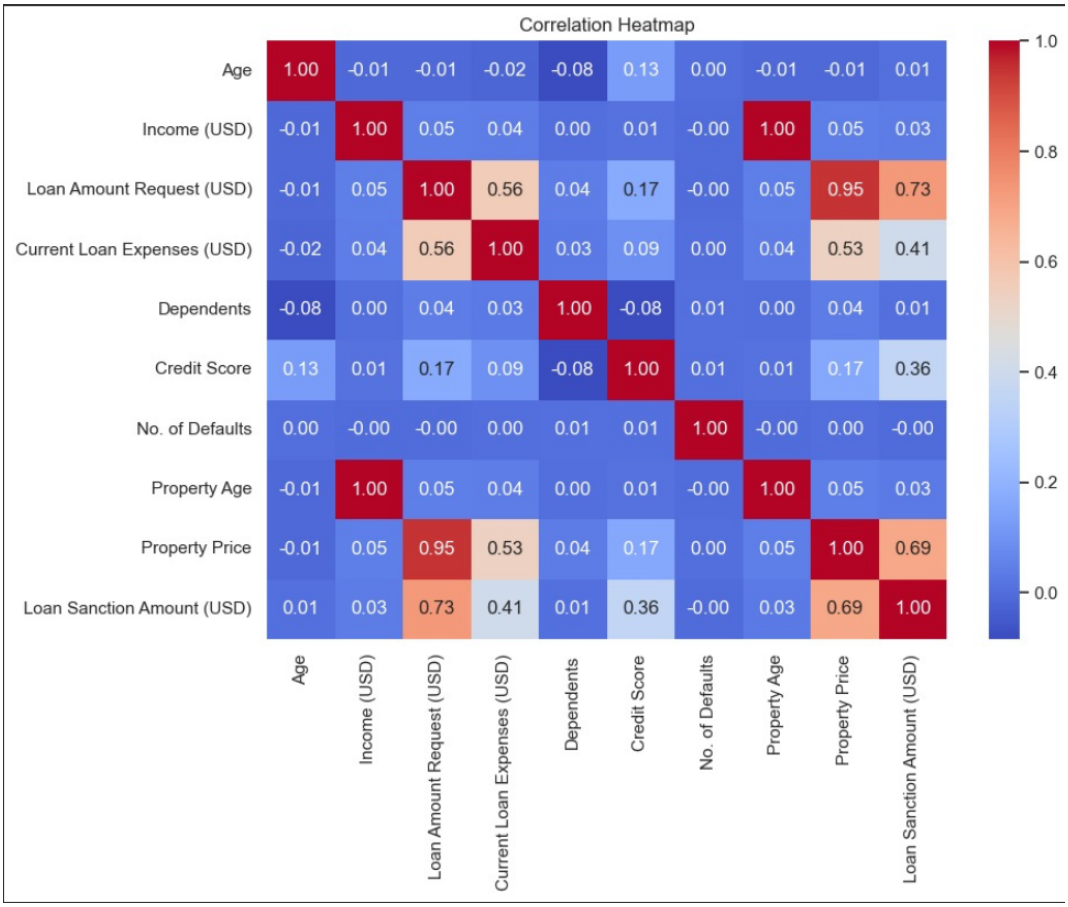

Figure 4: correlation

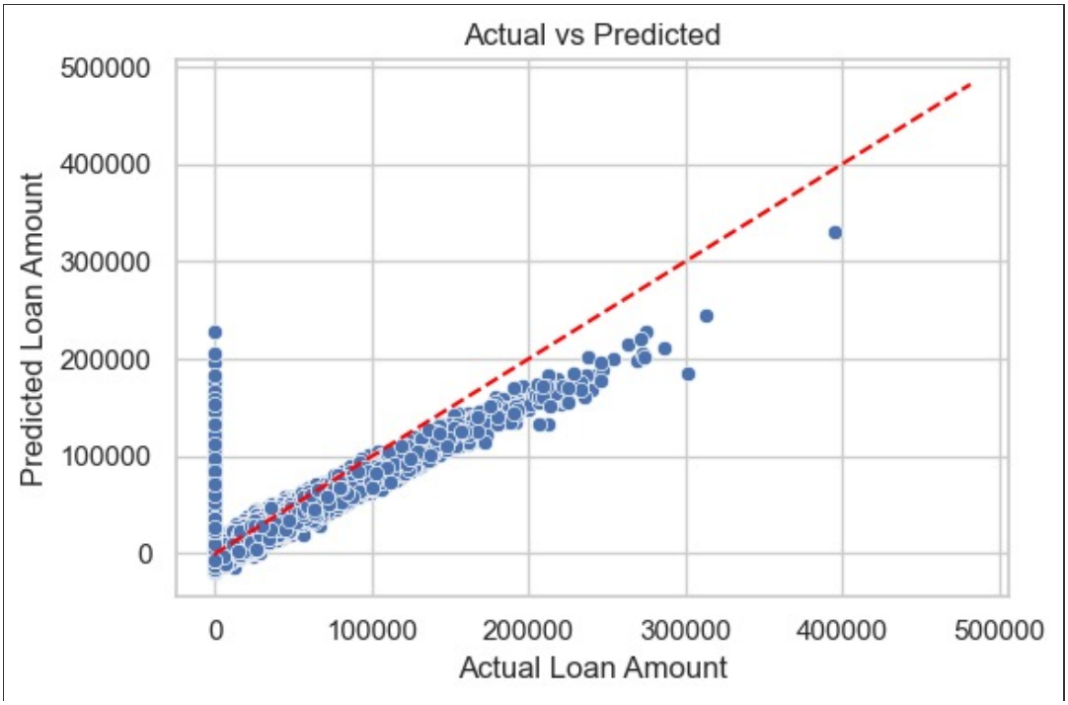

Figure 5: actual vs predicted

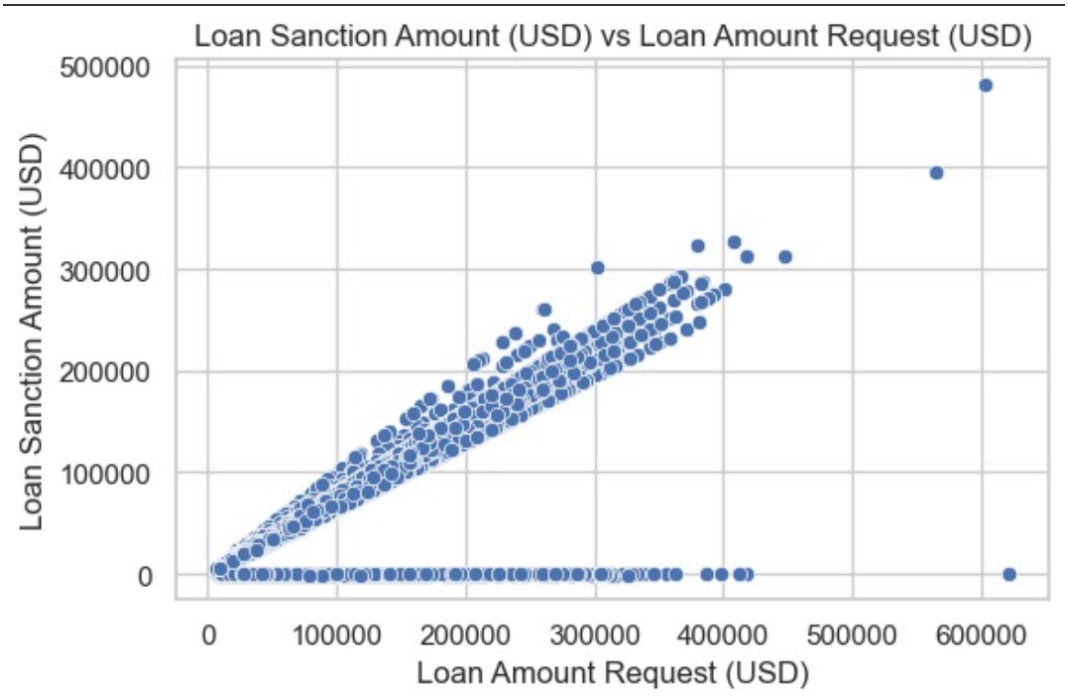

Figure 6: Loan sanction amount vs Loan amount request

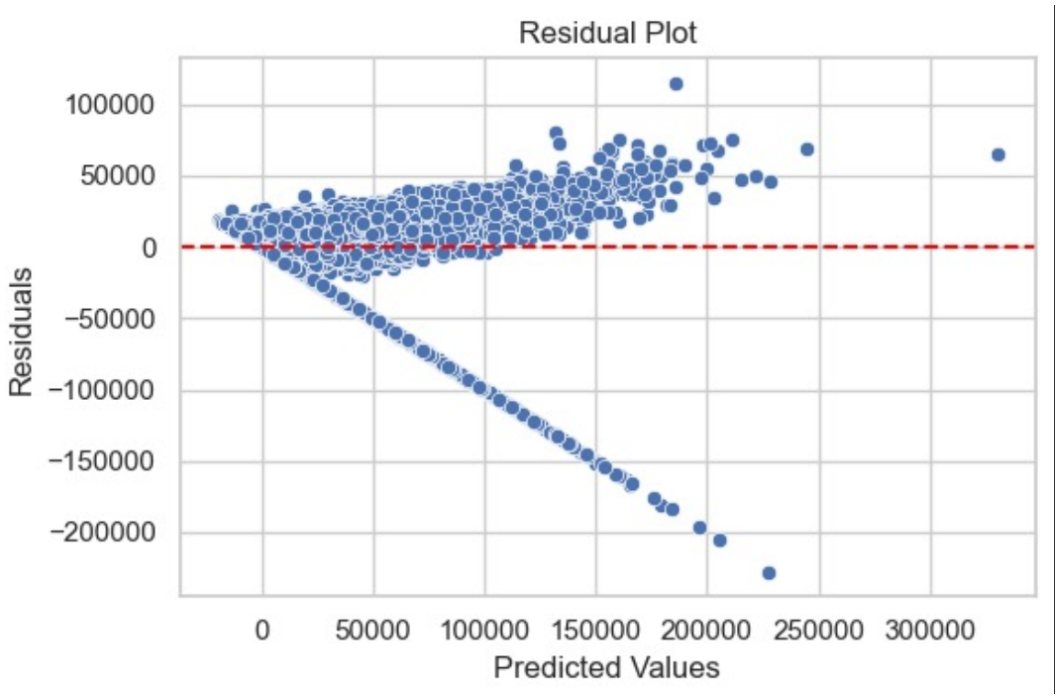

Figure 7: Residual plot

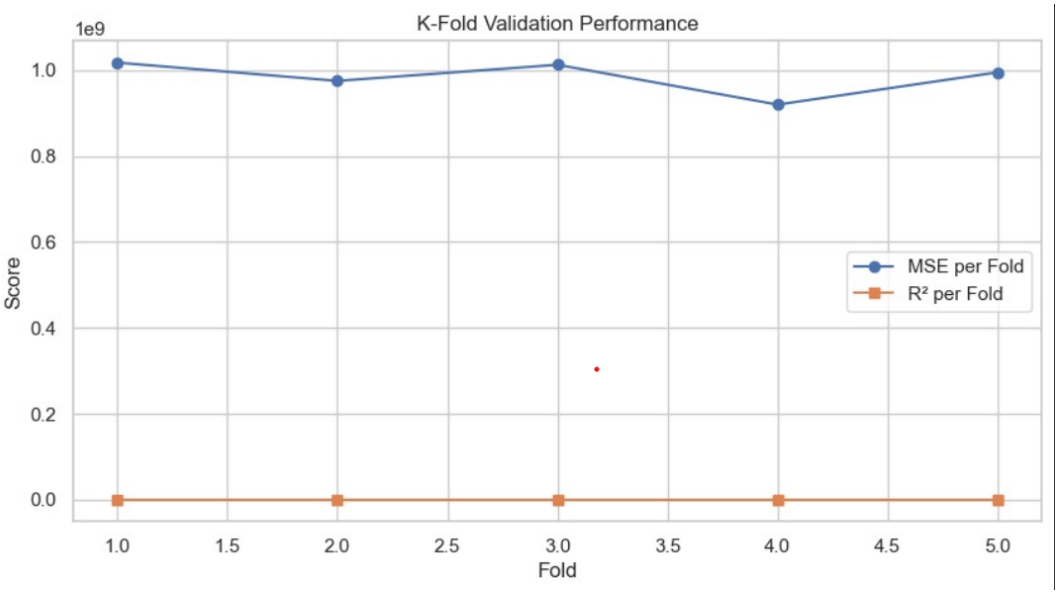

Figure 8: Kfold

Table 2: Summary of Results for Loan Amount Prediction

Description	**Student's Result**
Dataset Size (after preprocessing)	29660
Train/Test Split Ratio	70/30
Feature(s) Used for Prediction	Index(['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)'], dtype='object')
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	
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 0', 'Property Price', 'Credit Score']
Observations from Residual Plot	The residual plot shows a clear pattern with residuals de creasing as predicted values increase, indicating model bias and heteroscedasticity. This suggests the linear model may not fully capture the relationship.
Interpretation of Predicted vs Actual Plot	The residual plot shows a clear pattern with residuals decreasing as predicted values increase, indicating model bias and heteroscedasticity. This suggests the linear model may not fully capture the relationship.
Any Overfitting or Underfitting Observed?	
If Yes, Brief Justification (e.g., training vs test error, residual patterns)	No significant overfitting or underfitting observed. The training and validation errors are comparable, and resid uals do not show extreme patterns, indicating the model generalizes reasonably well. However, some bias at higher values suggests slight underfitting in that range.