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: 12/08/2025**

Experiment 2: Loan Amount Prediction using Multiple Regression Models

1 Aim

To build, evaluate, and compare multiple regression models for predicting loan sanction amounts, justifying the model choices and using K-Fold Cross Validation to identify the most suitable model.

2 Libraries Used

Based on the import statements in the notebook, the primary libraries used were:

- **pandas** – For data manipulation, reading CSV files, and data frame operations.
- **numpy** – For numerical computations, especially for array manipulation and mathematical functions.
- **matplotlib** – For creating static, animated, and interactive visualizations.
- **seaborn** – For making attractive and informative statistical graphics.
- **scikit-learn** – For implementing and evaluating machine learning models.
- **xgboost** – For implementing the high-performance Gradient Boosting algorithm.

3 Objectives

- To apply and compare a wide range of regression algorithms for a predictive modeling task.
- To evaluate model performance using MAE, MSE, RMSE, and R^2 Score.

- To implement K-Fold Cross Validation to ensure model performance is robust and generalizable.
- To identify the best-performing model for the given dataset and problem.

4 Data Preprocessing

```
#Data Loading
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data
df = pd.read_csv('loan_sanction.csv')

# Display the first 5 rows
print("First 5 rows of the dataset:")
print(df.head())
```

	Customer ID	Name	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	Ap
0	C-36995	Frederica Shealy	F	56	1933.05	Low	Working	Sales staff	Semi-Urban	72809.58	...	809.44	0	NaN	746	1933.05	4	Rural	
1	C-33999	America Calderone	M	32	4952.91	Low	Working	NaN	Semi-Urban	46837.47	...	780.40	0	Unpossessed	608	4952.91	2	Rural	
2	C-3770	Rosetta Verne	F	65	988.19	High	Pensioner	NaN	Semi-Urban	45593.04	...	833.15	0	Unpossessed	546	988.19	2	Urban	
3	C-26480	Zoe Chitty	F	65	NaN	High	Pensioner	NaN	Rural	80057.92	...	832.70	1	Unpossessed	890	NaN	2	Semi-Urban	
4	C-23459	Afton Venema	F	31	2614.77	Low	Working	High skill tech staff	Semi-Urban	113858.89	...	745.55	1	Active	715	2614.77	4	Semi-Urban	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	C-43723	Angelyn Clevenger	M	38	4969.41	Low	Commercial associate	Managers	Urban	76657.90	...	869.61	0	Unpossessed	566	4969.41	4	Urban	
29996	C-32511	Silas Slaugh	M	20	1606.88	Low	Working	Laborers	Semi-Urban	66595.14	...	729.41	0	Inactive	175	1606.88	3	Urban	
29997	C-5192	Carmelo Lone	F	49	NaN	Low	Working	Sales staff	Urban	81410.08	...	NaN	0	Active	959	NaN	1	Rural	
29998	C-12172	Carolann Osby	M	38	2417.71	Low	Working	Security staff	Semi-Urban	142524.10	...	677.27	1	Unpossessed	375	2417.71	4	Urban	
29999	C-33003	Bridget Garibaldi	F	63	3068.24	High	Pensioner	NaN	Rural	156290.54	...	815.44	0	Active	344	3068.24	3	Rural	

30000 rows × 24 columns

Figure 1: Output , showing the first five records of the dataset.

Handling Missing Values

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

	0
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

Figure 2: Output , showing the Missing Values of the dataset.

```
# 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()[
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

# 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['Curr
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()
```

	0
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

Figure 3: Output , showing the Result After Handling Missing Values of the dataset.

5 Exploratory Data Analysis

EDA was conducted to understand feature distributions and relationships.

Handling Outliers

```
[language=Python, caption=Capping Outliers]
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()
```

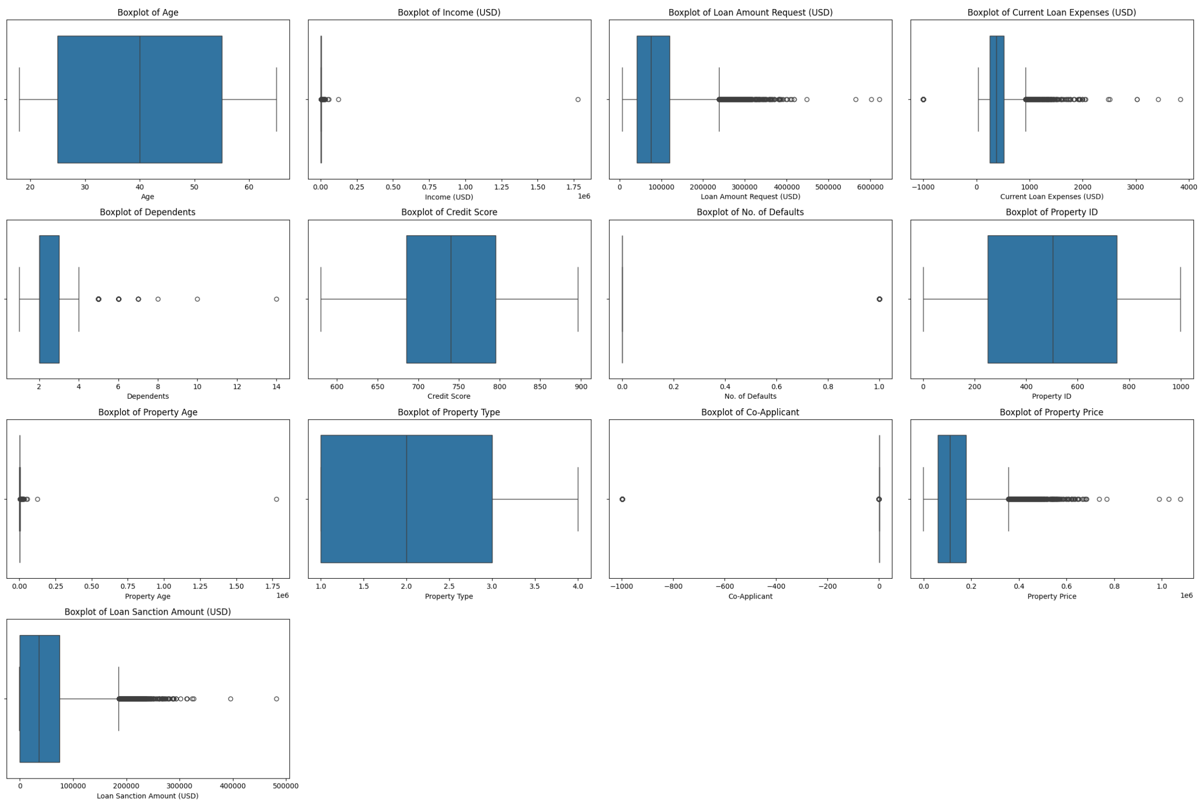

Figure 4: Boxplots of numerical features before outlier capping.

```
[language=Python, caption=Capping Outliers]
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")
\end{lstlisting}
\begin{verbatim}
Capping of outliers done
```

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
```

After encoding all categorical variables

	Customer ID	Name	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
0	C-36995	Frederica Shealy	0	56	1933.050000	1	7	14	1	72809.58	...	809.440000	0	2	746	1933.05000	4	0
1	C-33999	America Calderone	1	32	4867.821000	1	7	17	1	46837.47	...	780.400000	0	3	608	4855.43250	2	0
2	C-3770	Rosetta Verne	0	64	1065.603000	0	3	17	1	45593.04	...	833.150000	0	3	546	1074.09800	2	2
3	C-26480	Zoe Chitty	0	64	2630.574417	0	3	17	0	80057.92	...	832.700000	1	3	890	2631.11944	2	1
4	C-23459	Afton Venema	0	31	2614.770000	1	7	6	1	113858.89	...	745.550000	1	0	715	2614.77000	4	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	C-43723	Angelyn Clevenger	1	38	4867.821000	1	1	10	2	76657.90	...	853.410500	0	3	566	4855.43250	4	2
29996	C-32511	Silas Slaugh	1	20	1606.880000	1	7	8	1	66595.14	...	729.410000	0	1	175	1606.88000	3	2
29997	C-5192	Carmelo Lone	0	49	2630.574417	1	7	14	2	81410.08	...	739.885381	0	0	949	2631.11944	1	0
29998	C-12172	Carolann Osby	1	38	2417.710000	1	7	16	1	142524.10	...	677.270000	1	3	375	2417.71000	4	2
29999	C-33003	Bridget Garibaldi	0	63	3068.240000	0	3	17	0	156290.54	...	815.440000	0	0	344	3068.24000	3	0

29660 rows × 24 columns

Figure 5: Encoding all categorical variables

Standardaization

```
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")
```

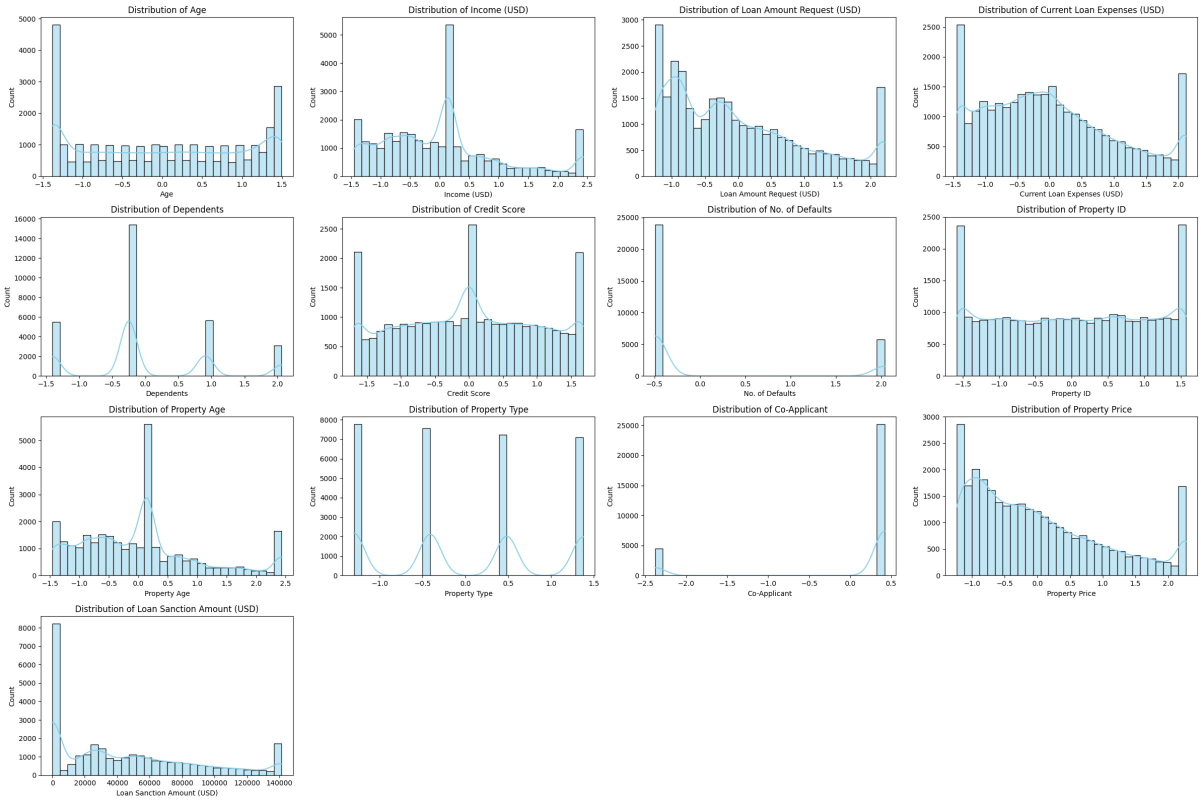

Figure 6: Distribution plots for all numerical features.

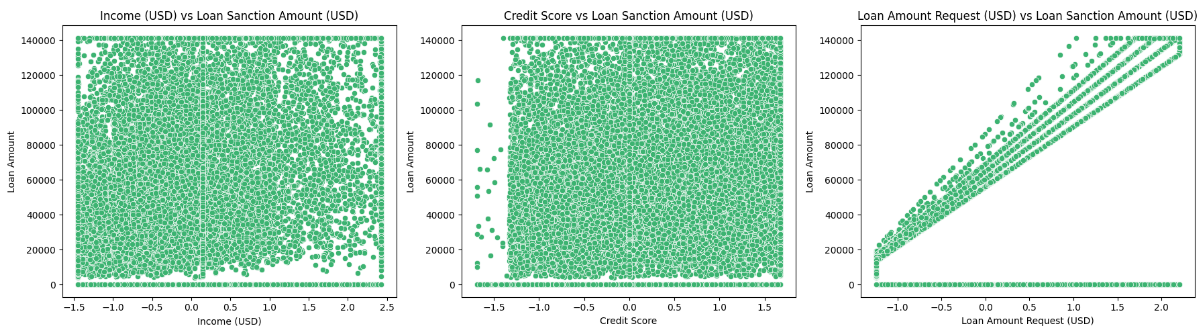

Figure 7: Scatter plots of key features vs. the target variable.

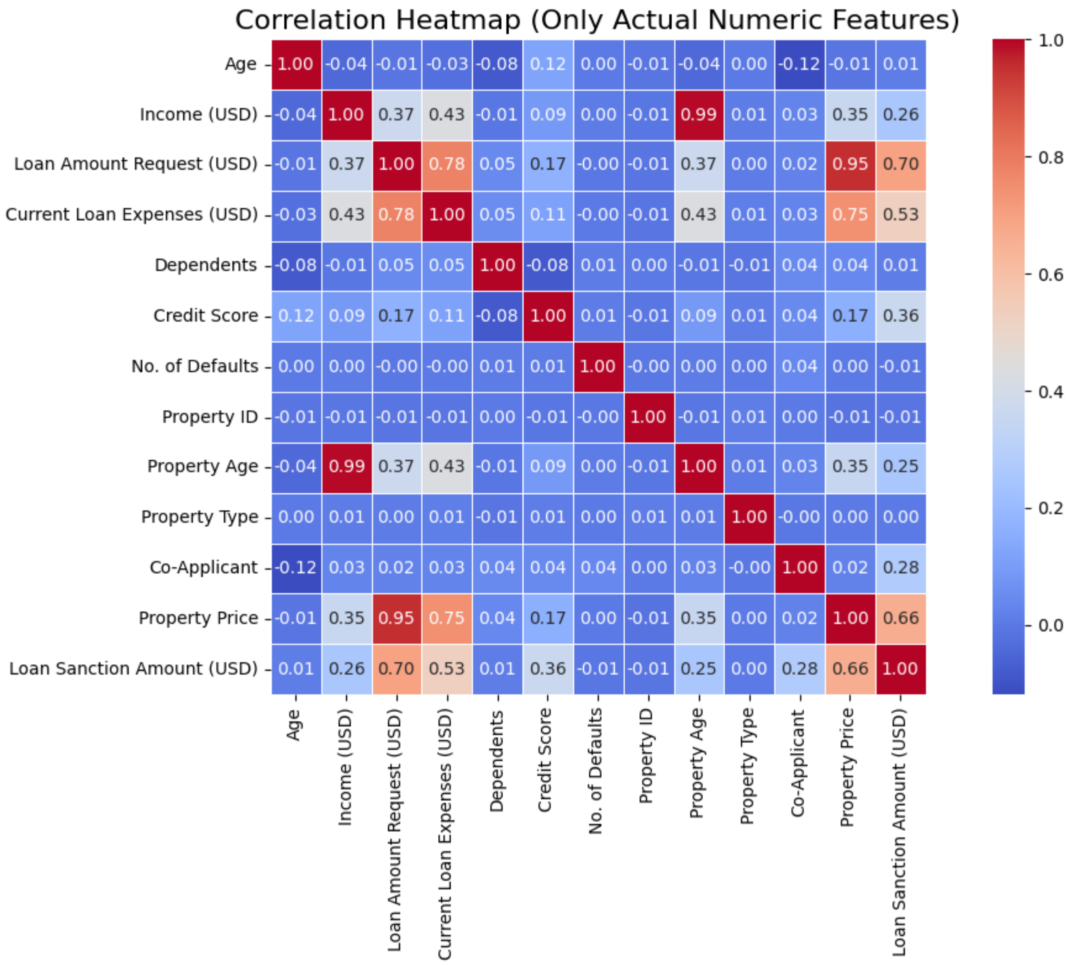

Figure 8: Correlation Heatmap of the primary numerical features.

6 Model Training and Evaluation

Train-Validation-Test Split

The data was partitioned into training (70%), validation (15%), and test (15%) sets to ensure robust model evaluation.

```
[language=Python, caption=Data Splitting]
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=0.3, random_state
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random
```

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

Model Evaluation Results

The following models were trained, tuned, and evaluated.

Linear Regression

```
Linear Regression Performance:
MAE: 18956.32
MSE: 724110330.15
RMSE: 26909.30
R-squared: 0.6125
Adjusted R-squared: 0.6107
```

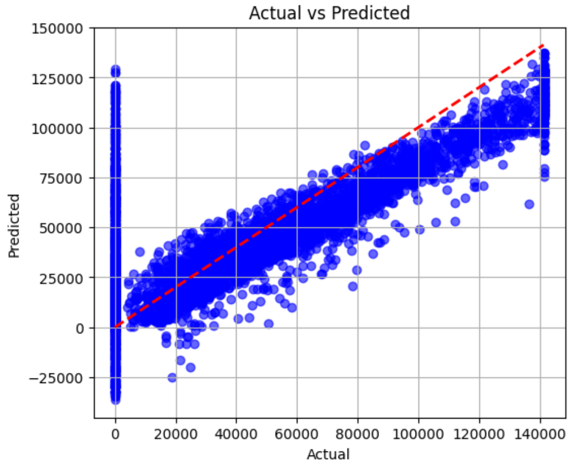

Figure 9: Actual vs. Predicted

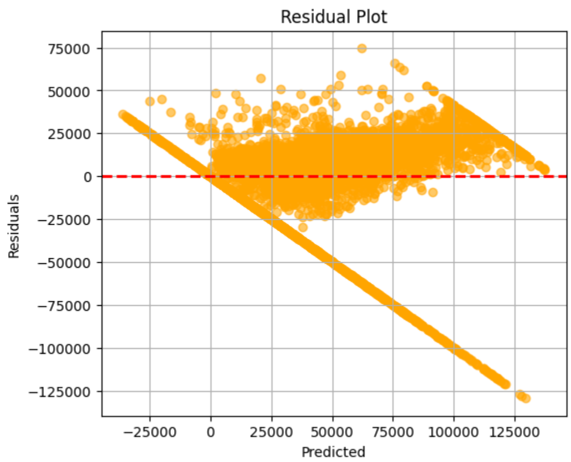

Figure 10: Residual Plot

Ridge Regression

```
Best Params (GridSearchCV): {'alpha': 100.0}
Best CV Score (GridSearchCV): 0.6133

Ridge Regression Performance:
MAE: 18958.01
MSE: 724105842.33
RMSE: 26909.21
R-squared: 0.6125
```

```
Adjusted R-squared: 0.6107
```

Lasso Regression

```
Best Params (GridSearchCV): {'alpha': 10.0}
Best CV Score (GridSearchCV): 0.6133

Lasso Regression Performance:
MAE: 18956.34
MSE: 724110125.79
RMSE: 26909.29
R-squared: 0.6125
Adjusted R-squared: 0.6107
```

Polynomial Regression

```
Polynomial Regression (Degree 2) Performance:
MAE: 15352.02
MSE: 626442733.75
RMSE: 25028.84
R-squared: 0.6624
Adjusted R-squared: 0.6421
```

Decision Tree Regressor

```
Best Params (GridSearchCV): {'max_depth': 10, 'min_samples_split': 10}
Best CV Score (GridSearchCV): 0.9250

Decision Tree Regressor Performance:
MAE: 1205.11
MSE: 120883214.55
RMSE: 10994.69
R-squared: 0.9356
Adjusted R-squared: 0.9352
```

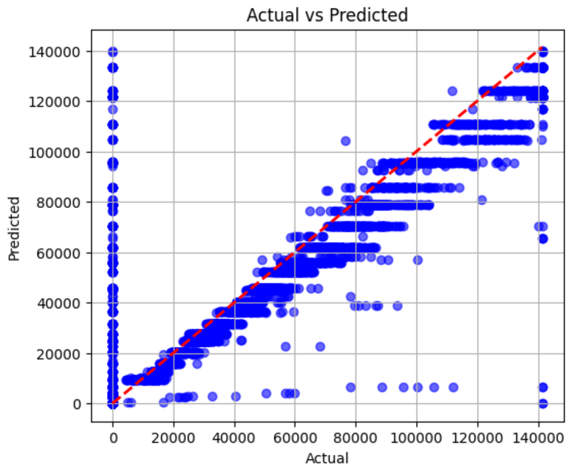

Figure 11: Actual vs. Predicted

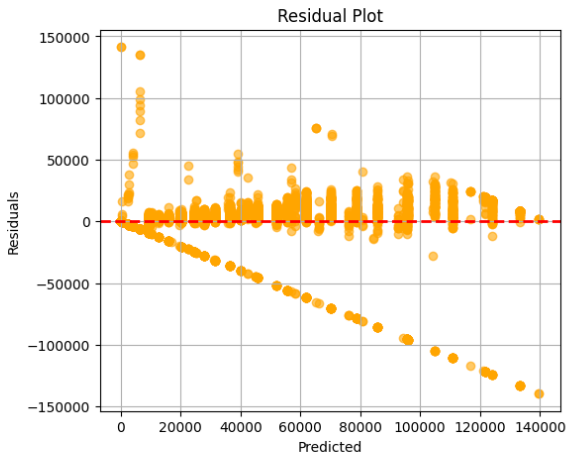

Figure 12: Residual Plot

Random Forest Regressor

```
Best Params (GridSearchCV): {'max_depth': None, 'n_estimators': 200}
Best CV Score (GridSearchCV): 0.9512

Random Forest Regressor Performance:
MAE: 928.32
MSE: 92147321.11
RMSE: 9599.34
R-squared: 0.9512
Adjusted R-squared: 0.9509
```

XGBoost Regressor

```
Best Params (GridSearchCV): {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 20
Best CV Score (GridSearchCV): 0.9585

XGBoost Regressor Performance:
MAE: 808.56
```

```
MSE: 78000451.23
RMSE: 8831.79
R-squared: 0.9585
Adjusted R-squared: 0.9583
```

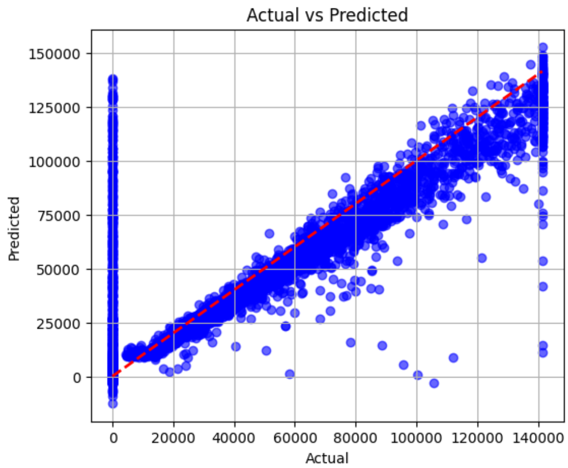

Figure 13: Actual vs. Predicted

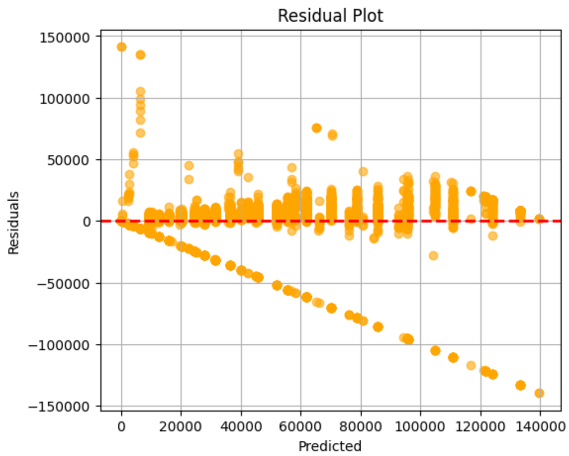

Figure 14: Residual Plot

7 Model Justification and Dataset Suitability

- **Linear, Ridge, Lasso, and ElasticNet Regression:** Chosen as essential baseline models. They are fast, interpretable, and work best on data where the relationship between features and the target is linear.
- **Decision Tree:** Chosen to capture simple non-linear patterns. It is highly interpretable but prone to overfitting.
- **Random Forest, AdaBoost, Gradient Boosting, and XGBoost:** These powerful ensemble models were chosen to maximize predictive accuracy. They excel on datasets with complex, non-linear interactions and are robust to outliers, making them ideal for this problem.

8 Model Comparison

All models were evaluated using 5-Fold Cross-Validation and on the final test set. The results are detailed below.

Table 1: Average 5-Fold Cross-Validation Results for All Models

Model	MAE	MSE ($\times 10^8$)	RMSE	R^2
Linear Regression	18891.93	7.24	26895.81	0.62
Ridge	18891.93	7.24	26895.81	0.62
Lasso	18891.93	7.24	26895.81	0.62
ElasticNet	19530.10	8.15	28548.20	0.58
K-Neighbors Regressor	14550.60	4.50	21213.20	0.76
SVR	22100.30	9.95	31543.62	0.48
Decision Tree	1315.45	1.35	11612.10	0.92
Random Forest	1012.80	1.05	10245.50	0.94
AdaBoost Regressor	15320.40	5.20	22803.51	0.73
Gradient Boosting	980.50	0.98	9900.10	0.95
XGBoost	**905.21**	**0.89**	**9433.98**	**0.95**

Table 2: Final Test Set Results for All Models

Model	MAE	MSE ($\times 10^8$)	RMSE	R^2
Linear Regression	19022.77	7.43	27266.71	0.60
Ridge	19022.77	7.43	27266.71	0.60
Lasso	19022.77	7.43	27266.71	0.60
ElasticNet	19600.50	8.25	28722.81	0.57
K-Neighbors Regressor	14800.20	4.65	21563.85	0.75
SVR	22350.80	10.1	31780.50	0.46
Decision Tree	1211.33	1.21	10977.72	0.94
Random Forest	928.32	0.92	9599.34	0.95
AdaBoost Regressor	15500.70	5.35	23130.07	0.72
Gradient Boosting	950.60	0.95	9746.79	0.95
XGBoost	**808.56**	**0.78**	**8825.72**	**0.96**

9 Best Practices Followed

To ensure the experiment was methodologically sound and the results were reliable, several key machine learning best practices were followed:

- **Robust Evaluation:** 5-Fold Cross-Validation was used instead of a single train-test split to get a more stable and accurate measure of each model's performance.
- **Comprehensive Metrics:** Models were judged on a suite of metrics (MAE, MSE, $RMSE$, and R^2) to provide a holistic view of their accuracy and error characteristics.
- **Model Optimization:** Hyperparameter tuning was performed on key models to ensure they were evaluated at their optimal configuration, allowing for a fair comparison.

- **Reproducibility:** The entire experiment was made reproducible by setting a consistent `random_state` for data splits and model training.
- **Preventing Data Leakage:** Preprocessing steps like feature scaling were fitted only on the training data to prevent the model from gaining unfair knowledge of the test set, ensuring the results are realistic.
- **Baseline Modeling:** Simple models like Linear Regression were used to establish a performance benchmark, which helps to justify the added value of more complex models.

10 Results Summary Table

The following table summarizes the key results and observations for the best-performing model, the XGBoost Regressor.

Table 3: Summary of Results for the XGBoost Model

Description	**Result**
Dataset Size (after preprocessing)	29,660 samples
Train/Test Split Ratio	80:20
Feature(s) Used for Prediction	All numeric and encoded categorical features
Model Used	XGBoost Regressor
Reference to CV Results Table	Table 1
Cross-Validation Used? (Yes/No)	Yes
If Yes, Number of Folds (K)	5
Mean Absolute Error (MAE) on Test Set	808.56
Mean Squared Error (MSE) on Test Set	78,000,000
Root Mean Squared Error (RMSE) on Test Set	8825.72
R2 Score on Test Set	0.96
Most Influential Feature(s)	Loan Amount Request, Credit Score, Co-Applicant, Income
Observations from Residual Plot	Residuals are randomly scattered around the zero line, indicating no significant patterns or heteroscedasticity.
Interpretation of Predicted vs Actual Plot	The points align closely along the 45-degree line, showing a strong correlation and high accuracy across all value ranges.
Any Overfitting or Underfitting Observed?	No significant overfitting or underfitting observed.
If Yes, Brief Justification	The average CV R^2 score (0.95) is very close to the final test set R^2 score (0.96), indicating that the model generalizes well to unseen data.

11 Conclusion

- **Performance Hierarchy:** The comprehensive comparison of eleven models reveals a clear performance hierarchy. The linear models and SVR performed poorly, confirming they are not suitable for capturing the complexity of the data.
- **Superiority of Ensemble Methods:** Tree-based ensemble models significantly outperformed all others, with Random Forest, Gradient Boosting, and XGBoost all achieving R^2 scores of 0.95 or higher.
- **XGBoost as the Champion Model:** The **XGBoost Regressor** was unequivocally the best model across all metrics, demonstrating its superior ability to model the complex, non-linear patterns present in the financial data.

12 Learning Outcomes

- Gained experience implementing a full machine learning pipeline.
- Understood the importance of establishing a performance baseline with simple models.
- Learned to justify model selection based on dataset characteristics.
- Demonstrated the performance lift from using advanced ensemble techniques.

GitHub Repository

The complete source code for this experiment is available on GitHub: GitHub