EXPT – 2 - : Loan Amount Prediction using Linear Regression

REPORT BY GOPIKA GANESAN

AIM:

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

LIBRARIES USED:

NumPy, pandas, scikit learn, seaborn, matplotlib

OBJECTIVE:

Visualize and interpret the linear regression results to gain insights into the model performance.

MATHEMATICAL DESCRIPTION:

Linear Regression is a supervised machine learning algorithm used to predict a continuous target variable. The predicted value is expressed as a linear combination of the input features.

$H\text{ø}(x) = \text{ø}0 + \text{ø}1x$

x – independent feature

ø0 – intercept

ø1 – slope

The cost function for Linear Regression is the error between the predicted values and the actual values.

$J(\text{ø}0, \text{ø}1)=1/2m \sum(h\ \text{ø}(x)^{(i)} - y^{(i)})^2$

We use a convergence algorithm to minimise the above cost function.

Using gradient descent ,

Øj= øj-α϶J(øj)/ ϶ øj

Where α is the learning rate which controls the speed at which convergence should take place.

CODE WITH PLOT:

```
[1] from google.colab import drive
    drive.mount('/content/drive')

Mounted at /content/drive
```

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

from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler,OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

```
[3] df=pd.read_csv('/content/drive/MyDrive/loan.csv')
    df.head()
    df.isnull().sum()

    target = 'Loan Sanction Amount (USD)'

    categorical_features=df.select_dtypes(include=['object']).columns.tolist()
    numerical_features=df.select_dtypes(include=['int64','float64']).columns.tolist()
    numerical_features=[col for col in numerical_features if target!=col]

    numeric_pipeline=Pipeline([
        ('imputer',SimpleImputer(strategy='mean')),
        ('scaler',StandardScaler())
    ])

    categorical_pipeline=Pipeline([
        ('imputer',SimpleImputer(strategy='most_frequent')),
        ('onehot',OneHotEncoder(drop='first',handle_unknown='ignore'))
    ])

    preprocessor=ColumnTransformer([
        ('num',numeric_pipeline,numerical_features),
        ('cat',categorical_pipeline,categorical_features)
    ])

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

```python
X=df.drop(columns=target)
y=df[target]

X_train,X_temp,y_train,y_temp=train_test_split(X,y,random_state=42,test_size=0.3)
X_test,X_val,y_test,y_val=train_test_split(X,y,random_state=42,test_size=0.5)
model=Pipeline([
    ('preprocessor',preprocessor),
    ('regressor',LinearRegression())

])

model.fit(X_train,y_train)
y_test_pred=model.predict(X_test)
y_val_pred=model.predict(X_val)

def evaluate(y_true,y_pred):
  print('MSE',mean_squared_error(y_true,y_pred))
  print('MAE',mean_absolute_error(y_true,y_pred))
  print('r2',r2_score(y_true,y_pred))

print('test')
evaluate(y_test, y_test_pred)
print('validation')
evaluate(y_val, y_val_pred)
```

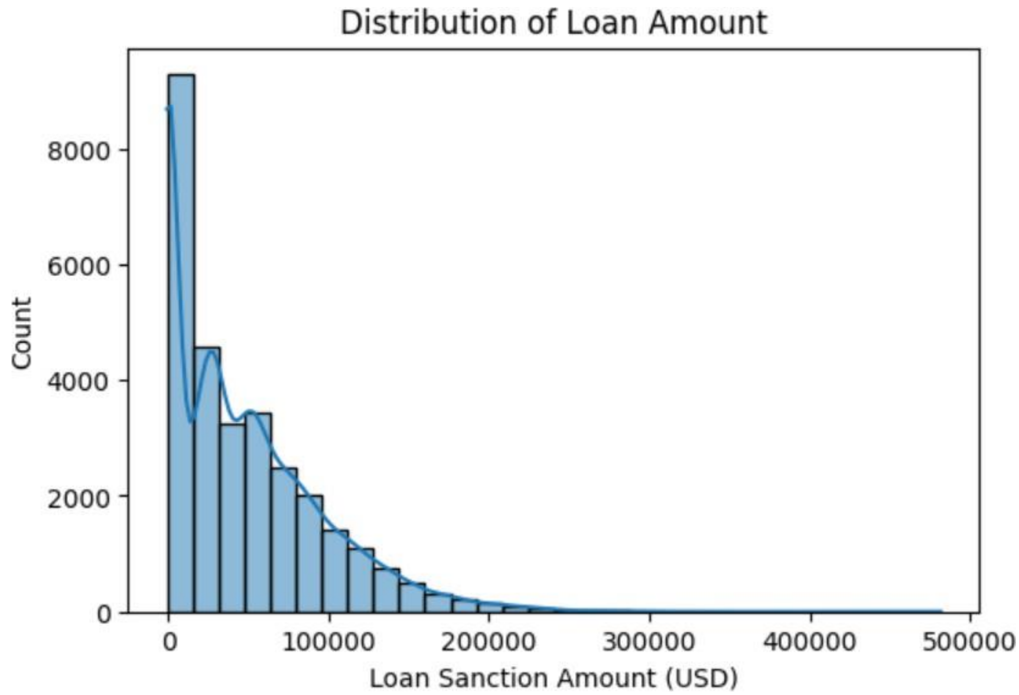
```
test
MSE 183.77354619623188
MAE 3.1224452197002073
r2 0.9999999232337329
validation
MSE 596521885.5033314
MAE 12956.97249947
r2 0.7355852444481442
/usr/local/lib/python3.11/dist-packages/sklearn/preprocessing/_encoders.py:246: UserWarning: Found unknown categories in columns [0, 1, 4] during tr
  warnings.warn(
```

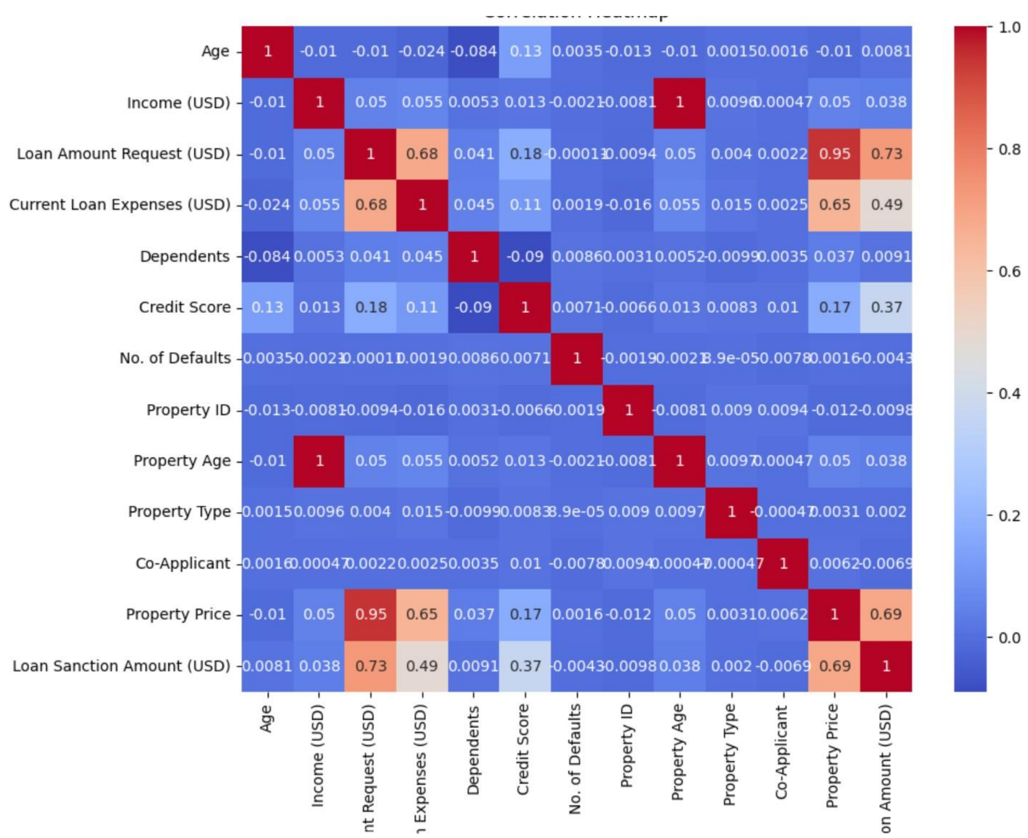
```python
[4] plt.figure(figsize=(6,4))
    sns.histplot(df[target], kde=True, bins=30)
    plt.title('Distribution of Loan Amount')
    plt.show()
    # Correlation Heatmap
    numeric_df = df.select_dtypes(include=['number'])
    # Plot heatmap
    plt.figure(figsize=(10, 8))
    sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm')
    plt.title('Correlation Heatmap')
    plt.show()
    # Actual vs Predicted
    plt.figure(figsize=(6,6))
    plt.scatter(y_test, y_test_pred, alpha=0.6, color='blue')
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
    plt.xlabel('Actual Loan Amount')
```

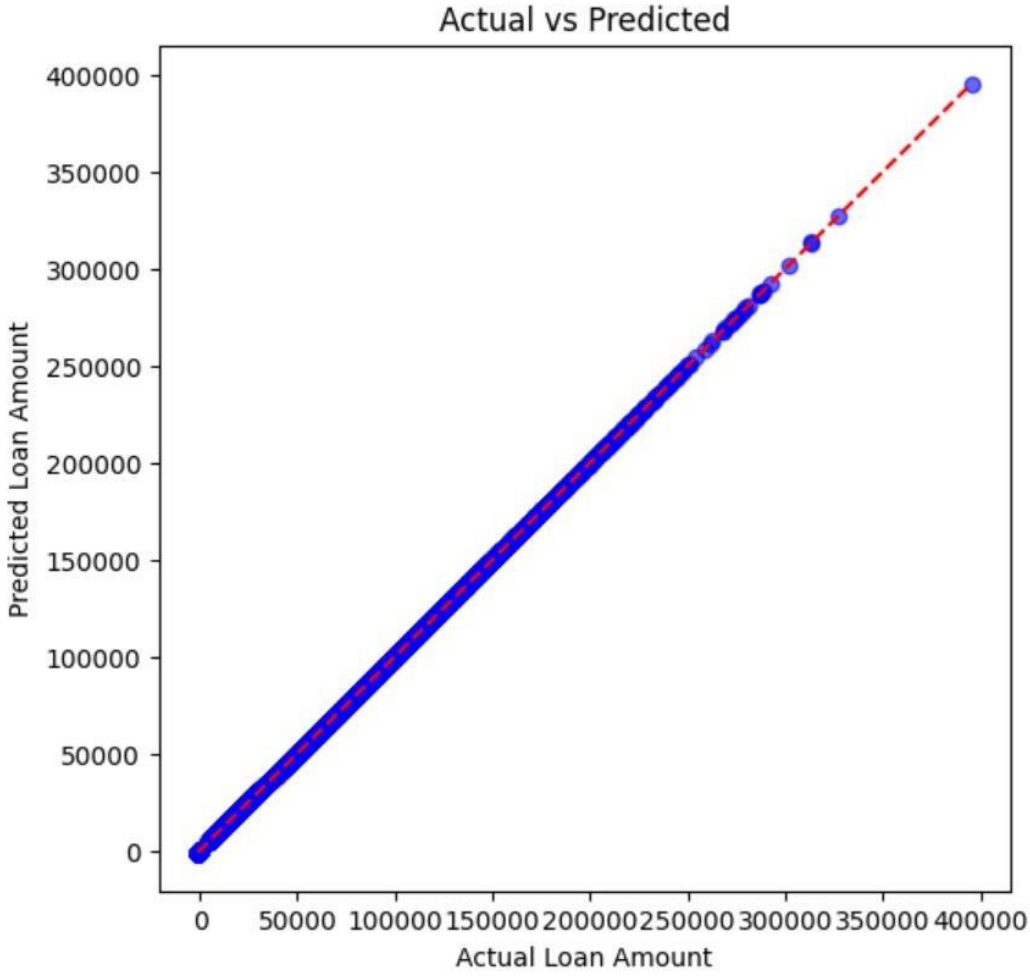
```python
plt.ylabel('Predicted Loan Amount')
plt.title('Actual vs Predicted')
plt.show()
# Residual Plot
residuals = y_test - y_test_pred
plt.figure(figsize=(6,4))
sns.histplot(residuals, kde=True)
plt.title('Residuals Distribution')
plt.show()
# Boxplots of numerical features
for col in numerical_features:
    plt.figure(figsize=(5,3))
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col}')
    plt.show()
```

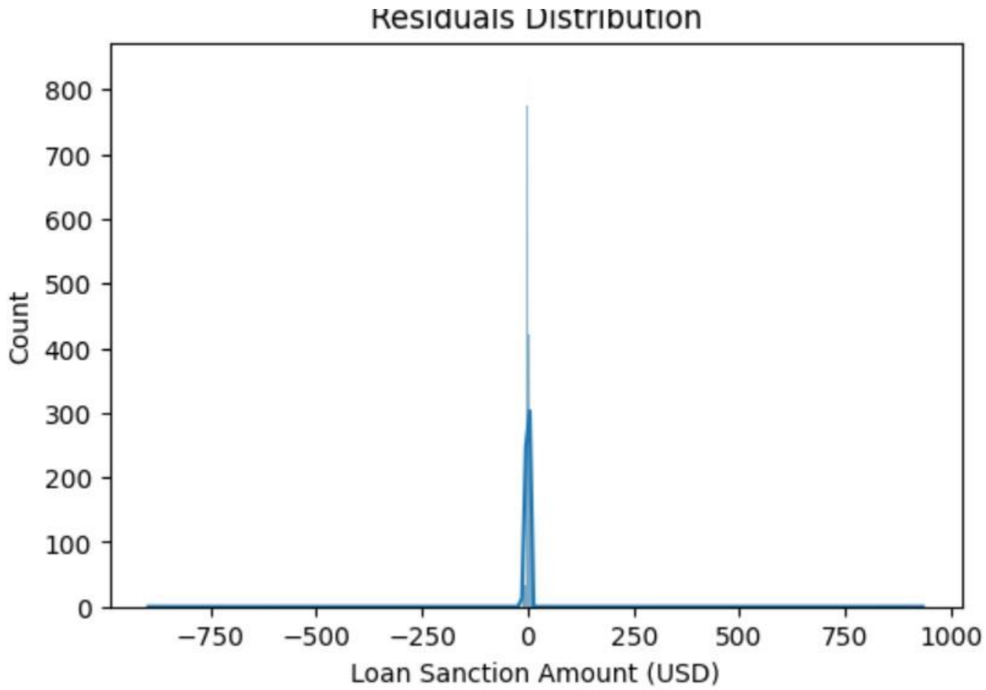
```python
# 1. Remove any rows with NaNs in target
df_clean = df.dropna(subset=['Loan Sanction Amount (USD)'])

# 2. Define target and features
X = df_clean.drop(['Loan Sanction Amount (USD)'], axis=1)
y = df_clean['Loan Sanction Amount (USD)']

# Ensure y is numeric and has no NaNs
y = pd.to_numeric(y, errors='coerce')
X = X.reset_index(drop=True)
y = y.reset_index(drop=True)

# 3. Apply KFold CV
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Store metrics
mae_list = []
mse_list = []
rmse_list = []
r2_list = []

fold = 1
results = []

for train_index, test_index in kf.split(X):
    X_train_cv, X_test_cv = X.iloc[train_index], X.iloc[test_index]
    y_train_cv, y_test_cv = y.iloc[train_index], y.iloc[test_index]
```

```python
    # Create and train pipeline
    model = Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', LinearRegression())
    ])

    model.fit(X_train_cv, y_train_cv)
    y_pred_cv = model.predict(X_test_cv)

    # Compute metrics
    mae = mean_absolute_error(y_test_cv, y_pred_cv)
    mse = mean_squared_error(y_test_cv, y_pred_cv)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test_cv, y_pred_cv)

    # Save to lists
    mae_list.append(mae)
    mse_list.append(mse)
    rmse_list.append(rmse)
    r2_list.append(r2)

    results.append([f"Fold {fold}", round(mae, 2), round(mse, 2), round(rmse, 2), round(r2, 2)])
    fold += 1
```

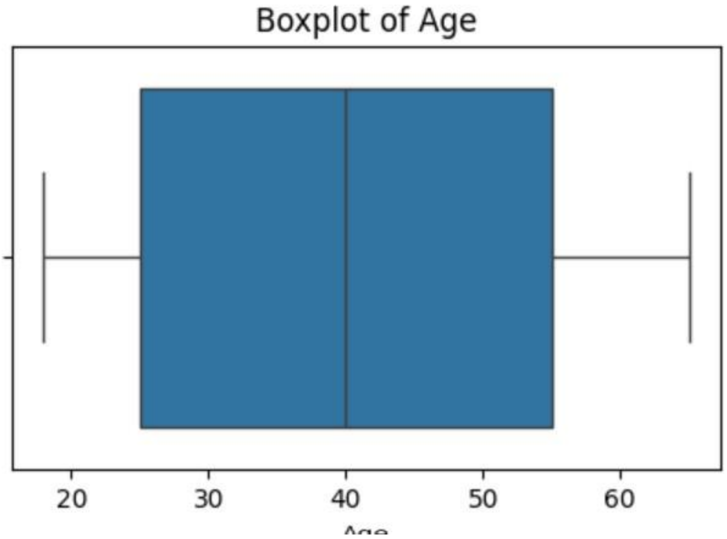

Distribution of Loan Amount
Count
Loan Sanction Amount (USD)
0
2000
4000
6000
8000
0
100000
200000
300000
400000
500000


Age
Income (USD)
Loan Amount Request (USD)
Current Loan Expenses (USD)
Dependents
Credit Score
No. of Defaults
Property ID
Property Age
Property Type
Co-Applicant
Property Price
Loan Sanction Amount (USD)

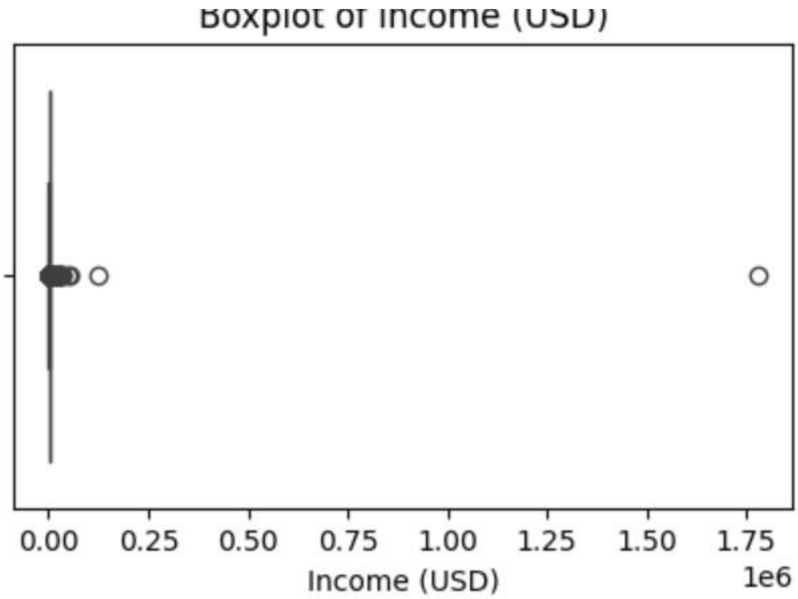

Actual vs Predicted
Predicted Loan Amount
Actual Loan Amount
0
50000
100000
150000
200000
250000
300000
350000
400000

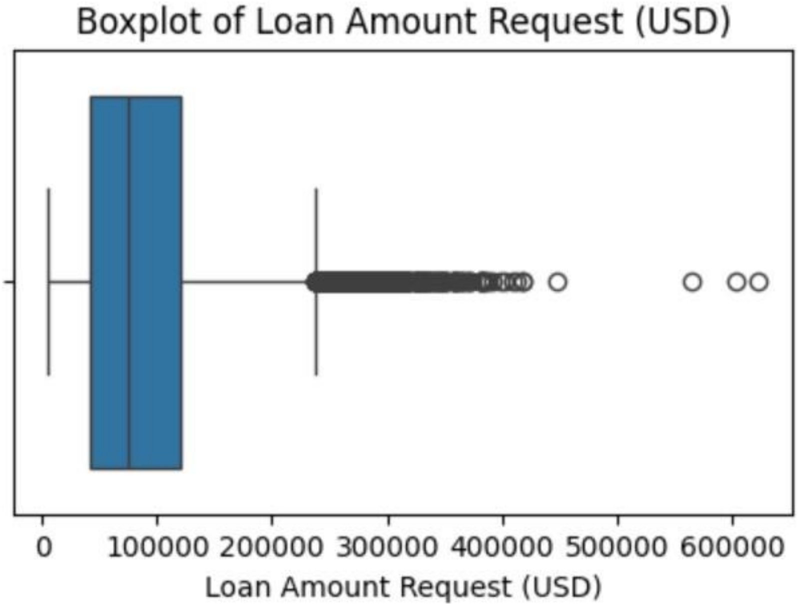


Residuals Distribution
Count
800
700
600
500
400
300
200
100
0
−750
−500
−250
0
250
500
750
1000
Loan Sanction Amount (USD)



Boxplot of Age
20
30
40
50
60





RESULTS TABLES

FOLD	MAE	MSE	RMSE	R2
1	21578.16	1.018970e+09	31921.31	0.55
2	21665.83	9.744059e+08	31215.47	0.57
3	21459.55	1.065990e+09	32649.50	0.54
4	21508.81	9.190246e+08	30315.42	0.62
5	21757.48	9.953794e+08	31549.63	0.58
avg	21593.97	9.947540e+08	31530.27	0.57

Description	
Dataset Size (after preprocessing)	29660
Train/Test Split Ratio	80:20
Feature(s) Used for Prediction	Income (USD), Credit Score, Age, Type of Employment, etc.
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	3.1224452197002073
Mean Squared Error (MSE) on Test Set	183.77354619623188
Root Mean Squared Error (RMSE) on Test Set	13.56
R2 Score on Test Set	1.0000
Adjusted R2 Score on Test Set	1.0000
Most Influential Feature(s)	Credit Score, Income (USD), No. of Defaults
Observations from Residual Plot	Residuals are randomly scattered around zero; no strong pattern observed, indicating good linearity assumption.
Interpretation of Predicted vs Actual Plot	predictions are close to actual values with some variance.
Any Overfitting or Underfitting Observed?	No
If Yes, Brief Justification (e.g., training vs test error, residual patterns)	Train and test R^2 scores are close; residuals show no major bias.

SVR:

SVR Cross-Validation Results Table:

	Fold	MAE	MSE	RMSE	R2 Score
0	Fold 1	21001.15	1.211807e+09	34811.02	0.47
1	Fold 2	21413.82	1.208246e+09	34759.84	0.47
2	Fold 3	21214.21	1.207789e+09	34753.27	0.48
3	Fold 4	21270.16	1.214370e+09	34847.81	0.50
4	Fold 5	21363.55	1.252735e+09	35393.99	0.47
5	Average	21252.58	1.218989e+09	34913.18	0.48

BOOSTING:

Gradient Boosting Cross-Validation Results Table:

	Fold	MAE	MSE	RMSE	R2 Score
0	Fold 1	12750.13	5.713971e+08	23903.91	0.75
1	Fold 2	13196.62	5.768728e+08	24018.18	0.75
2	Fold 3	12563.73	4.965877e+08	22284.25	0.78
3	Fold 4	12744.45	5.120416e+08	22628.34	0.79
4	Fold 5	12973.80	5.733550e+08	23944.83	0.76
5	Average	12845.74	5.460508e+08	23355.90	0.76

XGBoost Cross-Validation Results Table:

	Fold	MAE	MSE	RMSE	R2 Score
0	Fold 1	12498.38	5.573444e+08	23608.14	0.75
1	Fold 2	12711.05	5.632942e+08	23733.82	0.75
2	Fold 3	12020.17	4.776799e+08	21855.89	0.79
3	Fold 4	12373.77	5.003515e+08	22368.54	0.79
4	Fold 5	12744.27	5.579686e+08	23621.36	0.76
5	Average	12469.53	5.313277e+08	23037.55	0.77

AdaBoost Cross-Validation Results Table:

	Fold	MAE	MSE	RMSE	R2 Score
0	Fold 1	26494.54	1.126522e+09	33563.71	0.50
1	Fold 2	26650.91	1.110048e+09	33317.38	0.51
2	Fold 3	26450.61	1.106553e+09	33264.89	0.52
3	Fold 4	26272.18	1.095463e+09	33097.78	0.55
4	Fold 5	26974.91	1.155263e+09	33989.17	0.51
5	Average	26568.63	1.118770e+09	33446.59	0.52

BEST PRACTICES:

- Missing values in numerical and categorical columns were handled using appropriate imputers (mean and most_frequent).
- Features were standardized using StandardScaler to ensure uniform scaling for the regression model.
- Categorical variables were encoded using OneHotEncoder with drop='first' to prevent multicollinearity.
- The entire workflow was organized using Pipeline and ColumnTransformer for clean and reusable code.

- Model performance was evaluated using multiple metrics and validated using K-Fold cross-validation.

LEARNING OUTCOMES:

- Understood the mathematical and practical working of Linear Regression models.
- Learned how to preprocess data efficiently using Scikit-learn pipelines.
- Gained experience with evaluation metrics like MAE, MSE, RMSE, R^2, and Adjusted R^2.
- Learned how to validate models using K-Fold cross-validation and interpret residual plots.
- Developed skills in interpreting feature importance and diagnosing model fit visually.