**Ex2 - Loan Amount Prediction Using Linear Regression And Visualize The Interpretation**

**Colab Link:**

[Colab Link](https://colab.research.google.com/drive/1EJ94L06gKKylPuT3zz9lE7tA0LnncVoj?usp=sharing)

**Aim:**

To understand linear regression and implement it in predicting the loan amount of individuals

**Code & Output:**

**Import Dependencies**

import pandas as pd

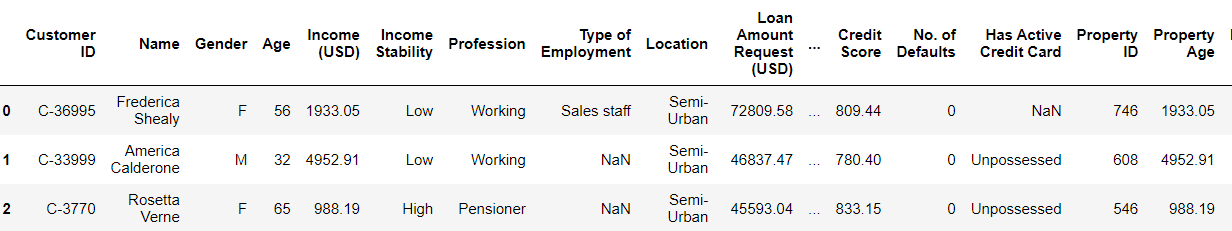
import numpy as np

**Load the Dataset**

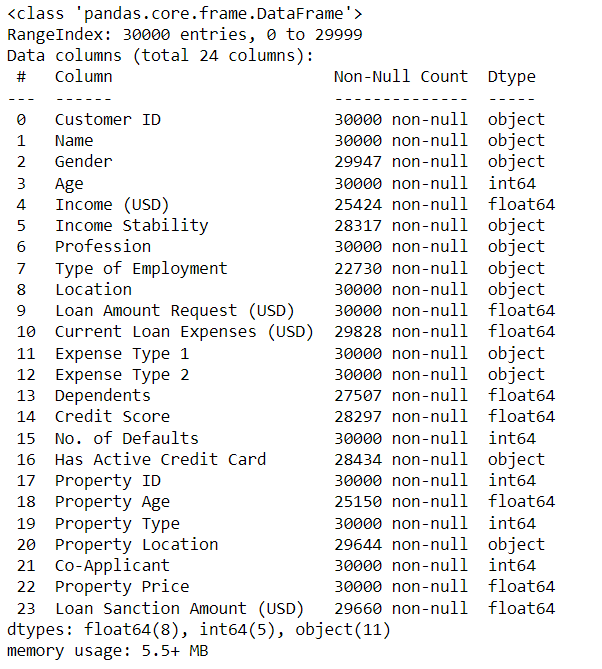
train = pd.read\_csv('train.csv')

**Display Sample Rows**

train.head()



train.info()



**Graphical Plots**

import matplotlib.pyplot as plt

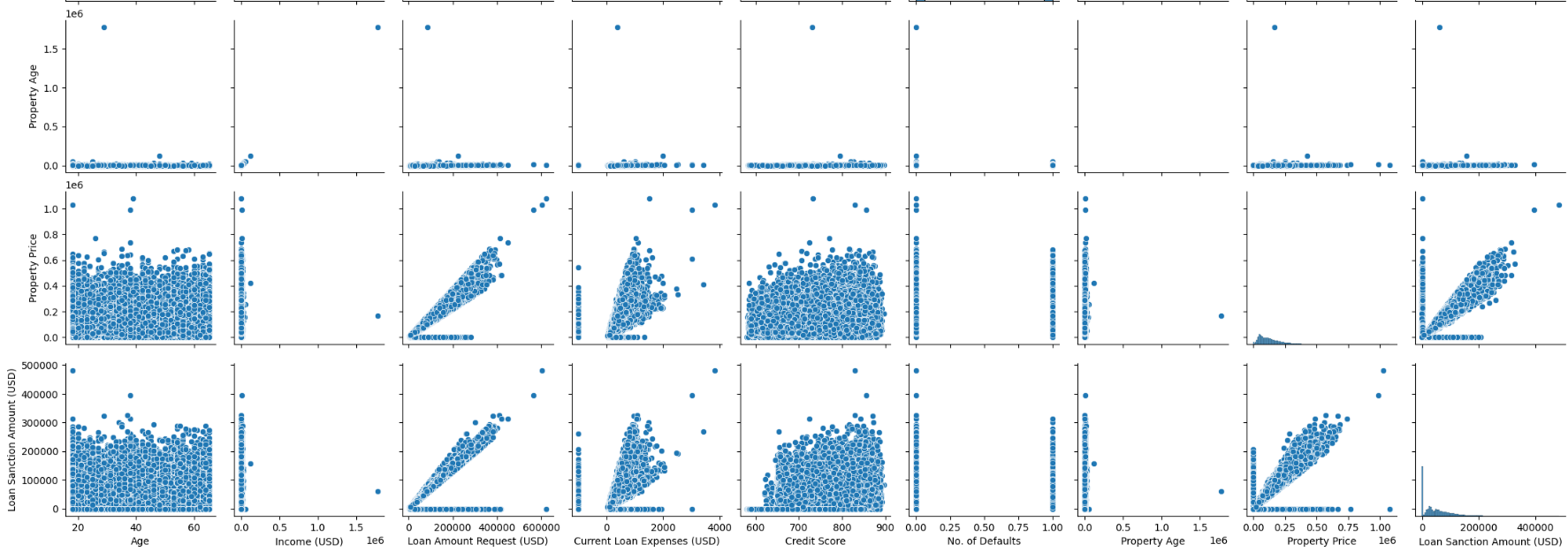
import seaborn as sns

# Distribution of Numerical Features

numerical\_features = ['Age', 'Income (USD)', 'Loan Amount Request (USD)', 'Current Loan Expenses (USD)', 'Credit Score', 'No. of Defaults', 'Property Age', 'Property Price', 'Loan Sanction Amount (USD)']

sns.pairplot(train[numerical\_features])

plt.show()



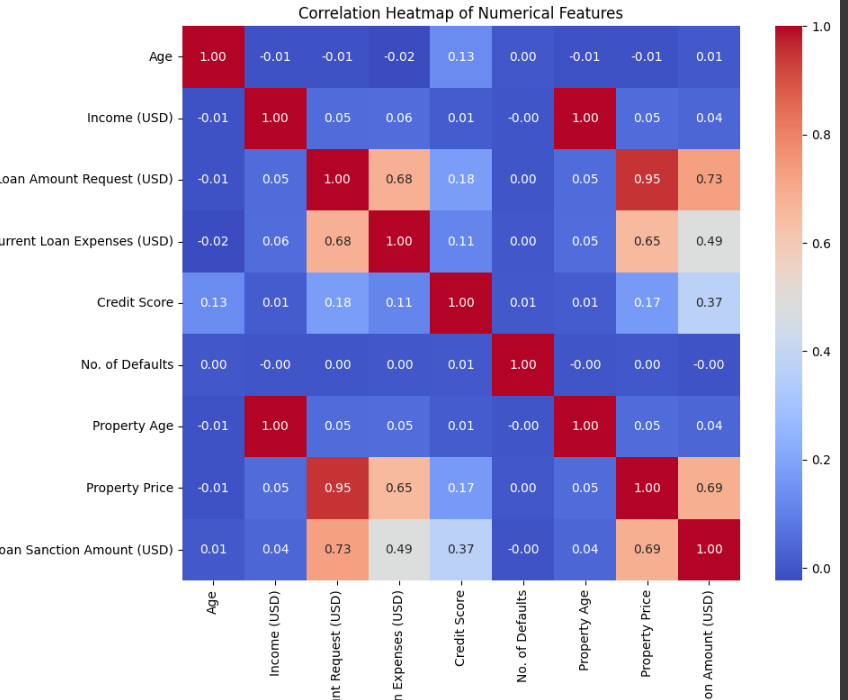
# Correlation Heatmap

plt.figure(figsize=(10, 8))

sns.heatmap(train[numerical\_features].corr(), annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Heatmap of Numerical Features')

plt.show()



# Count of Categorical Features

categorical\_features = ['Gender', 'Income Stability', 'Profession', 'Type of Employment',

'Location', 'Expense Type 1', 'Expense Type 2', 'Property Type',

'Property Location', 'Has Active Credit Card', 'Co-Applicant']

for feature in categorical\_features:

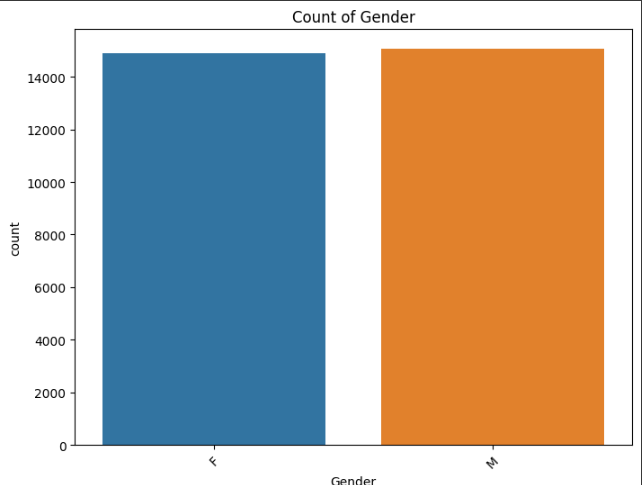
plt.figure(figsize=(8, 6))

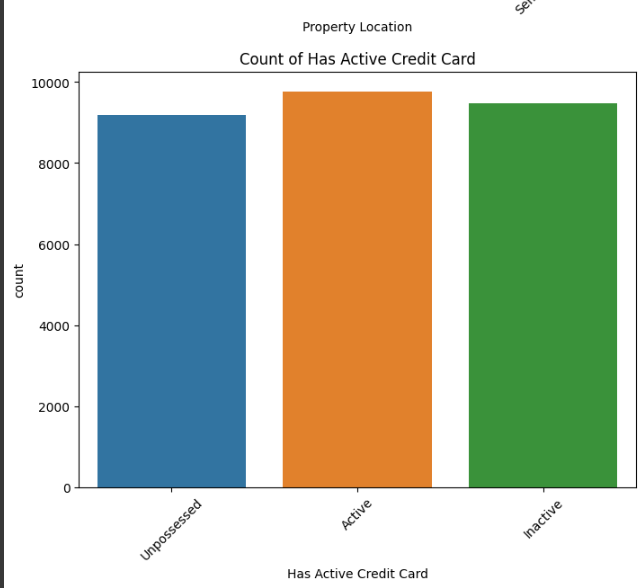
sns.countplot(data=train, x=feature)

plt.title(f'Count of {feature}')

plt.xticks(rotation=45)

plt.show()





# Relationship between Categorical Features and Loan Sanction Amount

for feature in categorical\_features:

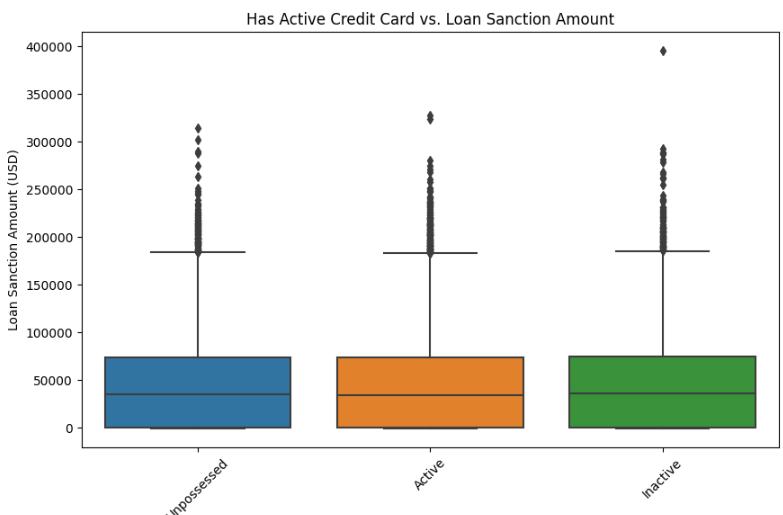
plt.figure(figsize=(10, 6))

sns.boxplot(data=train, x=feature, y='Loan Sanction Amount (USD)')

plt.title(f'{feature} vs. Loan Sanction Amount')

plt.xticks(rotation=45)

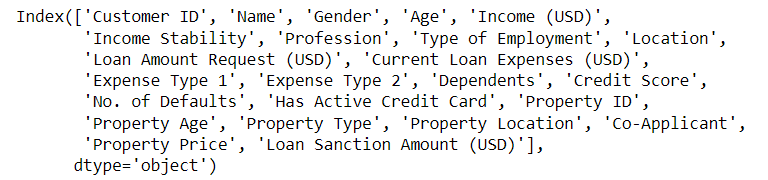
plt.show()



**Pre Processing (Handling missing values, Encoding, Normalization, Standardization)**

**Print Columns**

train.columns



**Print Index**

train.index

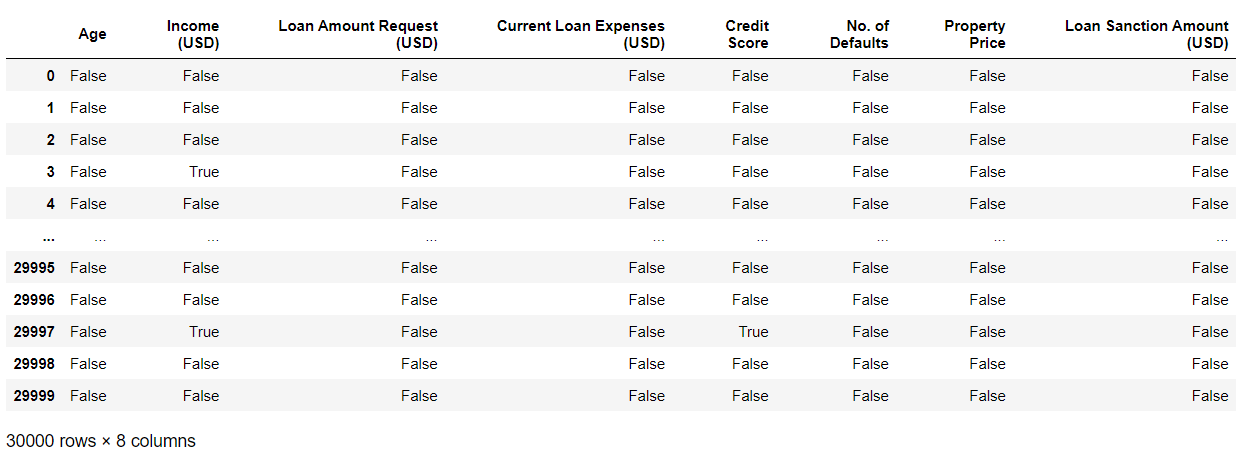


**Drop Uncessary Columns**

train = train.drop(['Customer ID','Name','Gender','Income Stability','Profession','Type of Employment','Location','Expense Type 1','Expense Type 2','Dependents','Has Active Credit Card','Property ID','Property Age','Property Type','Property Location','Co-Applicant'],axis=1)

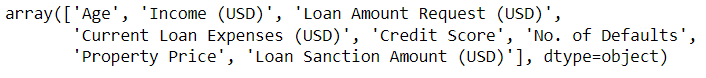
**Print Null Values**

train.isna()



arr = np.array(train.columns)

arr



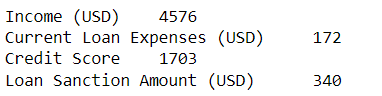
**Print Column and NULL Count**

for index in range(len(arr)):

count = train[train[arr[index]].isna() == True].shape[0]

if(count != 0):

print(f"{arr[index]}\t{count}")

\

**Handling NULL Values**

**Fill With Mean Values**

train["Age"].fillna(train["Age"].mean(), inplace=True)

train["Income (USD)"].fillna(train["Income (USD)"].mean(), inplace=True)

train["Loan Amount Request (USD)"].fillna(train["Loan Amount Request (USD)"].mean(),inplace=True)

train["Loan Sanction Amount (USD)"].fillna(train["Loan Sanction Amount (USD)"].mean(),inplace=True)

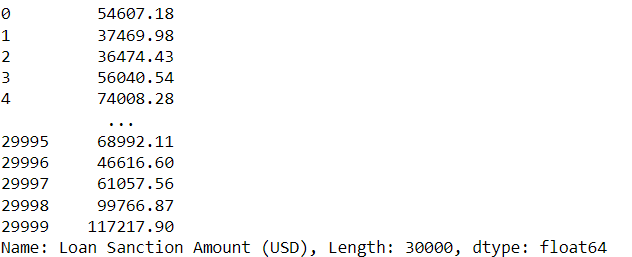
train["Current Loan Expenses (USD)"].fillna(train["Current Loan Expenses (USD)"].mean(),inplace=True)

train["Credit Score"].fillna(train["Credit Score"].mean(),inplace=True)

**Replace Outlier Values**

train["Loan Amount Request (USD)"].replace(-999,train["Loan Amount Request (USD)"].mean())

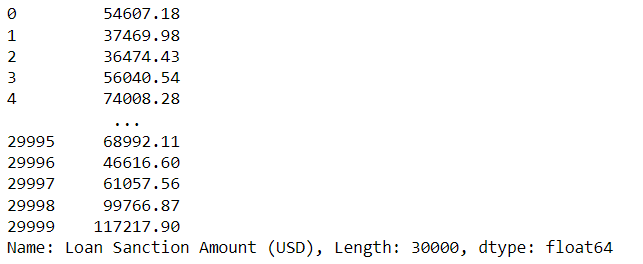
train["Loan Sanction Amount (USD)"].replace(-999,train["Loan Sanction Amount (USD)"].mean())



**Drop Column To Be Predicted**

y = train["Loan Sanction Amount (USD)"]

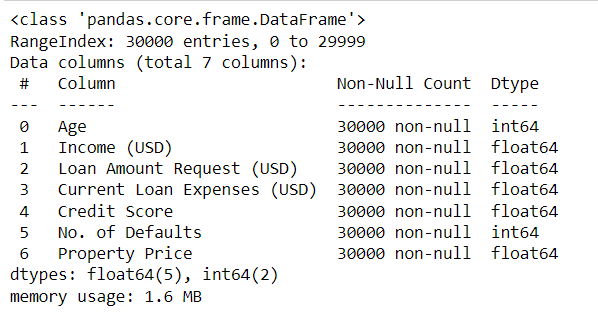
y



train = train.drop(["Loan Sanction Amount (USD)"],axis=1)

**Model**

train.info()



from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import mean\_absolute\_error, r2\_score

from sklearn.metrics import median\_absolute\_error, explained\_variance\_score

# Define features (X) and target variable (y)

X = train[['Age', 'Income (USD)', 'Loan Amount Request (USD)', 'Current Loan Expenses (USD)', 'Credit Score', 'No. of Defaults', 'Property Price']]

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Initialize the linear regression model

model = LinearRegression()

# Train the model

model.fit(X\_train, y\_train)

# Predict on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

print('Mean Squared Error:', mse)

# Calculate Mean Absolute Error

mae = mean\_absolute\_error(y\_test, y\_pred)

print('Mean Absolute Error:', mae)

# Calculate R-squared

r2 = r2\_score(y\_test, y\_pred)

print('R-squared:', r2)

# Calculate Median Absolute Error

medae = median\_absolute\_error(y\_test, y\_pred)

print('Median Absolute Error:', medae)

# Calculate Explained Variance Score

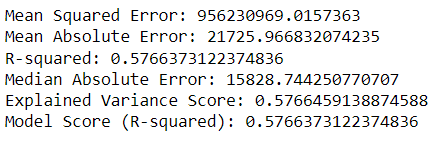
evs = explained\_variance\_score(y\_test, y\_pred)

print('Explained Variance Score:', evs)

# Score the model on the test set

score = model.score(X\_test, y\_test)

print('Model Score (R-squared):', score)



import matplotlib.pyplot as plt

import seaborn as sns

# Residual Plot

residuals = y\_test - y\_pred

plt.figure(figsize=(8, 6))

sns.scatterplot(x=y\_pred, y=residuals)

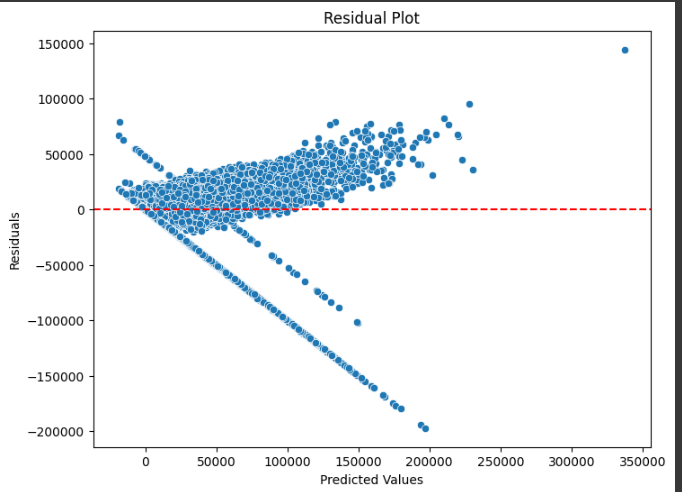
plt.axhline(y=0, color='r', linestyle='--')

plt.title('Residual Plot')

plt.xlabel('Predicted Values')

plt.ylabel('Residuals')

plt.show()



# Actual vs. Predicted Values Plot

plt.figure(figsize=(8, 6))

sns.scatterplot(x=y\_test, y=y\_pred)

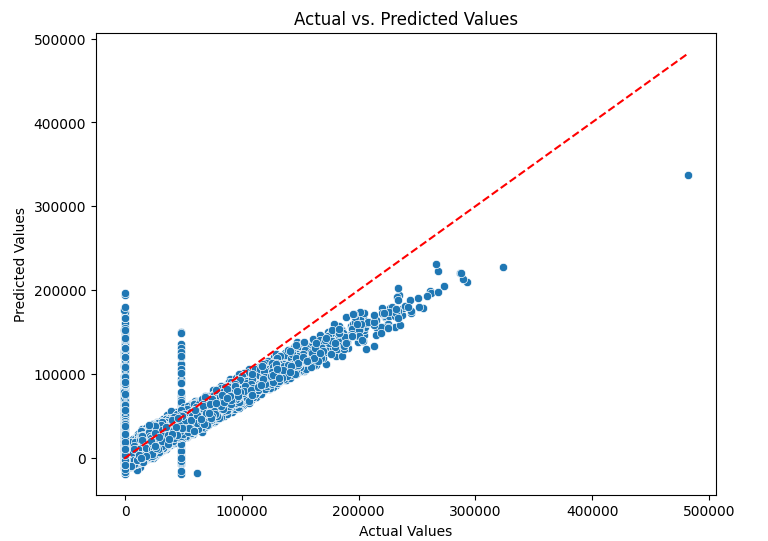
plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], color='r', linestyle='--')

plt.title('Actual vs. Predicted Values')

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.show()



# Distribution Plot of Residuals

plt.figure(figsize=(8, 6))

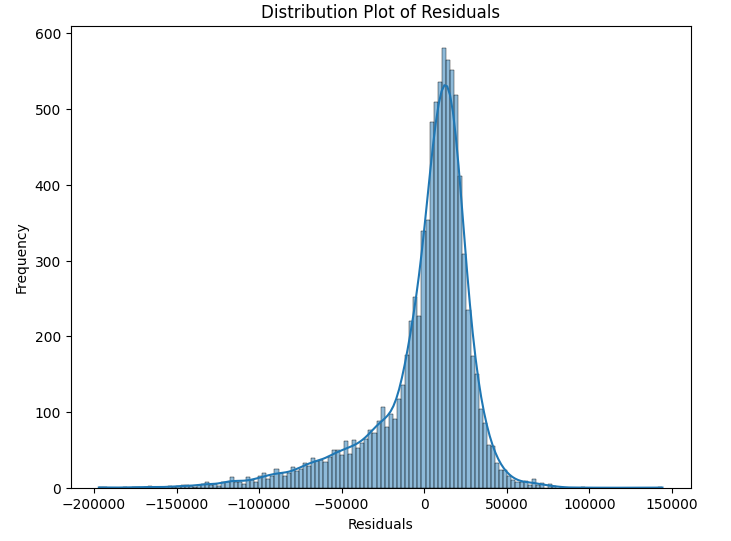
sns.histplot(residuals, kde=True)

plt.title('Distribution Plot of Residuals')

plt.xlabel('Residuals')

plt.ylabel('Frequency')

plt.show()



import matplotlib.pyplot as plt

import numpy as np

# Scatter plot of actual vs. predicted loan sanction amount

plt.figure(figsize=(10, 6))

plt.scatter(y\_test, y\_pred, color='blue', alpha=0.5)

plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], color='red', linestyle='--')

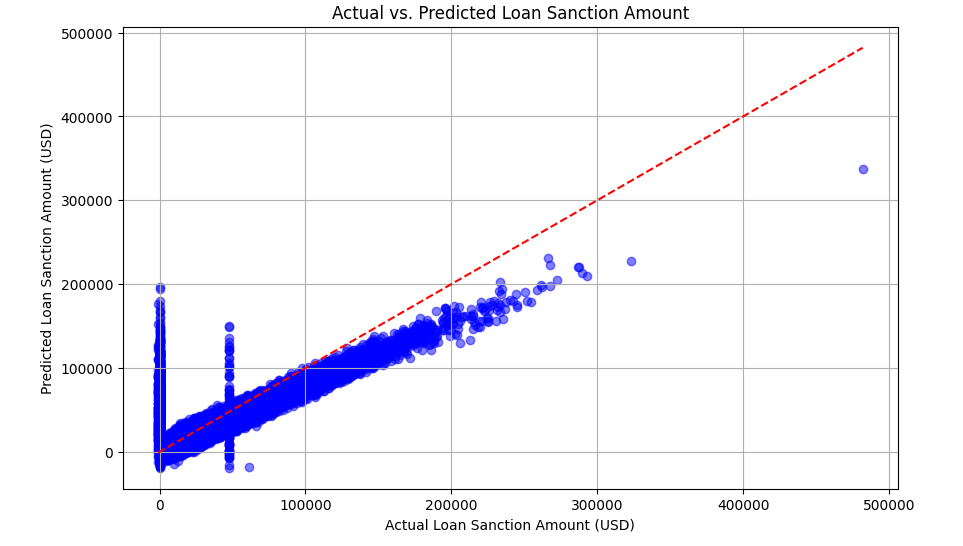
plt.xlabel('Actual Loan Sanction Amount (USD)')

plt.ylabel('Predicted Loan Sanction Amount (USD)')

plt.title('Actual vs. Predicted Loan Sanction Amount')

plt.grid(True)

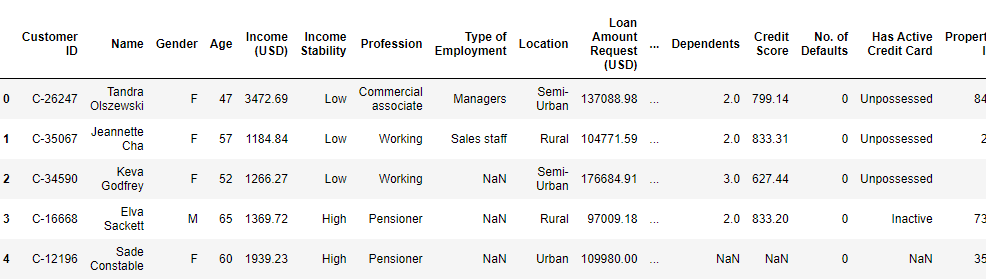
plt.show()



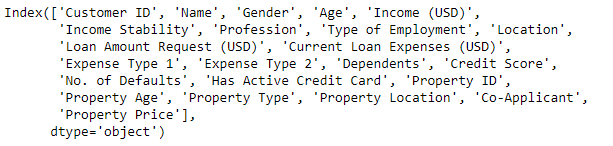
**Test**

test = pd.read\_csv('test.csv')

test.head()



test.columns



# Preprocess the test data, selecting the same features as in the training data

test\_features = test[['Age', 'Income (USD)', 'Loan Amount Request (USD)', 'Current Loan Expenses (USD)',

'Credit Score', 'No. of Defaults', 'Property Price']]

# Replace '?' with NaN

test\_features.replace('?', float('nan'), inplace=True)

# Convert object columns to numeric

test\_features = test\_features.apply(pd.to\_numeric, errors='coerce')

# Fill missing values with mean

test\_features.fillna(test\_features.mean(), inplace=True)

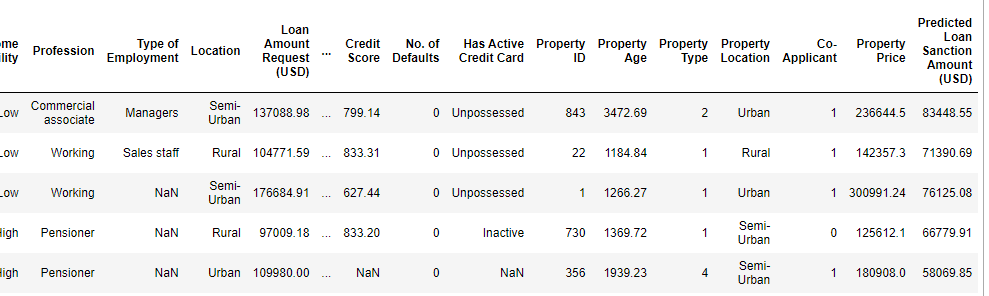
# Make predictions on the processed test data

test\_predictions = model.predict(test\_features)

# Add the predictions to the test dataframe

test['Predicted Loan Sanction Amount (USD)'] = test\_predictions.round(2)

test.head()



test['Predicted Loan Sanction Amount (USD)']

