In [30]:
```
import os
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classificat
from sklearn.model_selection import GridSearchCV
from sklearn.impute import SimpleImputer
from sklearn.metrics import roc_curve, auc

# Load dataset
os.chdir("/Users/ayeshasiddiqha/Downloads")
df = pd.read_csv('logistic_regression.csv')

# Inspect the first few rows
print(df.head())

# Data summary (info and description)
print(df.info())
print(df.describe())

# Check for missing values
print(df.isnull().sum())

# Convert categorical columns to 'category' dtype
categorical_columns = ['emp_title', 'emp_length', 'home_ownership', 'veri
                       'loan_status', 'purpose', 'title', 'earliest_cr_li
                       'application_type', 'address', 'issue_d']
for col in categorical_columns:
    df[col] = df[col].astype('category')

# Univariate Analysis - Loan Amount Distribution
sns.histplot(df['loan_amnt'], kde=True)
plt.title('Loan Amount Distribution')
plt.show()

# Univariate Analysis - Home Ownership Distribution
sns.countplot(x='home_ownership', data=df)
plt.title('Home Ownership Distribution')
plt.show()

# Bivariate Analysis - Loan Amount vs Installment
sns.scatterplot(x='loan_amnt', y='installment', data=df)
plt.title('Loan Amount vs Installment')
plt.show()

# Identify outliers with boxplot
sns.boxplot(x=df['loan_amnt'])
plt.title('Boxplot for Loan Amount')
plt.show()

sns.boxplot(x=df['installment'])
plt.title('Boxplot for Installment')
plt.show()

# Check for duplicate rows
```

```
print(f"Duplicate rows: {df.duplicated().sum()}")

# Impute missing values for numeric columns with the mean
df['annual_inc'] = df['annual_inc'].fillna(df['annual_inc'].mean())

# Impute missing values for categorical columns with the mode
for col in categorical_columns:
    df[col] = df[col].fillna(df[col].mode()[0])

# Convert emp_length to numeric (e.g., ' 60 months' -> 60)
df['emp_length'] = df['emp_length'].replace({'10+ years': '10', ' years':
df['emp_length'] = pd.to_numeric(df['emp_length'], errors='coerce')

# Clean the 'term' column by removing the 'months' part and converting it
df['term'] = df['term'].str.replace(' months', '').astype(int)
df['term'] = pd.to_numeric(df['term'], errors='coerce')

# Outlier Treatment using IQR for loan_amnt
Q1 = df['loan_amnt'].quantile(0.25)
Q3 = df['loan_amnt'].quantile(0.75)
IQR = Q3 - Q1
loan_amnt_outlier_condition = (df['loan_amnt'] < (Q1 - 1.5 * IQR)) | (df[
df = df[~loan_amnt_outlier_condition]

# Feature Engineering - High Loan Flag
df['high_loan'] = df['loan_amnt'].apply(lambda x: 1 if x > 20000 else 0)

# Extract month and year from 'issue_d'
df['issue_month'] = pd.to_datetime(df['issue_d'], errors='coerce').dt.mon
df['issue_year'] = pd.to_datetime(df['issue_d'], errors='coerce').dt.year

# Split data into features (X) and target (y)
X = df.drop(columns=['loan_status', 'grade', 'sub_grade', 'emp_title', 't
                     'issue_d', 'earliest_cr_line', 'initial_list_status'
                     'verification_status', 'purpose', 'application_type'
y = df['loan_status'].apply(lambda x: 1 if x == 'Fully Paid' else 0)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,

# Step 1: Impute missing values using the mean strategy
imputer = SimpleImputer(strategy='mean')

# Fit and transform the training data
X_train_imputed = imputer.fit_transform(X_train)

# Transform the test data using the same imputer
X_test_imputed = imputer.transform(X_test)

# Step 2: Scale the data using StandardScaler
scaler = StandardScaler()

# Fit and transform the training data
X_train_scaled = scaler.fit_transform(X_train_imputed)

# Transform the test data using the same scaler
X_test_scaled = scaler.transform(X_test_imputed)

# Step 3: Train the logistic regression model
model = LogisticRegression()
```

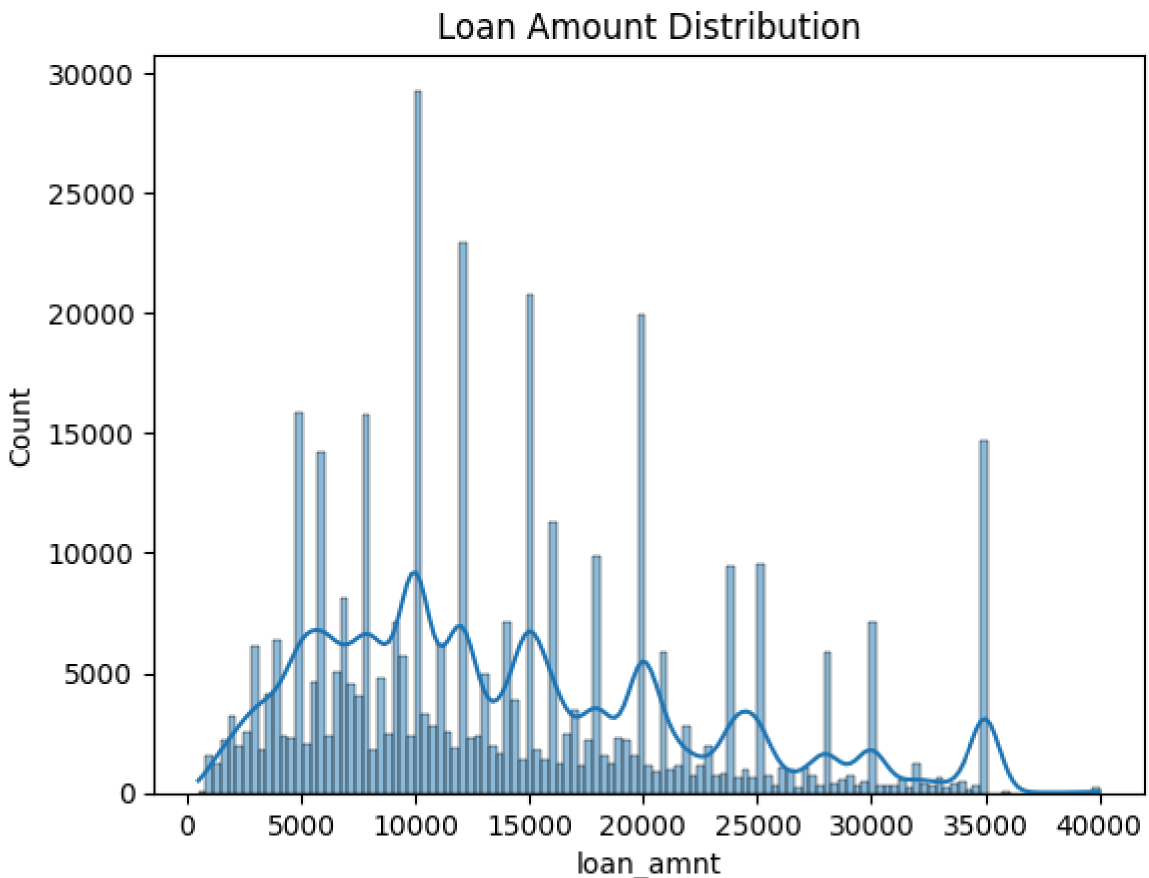
```
model.fit(X_train_scaled, y_train)

# Step 4: Predict on the test set
y_pred = model.predict(X_test_scaled)

# Output model performance (Optional: You can add accuracy or other metri
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')

# Evaluate the model
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
print(f"Confusion Matrix:\n {confusion_matrix(y_test, y_pred)}")
print(f"Classification Report:\n {classification_report(y_test, y_pred)}"

# Hyperparameter tuning with GridSearchCV
param_grid = {
    'C': [0.1, 1, 10],
    'penalty': ['l1', 'l2'],
}
grid_search = GridSearchCV(LogisticRegression(), param_grid, cv=5)
grid_search.fit(X_train_scaled, y_train)

# Best parameters found
print(f"Best Parameters: {grid_search.best_params_}")

# Logistic regression with balanced class weights
model_balanced = LogisticRegression(class_weight='balanced')
model_balanced.fit(X_train_scaled, y_train)

# Predict on the test set
y_pred_balanced = model_balanced.predict(X_test_scaled)

# Evaluate the model
print(f"Accuracy with balanced class weights: {accuracy_score(y_test, y_p
print(f"Confusion Matrix:\n {confusion_matrix(y_test, y_pred_balanced)}")

# Model Coefficients
# Store feature names before transformation to maintain them after scalin
feature_names = X.columns

coefficients = pd.DataFrame({
    'Feature': feature_names,
    'Coefficient': model_balanced.coef_[0]
})
coefficients = coefficients.sort_values(by='Coefficient', ascending=False
print(coefficients)

# ROC-AUC Curve
fpr, tpr, thresholds = roc_curve(y_test, model_balanced.predict_proba(X_t
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
```

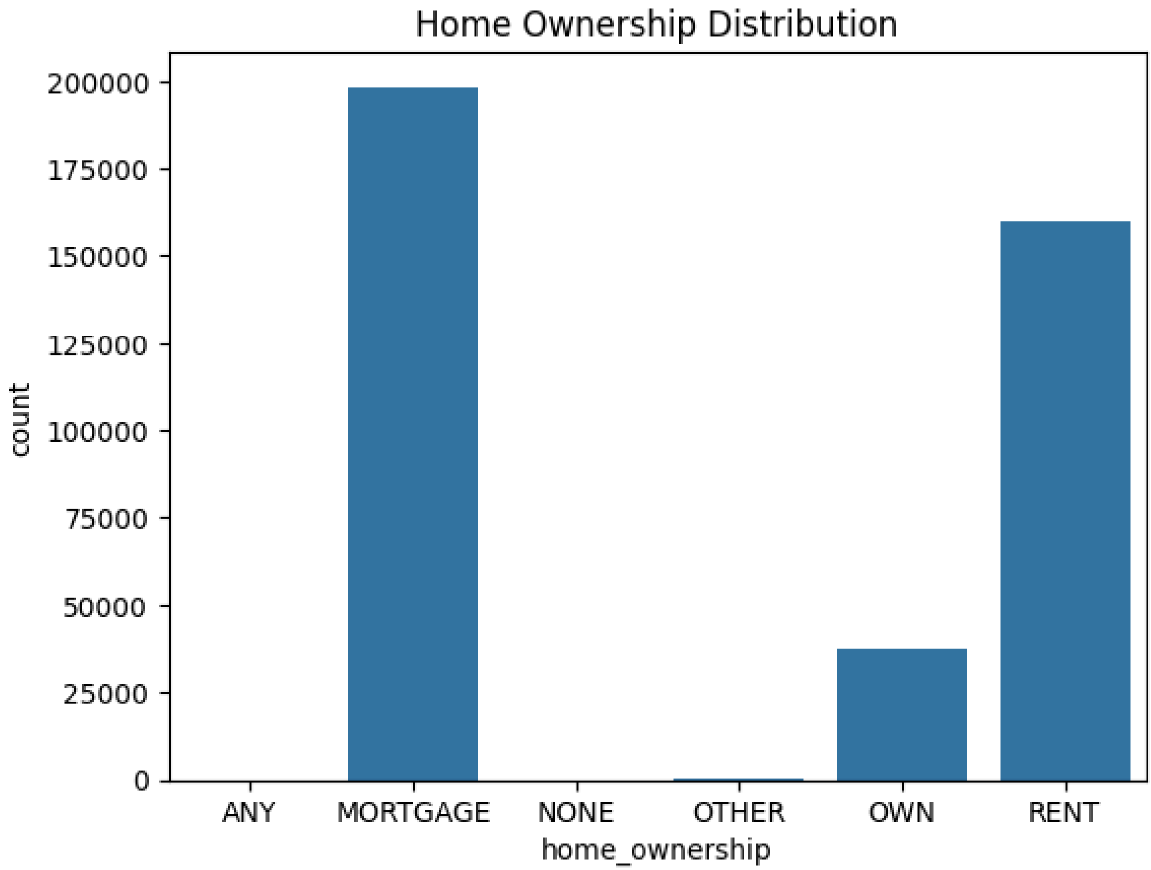
```
plt.show()

# Precision-Recall Curve
precision, recall, _ = precision_recall_curve(y_test, model_balanced.pred

plt.figure()
plt.plot(recall, precision, color='b', lw=2)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall curve')
plt.show()

# 1. Percentage of customers who fully paid their Loan Amount
fully_paid_percentage = (y.value_counts(normalize=True)[1]) * 100
print(f"Percentage of customers who fully paid their Loan Amount: {fully_

# 2. Correlation between Loan Amount and Installment
correlation = df['loan_amnt'].corr(df['installment'])
print(f"Correlation between Loan Amount and Installment: {correlation}")

# 3. Most common home ownership
most_common_home_ownership = df['home_ownership'].mode()[0]
print(f"The majority of people have home ownership as {most_common_home_o

# 4. People with grades 'A' are more likely to fully pay their loan. (T/F
grade_a_full_payment = df[df['grade'] == 'A']['loan_status'].value_counts
print(f"Grade A fully paid percentage: {grade_a_full_payment['Fully Paid'
```

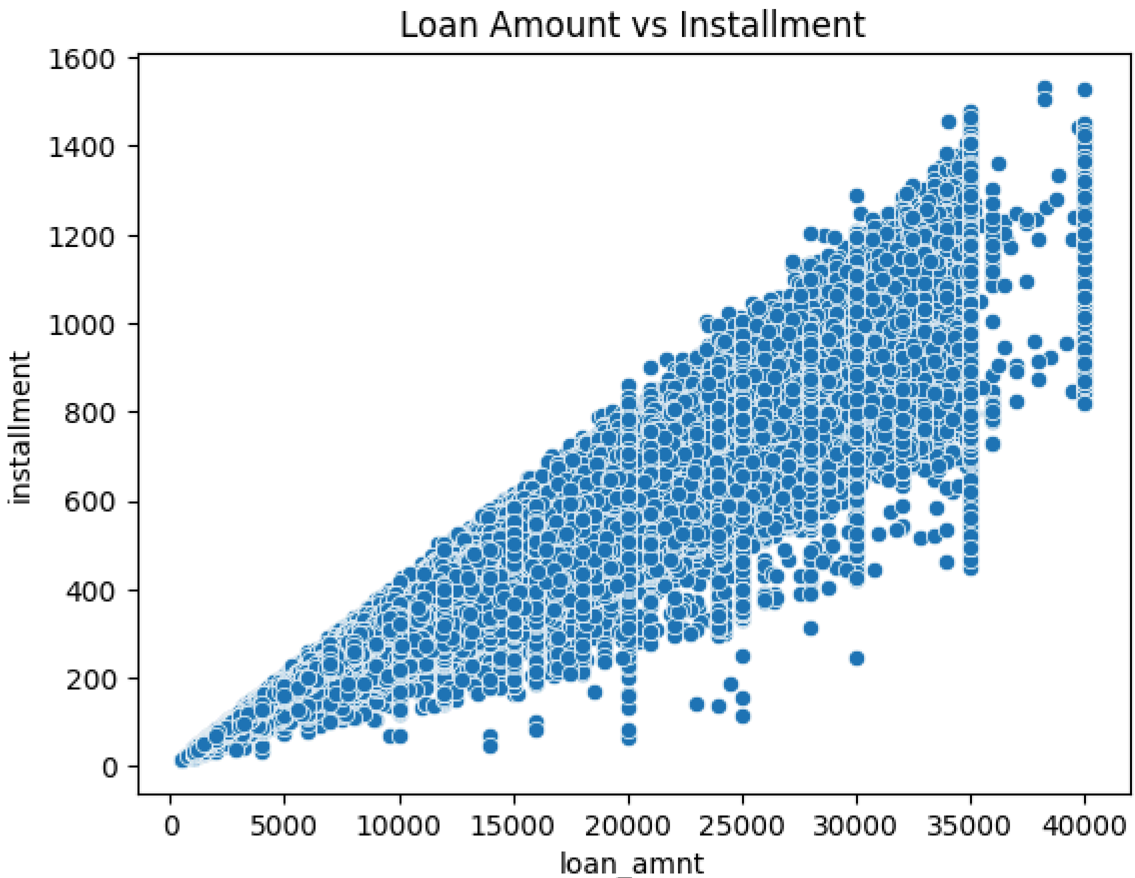
```
   loan_amnt        term  int_rate  installment grade sub_grade  \
0    10000.0   36 months     11.44       329.48     B        B4   
1     8000.0   36 months     11.99       265.68     B        B5   
2    15600.0   36 months     10.49       506.97     B        B3   
3     7200.0   36 months      6.49       220.65     A        A2   
4    24375.0   60 months     17.27       609.33     C        C5   

                 emp_title emp_length home_ownership  annual_inc  \
0                Marketing  10+ years           RENT    117000.0   
1          Credit analyst     4 years       MORTGAGE     65000.0   
2             Statistician   < 1 year           RENT     43057.0   
3           Client Advocate    6 years           RENT     54000.0   
4  Destiny Management Inc.    9 years       MORTGAGE     55000.0   

  verification_status   issue_d  loan_status             purpose  \
0        Not Verified  Jan-2015   Fully Paid            vacation   
1        Not Verified  Jan-2015   Fully Paid  debt_consolidation   
2     Source Verified  Jan-2015   Fully Paid         credit_card   
3        Not Verified  Nov-2014   Fully Paid         credit_card   
4            Verified  Apr-2013  Charged Off         credit_card   

                     title    dti earliest_cr_line  open_acc  pub_rec  \
0                 Vacation  26.24         Jun-1990      16.0      0.0   
1       Debt consolidation  22.05         Jul-2004      17.0      0.0   
2  Credit card refinancing  12.79         Aug-2007      13.0      0.0   
3  Credit card refinancing   2.60         Sep-2006       6.0      0.0   
4    Credit Card Refinance  33.95         Mar-1999      13.0      0.0   

   revol_bal  revol_util  total_acc initial_list_status application_type 
\
0    36369.0        41.8       25.0                   w       INDIVIDUAL   
1    20131.0        53.3       27.0                   f       INDIVIDUAL   
2    11987.0        92.2       26.0                   f       INDIVIDUAL   
3     5472.0        21.5       13.0                   f       INDIVIDUAL   
4    24584.0        69.8       43.0                   f       INDIVIDUAL   

   mort_acc  pub_rec_bankruptcies  \
0       0.0                   0.0   
1       3.0                   0.0   
2       0.0                   0.0   
3       0.0                   0.0   
4       1.0                   0.0   

                                              address  
0     0174 Michelle Gateway\r\nMendozaberg, OK 22690  
1  1076 Carney Fort Apt. 347\r\nLoganmouth, SD 05113  
2  87025 Mark Dale Apt. 269\r\nNew Sabrina, WV 05113  
3            823 Reid Ford\r\nDelacruzside, MA 00813  
4             679 Luna Roads\r\nGreggshire, VA 11650  
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  object 
 2   int_rate              396030 non-null  float64
 3   installment           396030 non-null  float64
 4   grade                 396030 non-null  object 
 5   sub_grade             396030 non-null  object 
```

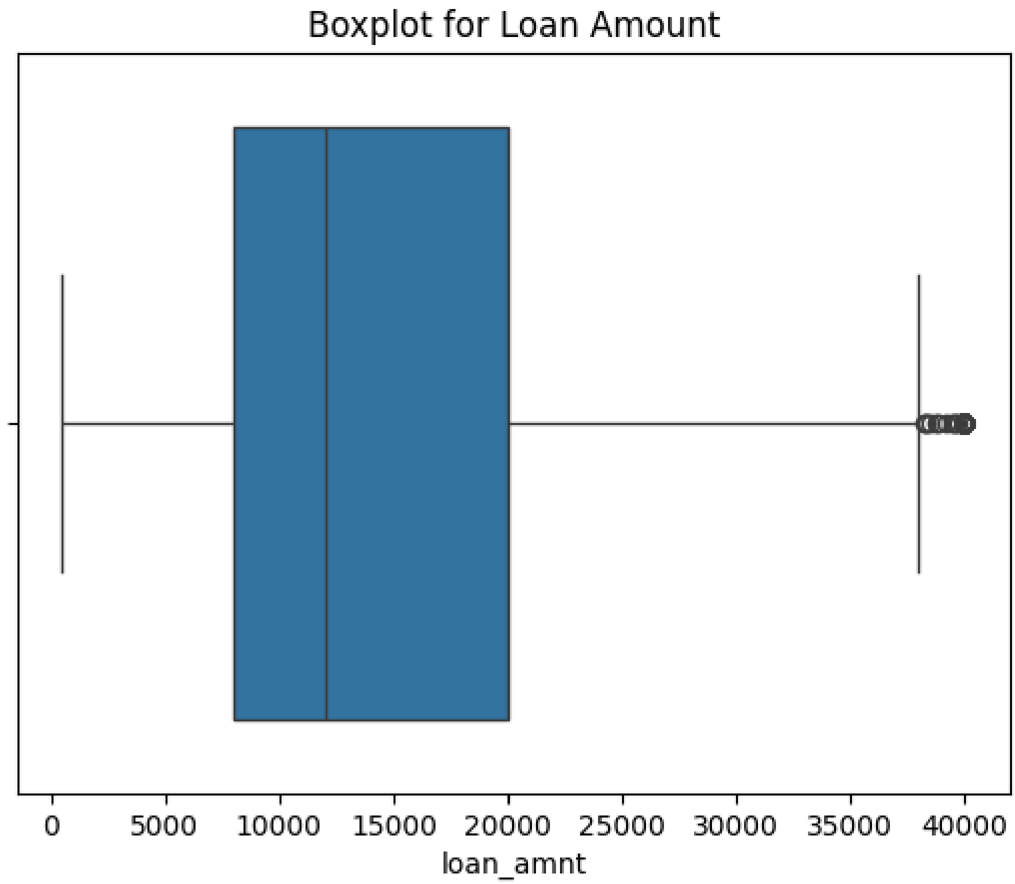
```
 6   emp_title             373103 non-null  object 
 7   emp_length            377729 non-null  object 
 8   home_ownership        396030 non-null  object 
 9   annual_inc            396030 non-null  float64
 10  verification_status   396030 non-null  object 
 11  issue_d               396030 non-null  object 
 12  loan_status           396030 non-null  object 
 13  purpose               396030 non-null  object 
 14  title                 394274 non-null  object 
 15  dti                   396030 non-null  float64
 16  earliest_cr_line      396030 non-null  object 
 17  open_acc              396030 non-null  float64
 18  pub_rec               396030 non-null  float64
 19  revol_bal             396030 non-null  float64
 20  revol_util            395754 non-null  float64
 21  total_acc             396030 non-null  float64
 22  initial_list_status   396030 non-null  object 
 23  application_type      396030 non-null  object 
 24  mort_acc              358235 non-null  float64
 25  pub_rec_bankruptcies  395495 non-null  float64
 26  address               396030 non-null  object 
dtypes: float64(12), object(15)
memory usage: 81.6+ MB
None
           loan_amnt       int_rate    installment    annual_inc  \
count  396030.000000  396030.000000  396030.000000  3.960300e+05   
mean    14113.888089      13.639400     431.849698  7.420318e+04   
std      8357.441341       4.472157     250.727790  6.163762e+04   
min       500.000000       5.320000      16.080000  0.000000e+00   
25%      8000.000000      10.490000     250.330000  4.500000e+04   
50%     12000.000000      13.330000     375.430000  6.400000e+04   
75%     20000.000000      16.490000     567.300000  9.000000e+04   
max     40000.000000      30.990000    1533.810000  8.706582e+06   

                 dti       open_acc        pub_rec     revol_bal  \
count  396030.000000  396030.000000  396030.000000  3.960300e+05   
mean       17.379514      11.311153       0.178191  1.584454e+04   
std        18.019092       5.137649       0.530671  2.059184e+04   
min         0.000000       0.000000       0.000000  0.000000e+00   
25%        11.280000       8.000000       0.000000  6.025000e+03   
50%        16.910000      10.000000       0.000000  1.118100e+04   
75%        22.980000      14.000000       0.000000  1.962000e+04   
max      9999.000000      90.000000      86.000000  1.743266e+06   

          revol_util      total_acc       mort_acc  pub_rec_bankruptcies  
count  395754.000000  396030.000000  358235.000000         395495.000000  
mean       53.791749      25.414744       1.813991              0.121648  
std        24.452193      11.886991       2.147930              0.356174  
min         0.000000       2.000000       0.000000              0.000000  
25%        35.800000      17.000000       0.000000              0.000000  
50%        54.800000      24.000000       1.000000              0.000000  
75%        72.900000      32.000000       3.000000              0.000000  
max       892.300000     151.000000      34.000000              8.000000  
loan_amnt                   0
term                        0
int_rate                    0
installment                 0
grade                       0
sub_grade                   0
emp_title               22927
```

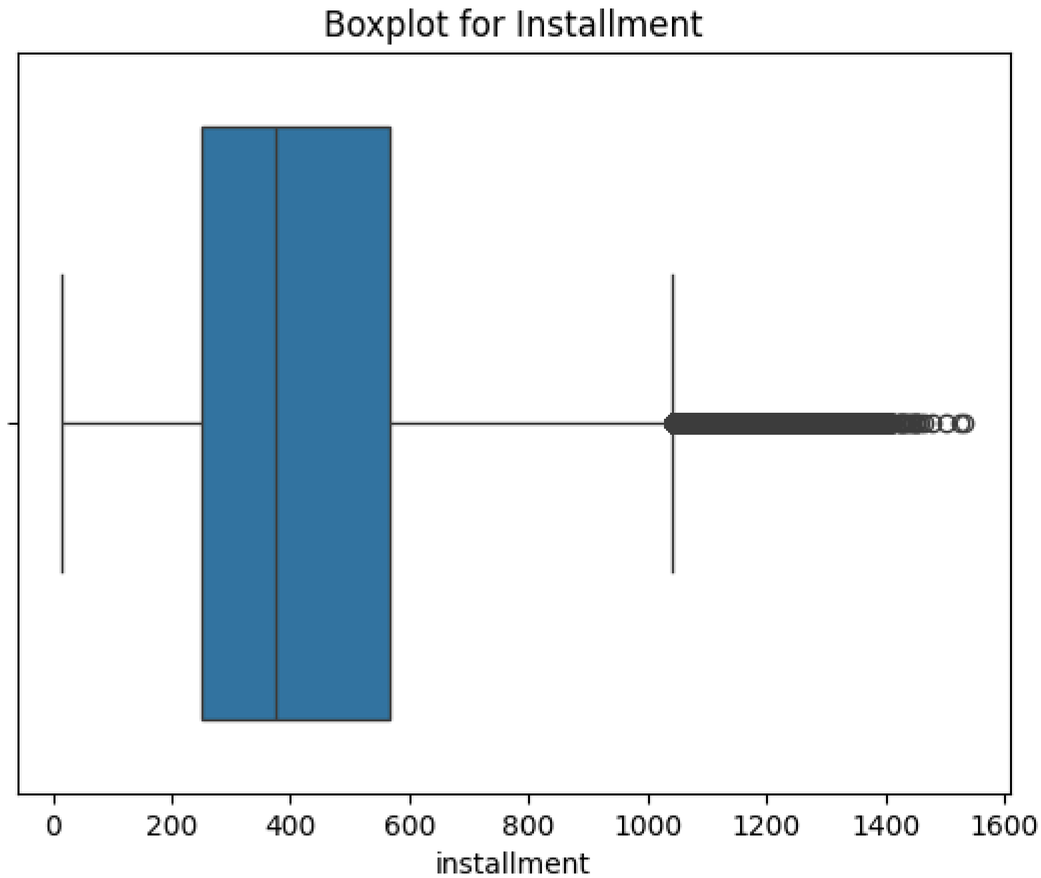
```
emp_length              18301
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
loan_status                 0
purpose                     0
title                    1756
dti                         0
earliest_cr_line            0
open_acc                    0
pub_rec                     0
revol_bal                   0
revol_util                276
total_acc                   0
initial_list_status         0
application_type            0
mort_acc                37795
pub_rec_bankruptcies      535
address                     0
dtype: int64
```




Home Ownership Distribution
200000
175000
150000
125000
100000
75000
50000
25000
0
count
ANY
MORTGAGE
NONE
OTHER
OWN
RENT
home_ownership


Loan Amount vs Installment
1600
1400
1200
1000
800
600
400
200
0
installment
0
5000
10000
15000
20000
25000
30000
35000
40000
loan_amnt



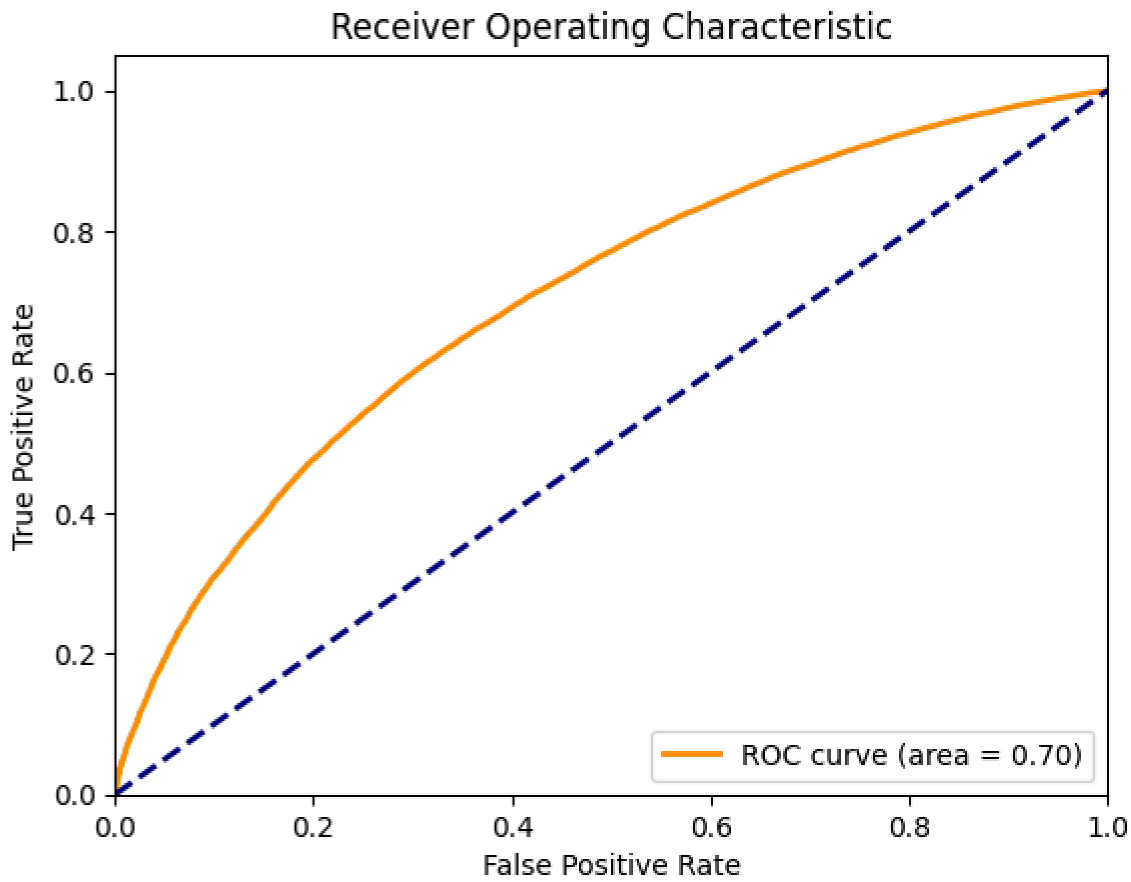


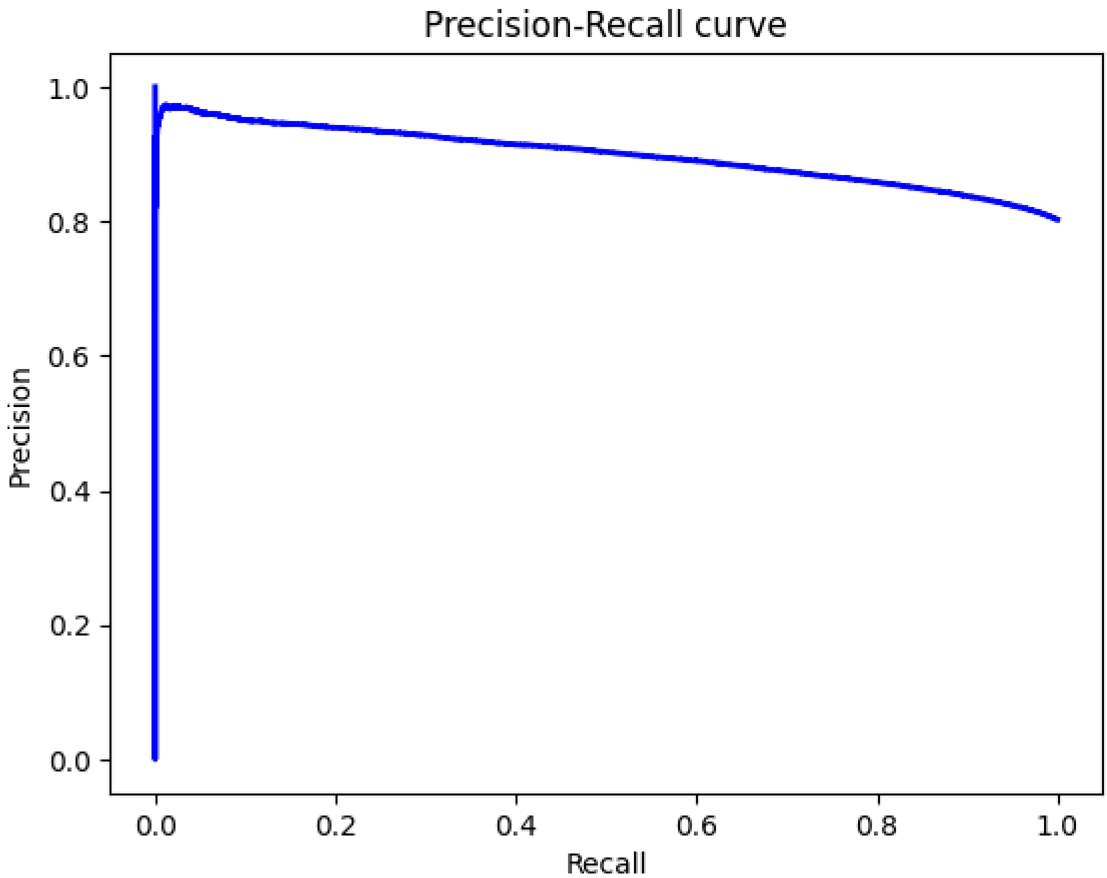

```
Duplicate rows: 0
```

```
/var/folders/c8/n9hz87597yz68gbmzzks3v_00000gn/T/ipykernel_43383/218636584
6.py:69: FutureWarning: The behavior of Series.replace (and DataFrame.repl
ace) with CategoricalDtype is deprecated. In a future version, replace wil
l only be used for cases that preserve the categories. To change the categ
ories, use ser.cat.rename_categories instead.
  df['emp_length'] = df['emp_length'].replace({'10+ years': '10', ' year
s': '', ' month': '', ' months': ''}, regex=True)
/var/folders/c8/n9hz87597yz68gbmzzks3v_00000gn/T/ipykernel_43383/218636584
6.py:87: UserWarning: Could not infer format, so each element will be pars
ed individually, falling back to `dateutil`. To ensure parsing is consiste
nt and as-expected, please specify a format.
  df['issue_month'] = pd.to_datetime(df['issue_d'], errors='coerce').dt.mo
nth
/var/folders/c8/n9hz87597yz68gbmzzks3v_00000gn/T/ipykernel_43383/218636584
6.py:88: UserWarning: Could not infer format, so each element will be pars
ed individually, falling back to `dateutil`. To ensure parsing is consiste
nt and as-expected, please specify a format.
  df['issue_year'] = pd.to_datetime(df['issue_d'], errors='coerce').dt.yea
r
Accuracy: 0.8033380490433846
Accuracy: 0.8033380490433846
Confusion Matrix:
 [[ 1523 21932]
 [ 1422 93875]]
Classification Report:
               precision    recall  f1-score   support

           0       0.52      0.06      0.12     23455
           1       0.81      0.99      0.89     95297

    accuracy                           0.80    118752
   macro avg       0.66      0.53      0.50    118752
weighted avg       0.75      0.80      0.74    118752
```

```
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-pac
kages/sklearn/model_selection/_validation.py:528: FitFailedWarning:
15 fits failed out of a total of 30.
The score on these train-test partitions for these parameters will be set
to nan.
If these failures are not expected, you can try to debug them by setting e
rror_score='raise'.

Below are more details about the failures:
---------------------------------------------------------------------------
------
15 fits failed with the following error:
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/
site-packages/sklearn/model_selection/_validation.py", line 866, in _fit_a
nd_score
    estimator.fit(X_train, y_train, **fit_params)
    ~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/
site-packages/sklearn/base.py", line 1389, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/
site-packages/sklearn/linear_model/_logistic.py", line 1193, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/
site-packages/sklearn/linear_model/_logistic.py", line 63, in _check_solve
r
    raise ValueError(
    ...<2 lines>...
    )
ValueError: Solver lbfgs supports only 'l2' or None penalties, got l1 pena
lty.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-pac
kages/sklearn/model_selection/_search.py:1107: UserWarning: One or more of
the test scores are non-finite: [       nan 0.80495657        nan 0.804952
96        nan 0.80494935]
  warnings.warn(
```

```
Best Parameters: {'C': 0.1, 'penalty': 'l2'}
Accuracy with balanced class weights: 0.6575889248181084
Confusion Matrix:
 [[14822  8633]
 [32029 63268]]
                 Feature  Coefficient
5             annual_inc     0.165402
12              mort_acc     0.115370
11             total_acc     0.095639
9              revol_bal     0.045879
14             high_loan     0.035784
0              loan_amnt     0.027683
15           issue_month     0.024780
13  pub_rec_bankruptcies     0.018690
4             emp_length     0.000000
8                pub_rec    -0.054458
10            revol_util    -0.074484
7               open_acc    -0.082104
16            issue_year    -0.089707
3            installment    -0.105067
1                   term    -0.223701
2               int_rate    -0.475024
6                    dti    -0.487046
```






```
Percentage of customers who fully paid their Loan Amount: 80.3799019298250
1%
Correlation between Loan Amount and Installment: 0.9538590568066893
The majority of people have home ownership as MORTGAGE.
Grade A fully paid percentage: 0.937116134060795
```

In [ ]: