Predicting loan default is a critical task for financial institutions to mitigate risk and ensure profitability. In this project, we will develop a complex machine learning model using Random Forest to predict the loan status in a given test dataset. We will cover every aspect of the Random Forest algorithm and delve deep into data preprocessing, feature engineering, model tuning, and evaluation.

Table of Contents

Step1: Introduction

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the class that is the mode of the classes of individual trees. It is known for its robustness, ability to handle high-dimensional data, and effectiveness in preventing overfitting.

Objective: Build a Random Forest model to predict the loan status (Loan_Status) in the test dataset.

Step2: Dataset Overview

Loan_ID: Unique Loan ID

Gender: Male/Female

Married: Applicant married (Y/N)

Dependents: Number of dependents

Education: Applicant Education (Graduate/Undergraduate)

Self_Employed: Self-employed (Y/N)

ApplicantIncome: Applicant income

CoapplicantIncome: Coapplicant income

LoanAmount: Loan amount in thousands

Loan_Amount_Term: Term of loan in months

Credit_History: Credit history meets guidelines

Property_Area: Urban/Semi-Urban/Rural

Loan_Status: Loan approved (Y/N)

Step3: EDA

```
#pip install imblearn
```

3.1 Library import

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

# For modeling
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV,
RandomizedSearchCV, cross_val_score
from sklearn.metrics import classification_report, confusion_matrix,
roc_auc_score, roc_curve, accuracy_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from imblearn.over_sampling import SMOTE
```

3.2 Load Data

```
ls

data = pd.read_csv('loan_data.csv')
```

3.3 Data Overview

```
data.head() # top five elements/rows
```

```
   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  ...


  open_acc pub_rec revol_bal revol_util total_acc  initial_list_status
\
0     16.0     0.0   36369.0       41.8      25.0                    w

1     17.0     0.0   20131.0       53.3      27.0                    f

2     13.0     0.0   11987.0       92.2      26.0                    f

3      6.0     0.0    5472.0       21.5      13.0                    f

4     13.0     0.0   24584.0       69.8      43.0                    f


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

                                              address  
0     0174 Michelle Gateway\nMendozaberg, OK 22690  
1  1076 Carney Fort Apt. 347\nLoganmouth, SD 05113  
2  87025 Mark Dale Apt. 269\nNew Sabrina, WV 05113  
3            823 Reid Ford\nDelacruzside, MA 00813  
4             679 Luna Roads\nGreggshire, VA 11650  

[5 rows x 27 columns]

data.info()
```

```
<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                 394275 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
```

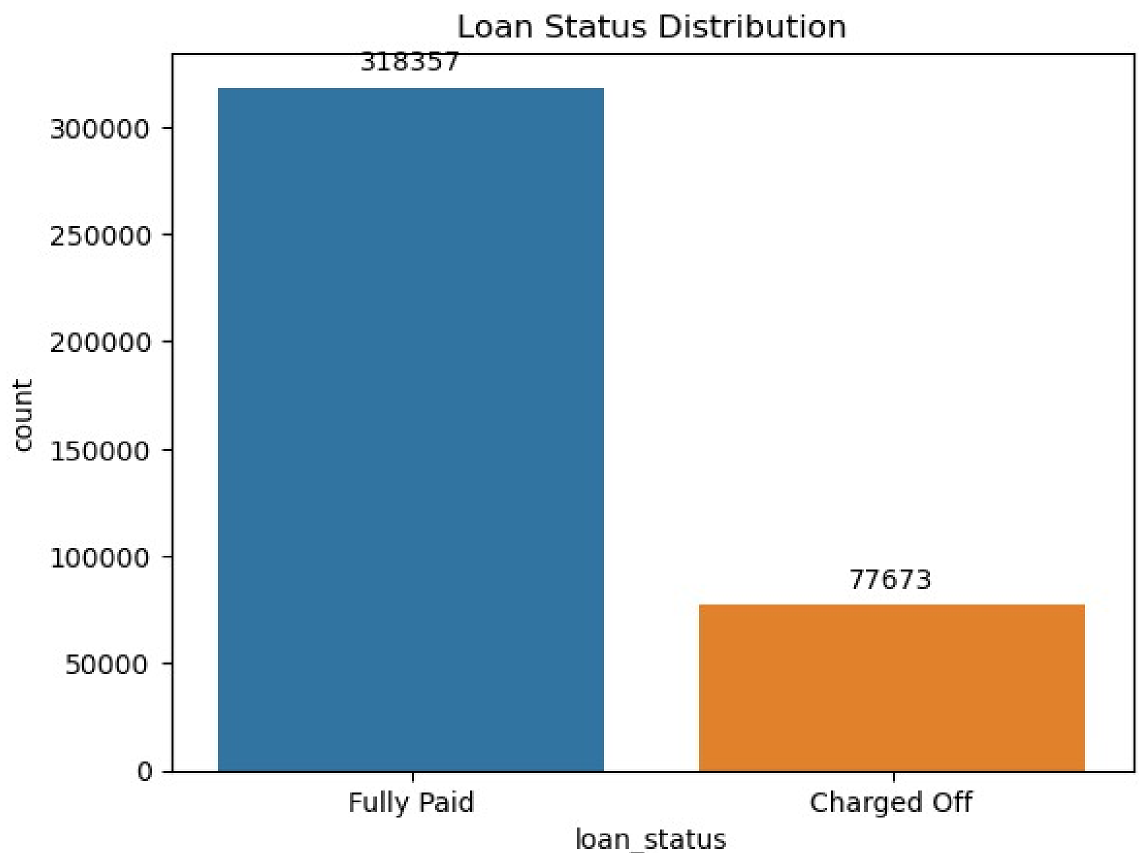
```
data.describe()
```

```
           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
```

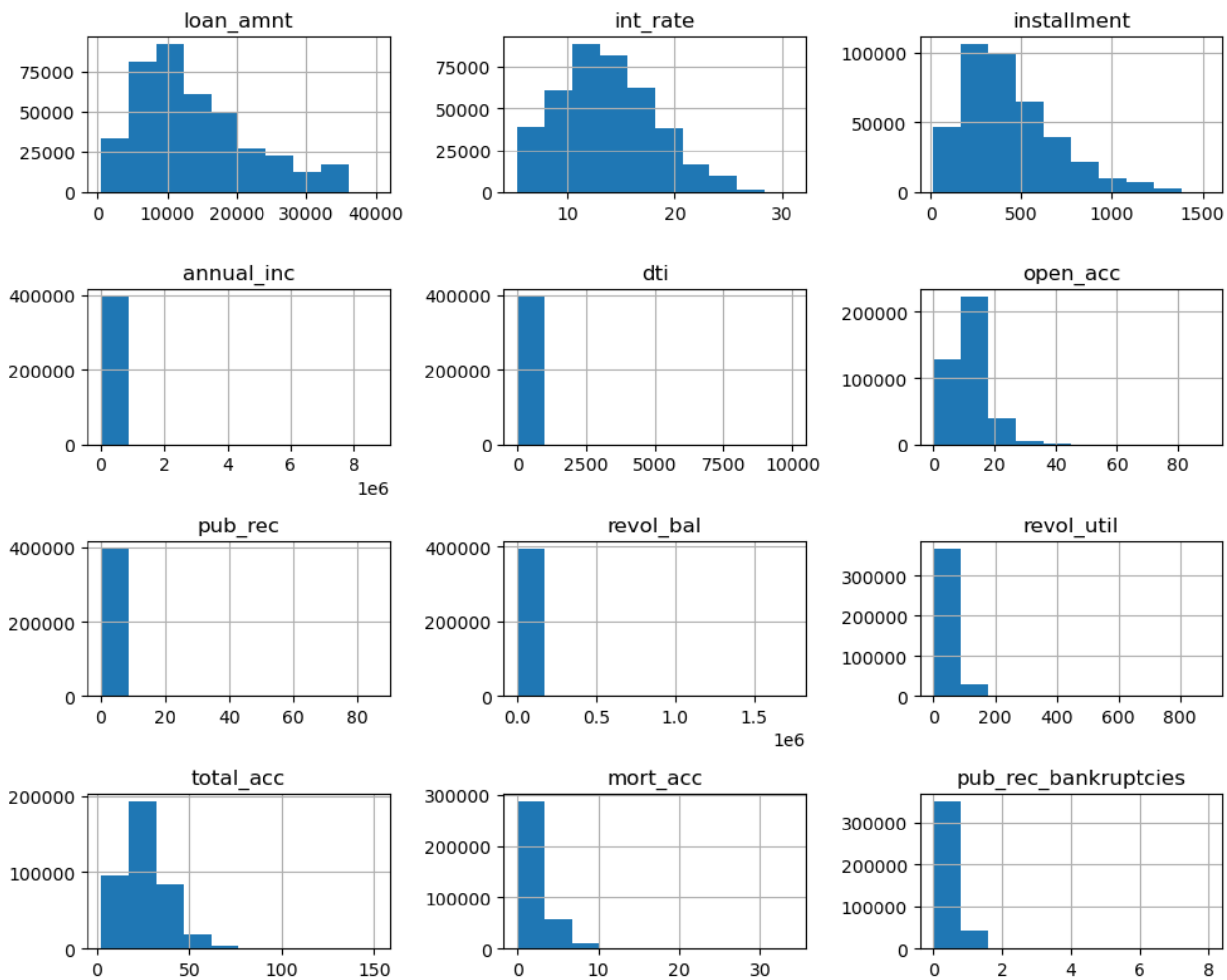
3.4 Univariate Analysis

```python
sns.countplot(x='loan_status', data=data)
plt.title('Loan Status Distribution')
# Get the current axis (gca) and iterate over the bars to add the text
on top
ax = plt.gca()
for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}',
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', xytext=(0, 9),
textcoords='offset points')

plt.show()
```



3.4.1 Numerical Features Distribution

```
cat_columns = data.select_dtypes(include=['object',
'category']).columns.tolist()
num_columns = data.select_dtypes(include=['int64',
'float64']).columns.tolist()

data[num_columns].hist(figsize=(10, 8))
plt.tight_layout()
plt.show()
```



3.4.2 Categorical Features Distribution

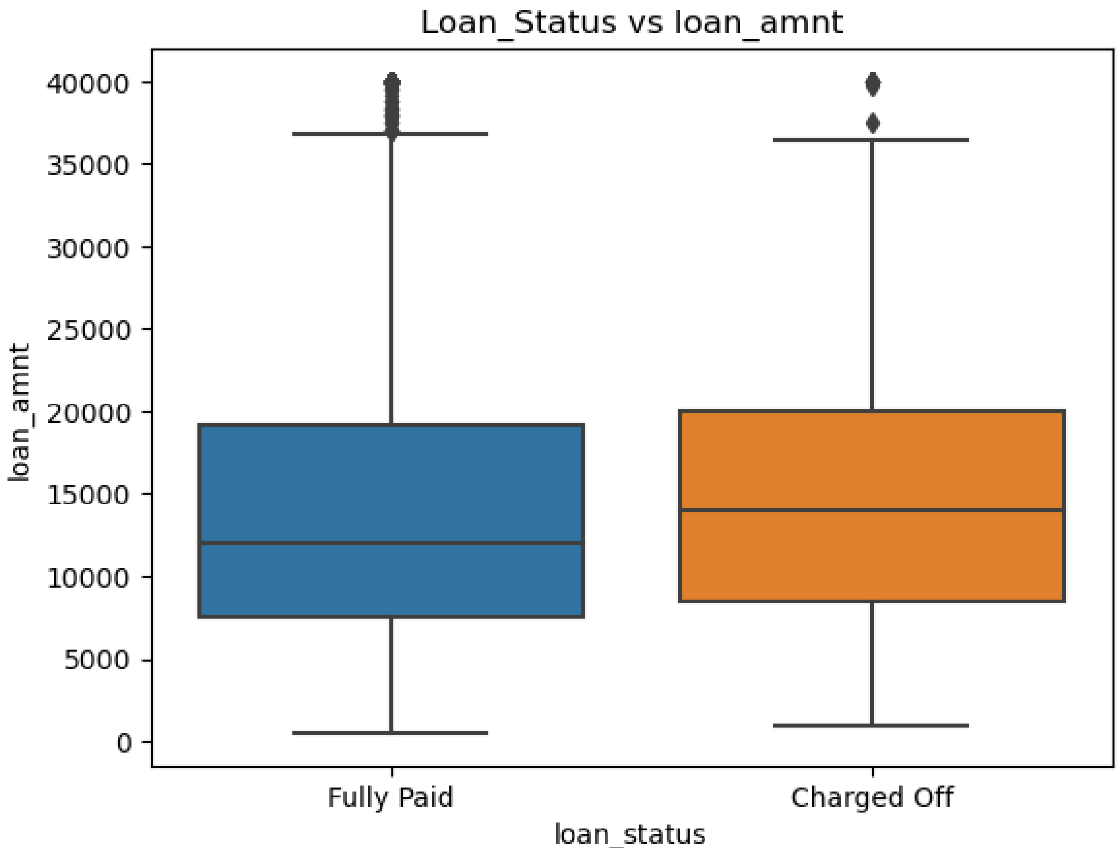
```
# this might take a bit of time
'''
for feature in cat_columns:
    sns.countplot(x=feature, data=data)
    plt.title(f'{feature} Distribution')
    plt.show()
'''

"\nfor feature in cat_columns:\n    sns.countplot(x=feature,
data=data)\n    plt.title(f'{feature} Distribution')\n    plt.show()\
n"
```

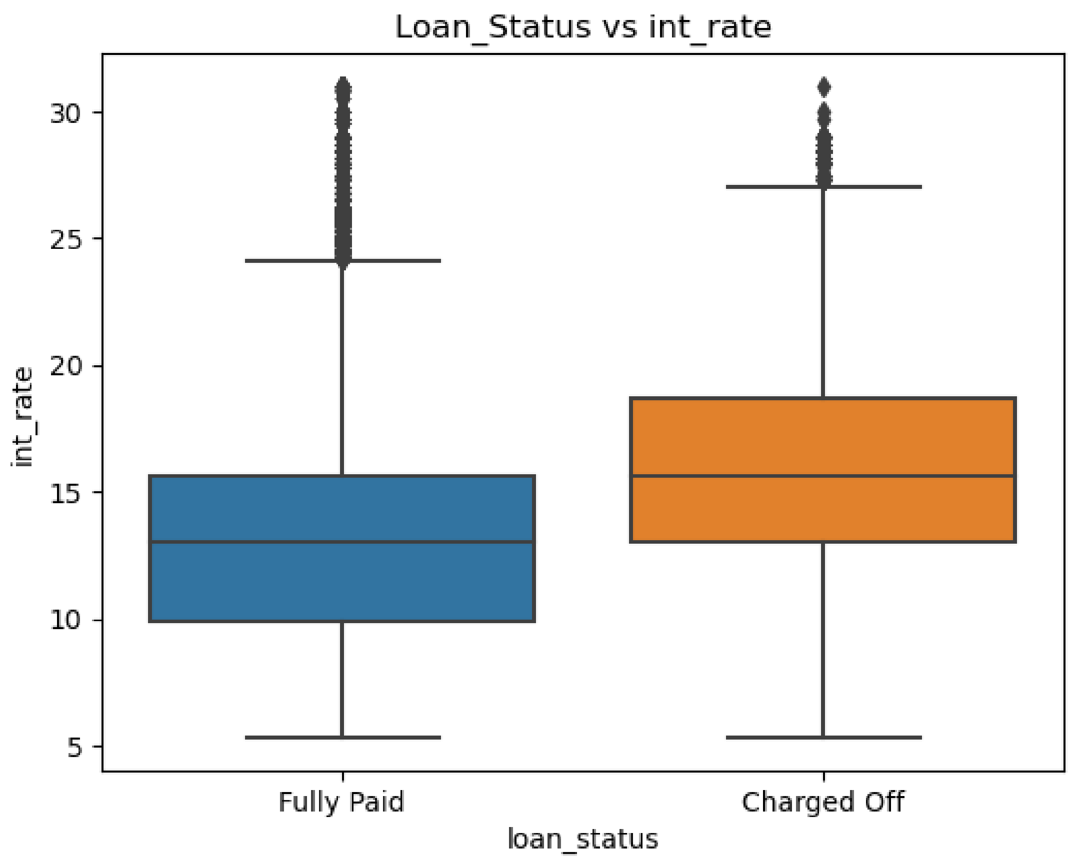
3.5 Bivariate Analysis

3.5.1 Loan_Status vs Numerical Features

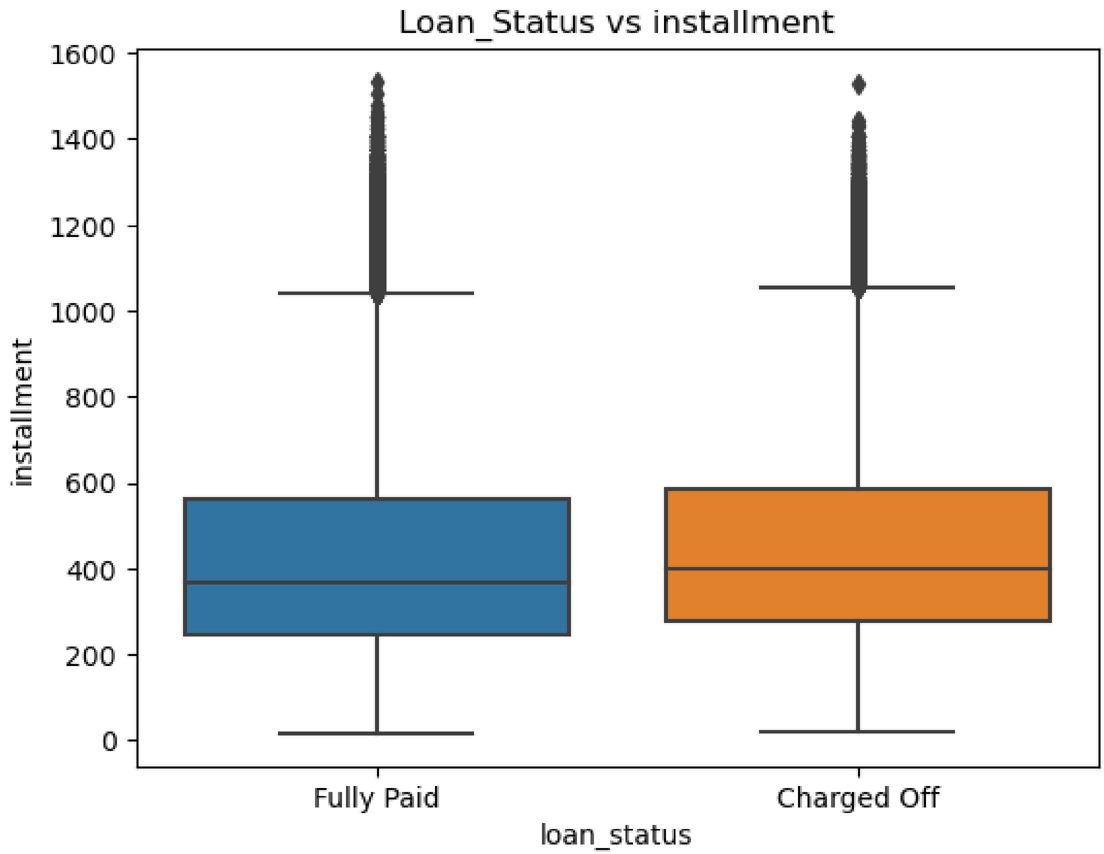
```
for feature in num_columns:
    sns.boxplot(x='loan_status', y=feature, data=data)
```

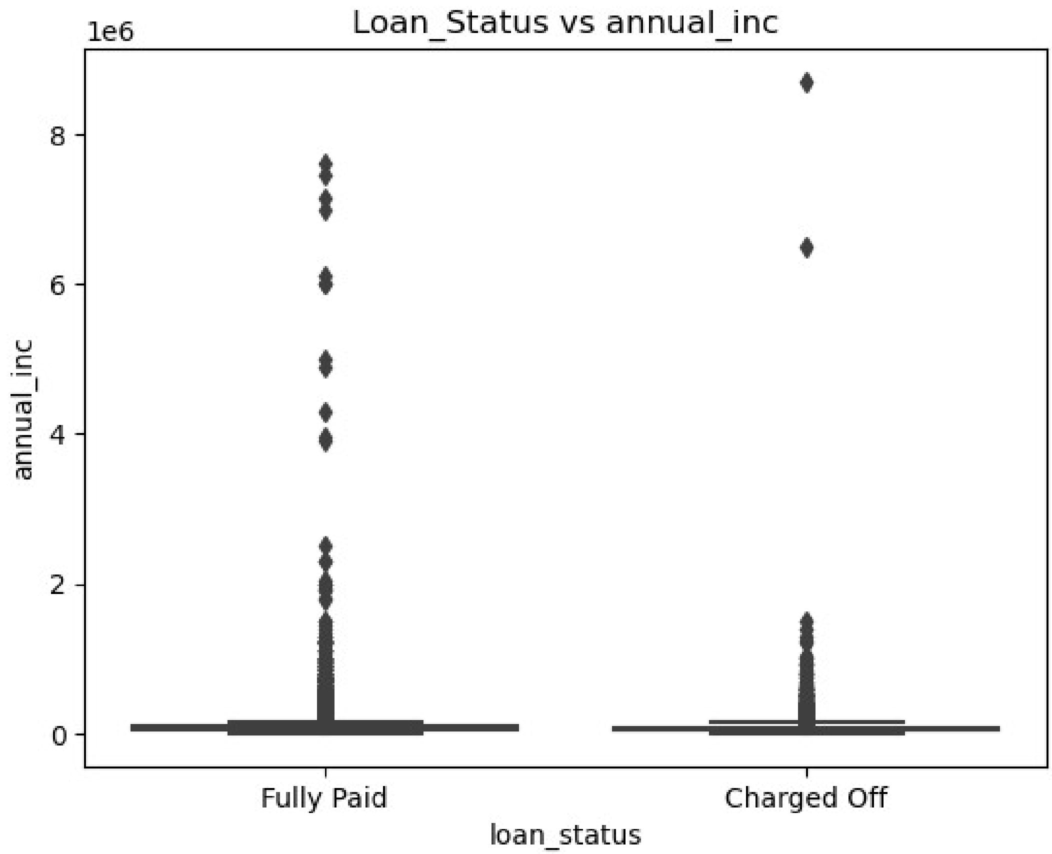
```
    plt.title(f'Loan_Status vs {feature}')
    plt.show()
```



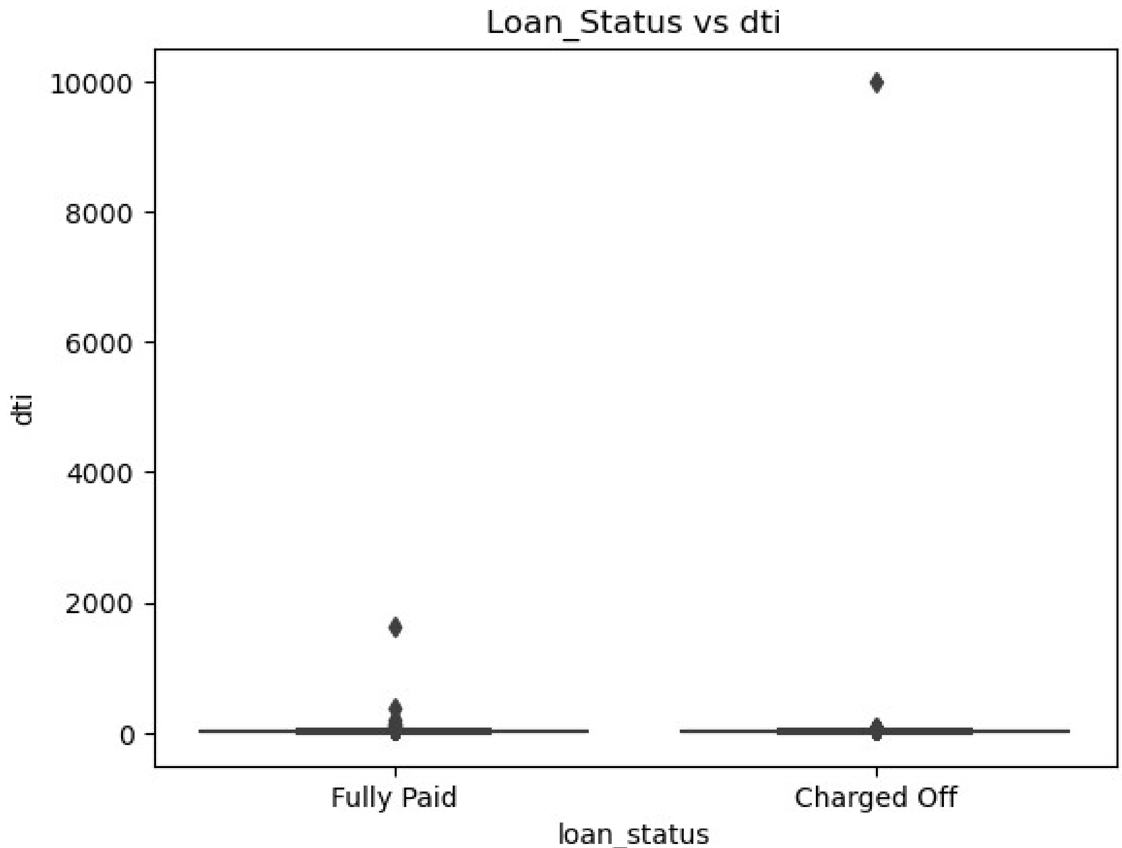


Loan_Status vs int_rate
30
25
20
15
10
5
int_rate
Fully Paid
Charged Off
loan_status

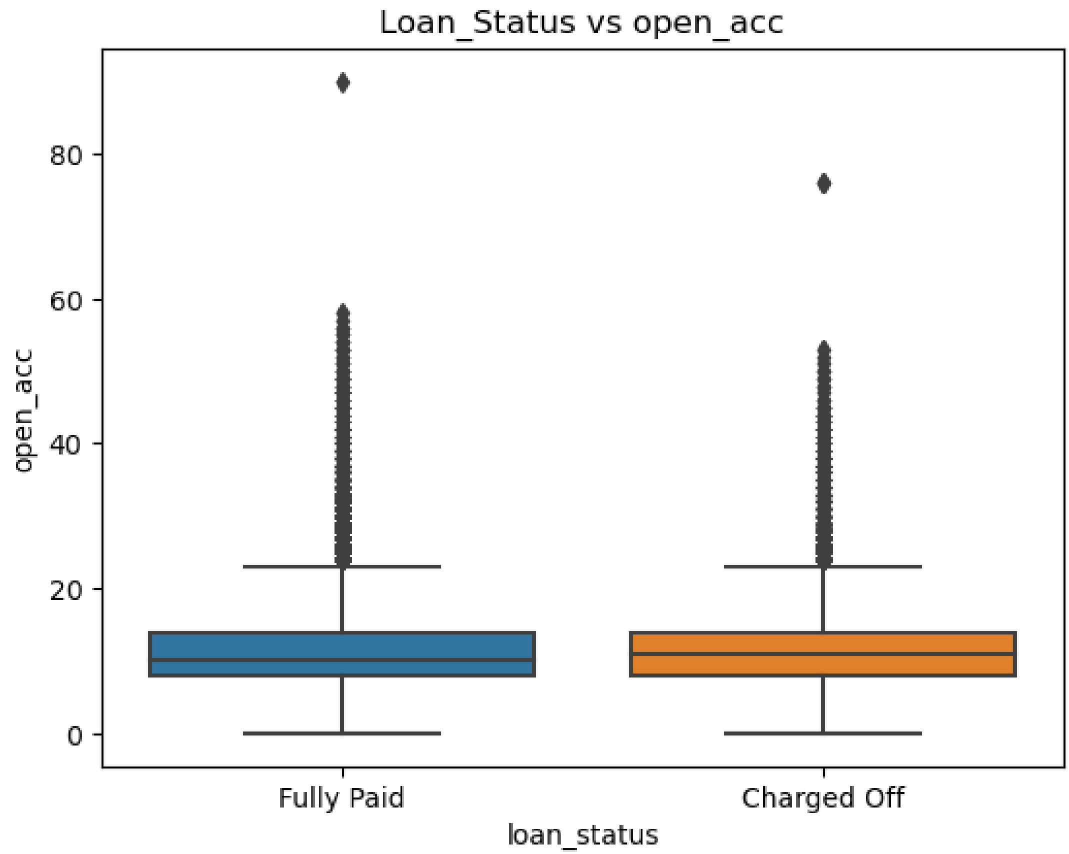

Loan_Status vs installment
1600
1400
1200
1000
800
600
400
200
0
installment
Fully Paid
Charged Off
loan_status

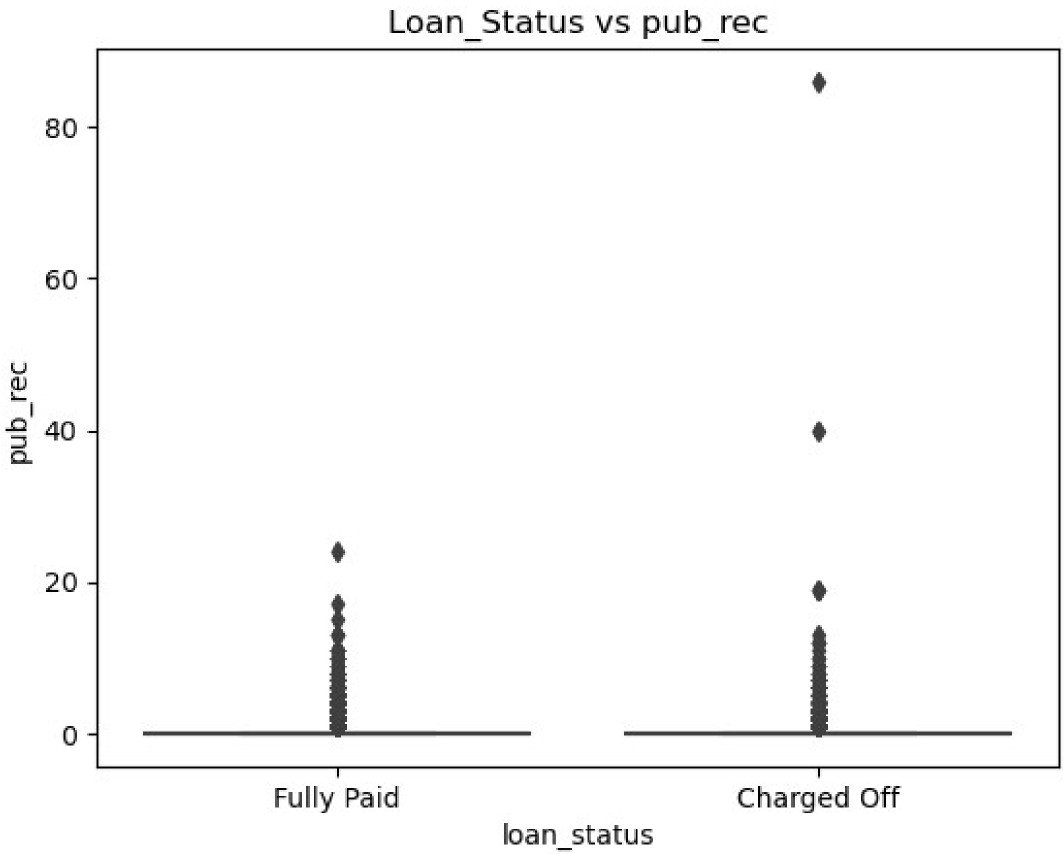


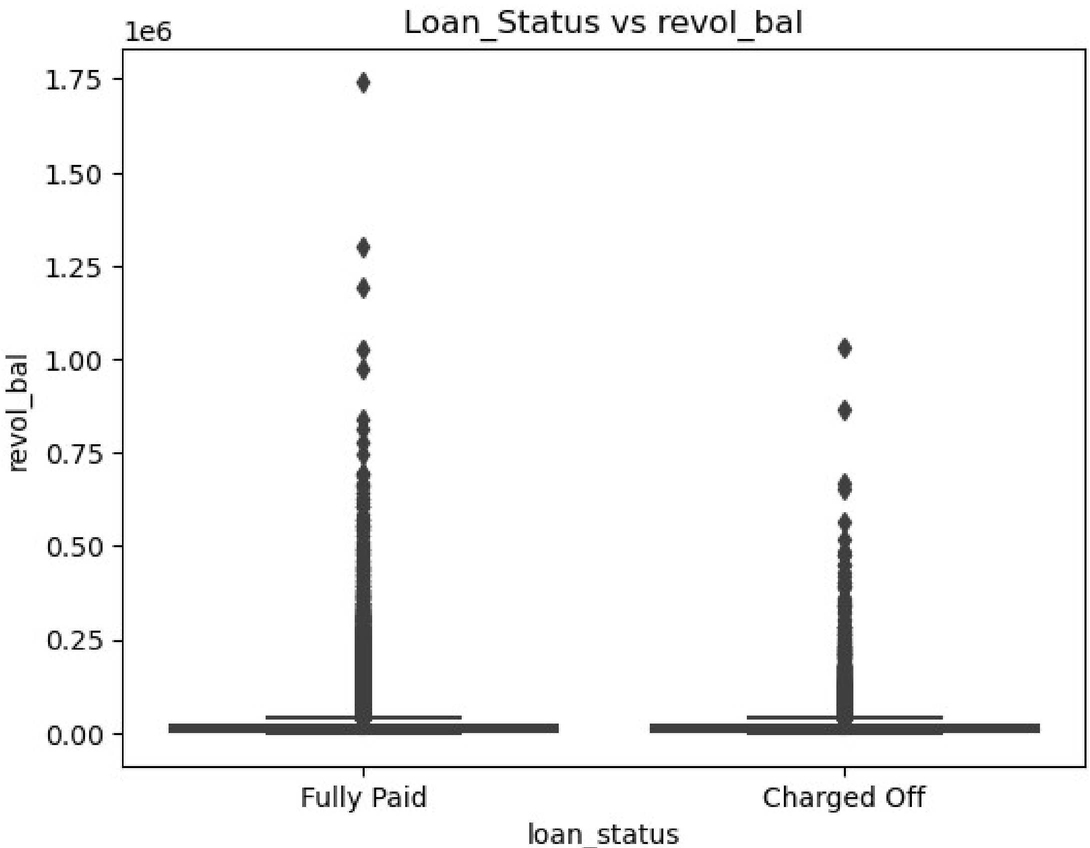
Loan_Status vs annual_inc
1e6
8
6
4
2
0
annual_inc
Fully Paid
Charged Off
loan_status

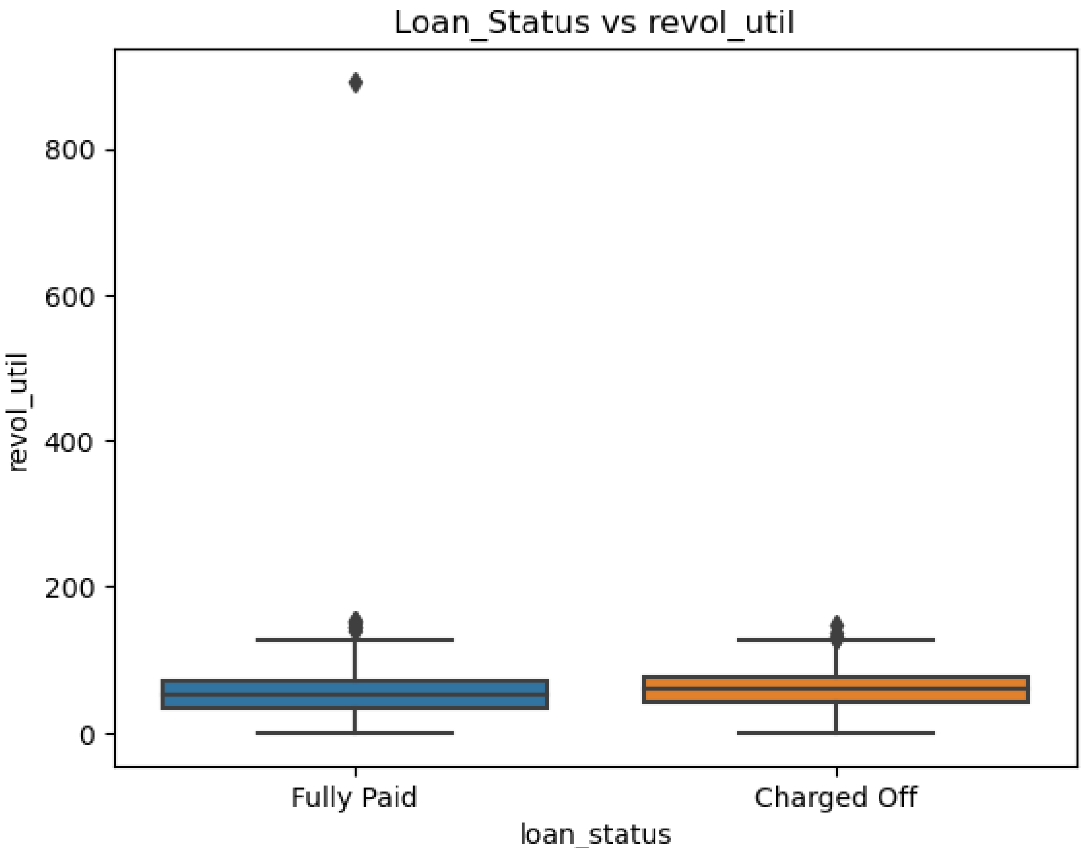


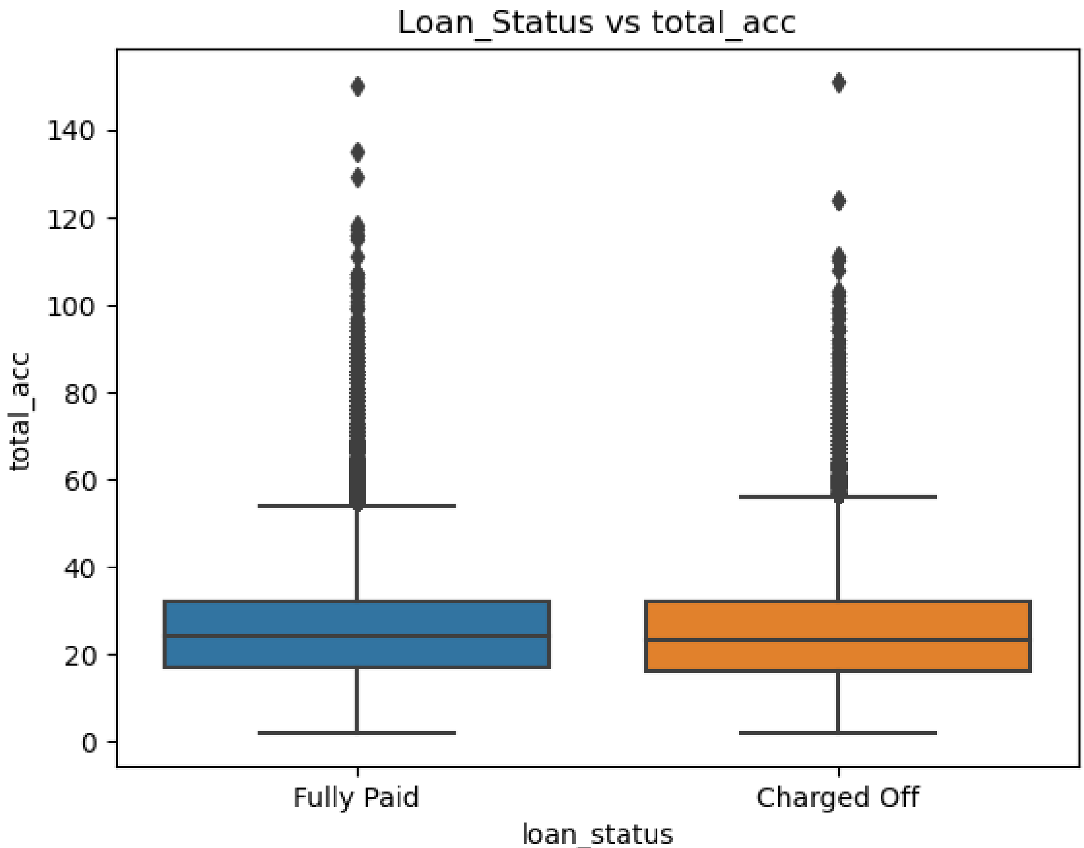
Loan_Status vs dti
10000
8000
6000
4000
2000
0
dti
Fully Paid
Charged Off
loan_status


Loan_Status vs open_acc
open_acc
80
60
40
20
0
Fully Paid
Charged Off
loan_status

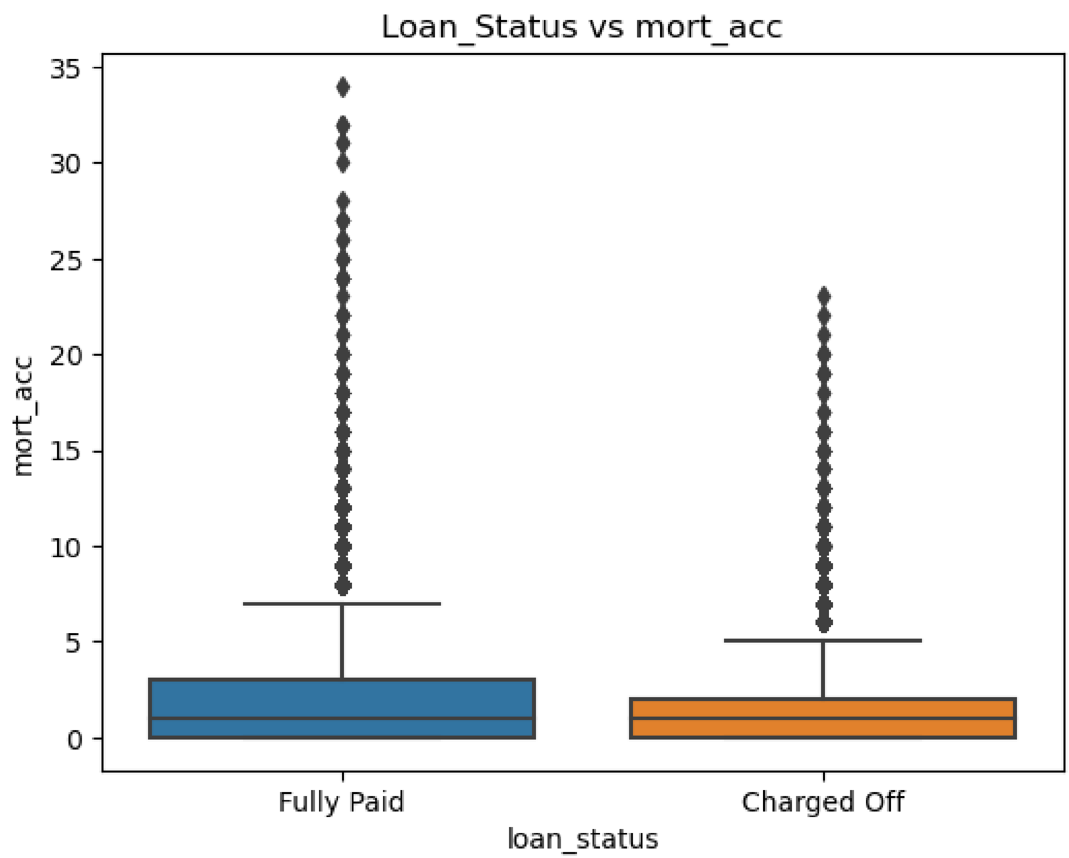

Loan_Status vs pub_rec
80
60
40
20
0
pub_rec
Fully Paid
Charged Off
loan_status

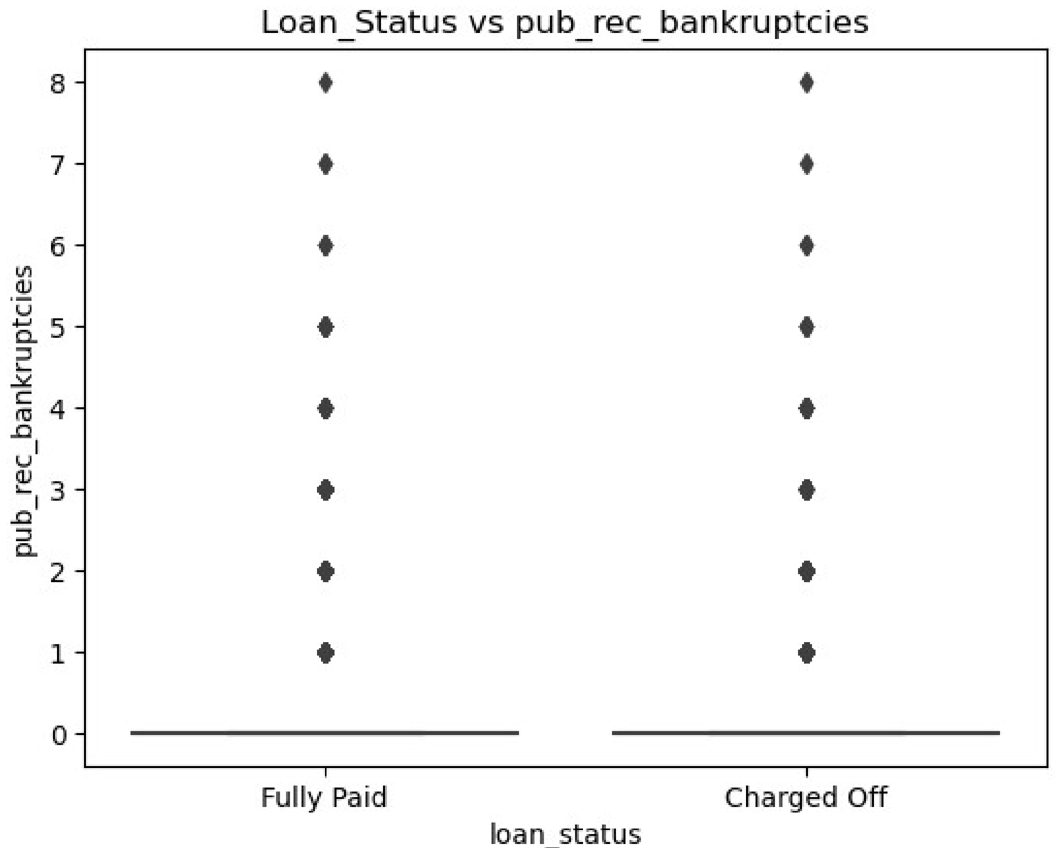

Loan_Status vs revol_bal
1e6
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
revol_bal
Fully Paid
Charged Off
loan_status


Loan_Status vs revol_util
800
600
400
200
0
revol_util
Fully Paid
Charged Off
loan_status



Loan_Status vs total_acc
140
120
100
80
60
40
20
0
total_acc
Fully Paid
Charged Off
loan_status


Loan_Status vs mort_acc
35
30
25
20
15
10
5
0
mort_acc
Fully Paid
Charged Off
loan_status



3.5.2 Loan_Status vs Categorical Features

```
# this also might take a bit of time

'''

for feature in cat_columns:
    sns.countplot(x=feature, hue='loan_status', data=data)
    plt.title(f'Loan_Status vs {feature}')
    plt.show()
'''

"\n\nfor feature in cat_columns:\n    sns.countplot(x=feature,
hue='loan_status', data=data)\n    plt.title(f'Loan_Status vs
{feature}')\n    plt.show()\n"
```

Step 4: Data Preprocessing

4.1 Handling Missing Values

```
# Check for missing values
print(data.isnull().sum())
```

```
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                    1755
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
```

```
# Fill missing values in numerical columns with the median
for col in num_columns:
    data[col].fillna(data[col].median(), inplace=True)

# Fill missing values in categorical columns with the mode
for col in cat_columns:
    data[col].fillna(data[col].mode()[0], inplace=True)

# Check if there are any remaining missing values
print(data.isnull().sum())
```

```
loan_amnt               0
term                    0
int_rate                0
installment             0
grade                   0
sub_grade               0
emp_title               0
emp_length              0
home_ownership          0
annual_inc              0
```

```
verification_status     0
issue_d                 0
loan_status             0
purpose                 0
title                   0
dti                     0
earliest_cr_line        0
open_acc                0
pub_rec                 0
revol_bal               0
revol_util              0
total_acc               0
initial_list_status     0
application_type        0
mort_acc                0
pub_rec_bankruptcies    0
address                 0
dtype: int64
```

4.2 Encoding Categorical Variables

```
# 1. Label Encoding for 'grade' and 'sub_grade' (Ordinal Variables)
# 'grade' can be manually encoded, and 'sub_grade' can be ordinally
encoded as it contains values like A1, A2, etc.

# Map grades (A > B > C > D > E > F > G)
grade_mapping = {'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6, 'G':
7}
data['grade'] = data['grade'].map(grade_mapping)

# Convert 'sub_grade' to integers based on ordinal relationship
data['sub_grade'] = data['sub_grade'].apply(lambda x: int(x[1]))

# 2. Binary Encoding for 'loan_status'
data['loan_status'].replace({'N': 0, 'Y': 1}, inplace=True)

# 3. One-Hot Encoding for Nominal Variables
nominal_columns = ['term', 'home_ownership', 'verification_status',
'purpose',
                   'initial_list_status', 'application_type']

# Apply pd.get_dummies to these columns
data = pd.get_dummies(data, columns=nominal_columns, drop_first=True)

# 4. Date Handling for 'issue_d' and 'earliest_cr_line'
# Convert 'issue_d' to datetime and extract useful features
data['issue_d'] = pd.to_datetime(data['issue_d'], format='%b-%Y')
data['issue_year'] = data['issue_d'].dt.year
data['issue_month'] = data['issue_d'].dt.month
```

```
data.drop('issue_d', axis=1, inplace=True)

# Convert 'earliest_cr_line' to datetime and extract useful features
data['earliest_cr_line'] = pd.to_datetime(data['earliest_cr_line'],
format='%b-%Y')
data['earliest_cr_line_year'] = data['earliest_cr_line'].dt.year
data.drop('earliest_cr_line', axis=1, inplace=True)

# Check for the new dataframe after encoding
print(data.head())
```

```
   loan_amnt  int_rate  installment  grade  sub_grade  \
0    10000.0     11.44       329.48      2          4
1     8000.0     11.99       265.68      2          5
2    15600.0     10.49       506.97      2          3
3     7200.0      6.49       220.65      1          2
4    24375.0     17.27       609.33      3          5

                 emp_title emp_length  annual_inc  loan_status  \
0                Marketing  10+ years    117000.0   Fully Paid
1          Credit analyst     4 years     65000.0   Fully Paid
2             Statistician   < 1 year     43057.0   Fully Paid
3          Client Advocate    6 years     54000.0   Fully Paid
4  Destiny Management Inc.    9 years     55000.0  Charged Off

                     title  ...  purpose_renewable_energy  \
0                 Vacation  ...                         0
1       Debt consolidation  ...                         0
2  Credit card refinancing  ...                         0
3  Credit card refinancing  ...                         0
4    Credit Card Refinance  ...                         0

   purpose_small_business  purpose_vacation  purpose_wedding  \
0                       0                 1                0
1                       0                 0                0
2                       0                 0                0
3                       0                 0                0
4                       0                 0                0

   initial_list_status_w  application_type_INDIVIDUAL
application_type_JOINT  \
0                      1                            1
0
1                      0                            1
0
2                      0                            1
0
3                      0                            1
0
4                      0                            1
```

```
0

   issue_year issue_month  earliest_cr_line_year
0        2015           1                   1990
1        2015           1                   2004
2        2015           1                   2007
3        2014          11                   2006
4        2013           4                   1999

[5 rows x 46 columns]
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 46 columns):
 #   Column                               Non-Null Count   Dtype
---  ------                               --------------   -----
 0   loan_amnt                            396030 non-null  float64
 1   int_rate                             396030 non-null  float64
 2   installment                          396030 non-null  float64
 3   grade                                396030 non-null  int64
 4   sub_grade                            396030 non-null  int64
 5   emp_title                            396030 non-null  object
 6   emp_length                           396030 non-null  object
 7   annual_inc                           396030 non-null  float64
 8   loan_status                          396030 non-null  object
 9   title                                396030 non-null  object
 10  dti                                  396030 non-null  float64
 11  open_acc                             396030 non-null  float64
 12  pub_rec                              396030 non-null  float64
 13  revol_bal                            396030 non-null  float64
 14  revol_util                           396030 non-null  float64
 15  total_acc                            396030 non-null  float64
 16  mort_acc                             396030 non-null  float64
 17  pub_rec_bankruptcies                 396030 non-null  float64
 18  address                              396030 non-null  object
 19  term_ 60 months                      396030 non-null  uint8
 20  home_ownership_MORTGAGE              396030 non-null  uint8
 21  home_ownership_NONE                  396030 non-null  uint8
 22  home_ownership_OTHER                 396030 non-null  uint8
 23  home_ownership_OWN                   396030 non-null  uint8
 24  home_ownership_RENT                  396030 non-null  uint8
 25  verification_status_Source Verified  396030 non-null  uint8
 26  verification_status_Verified         396030 non-null  uint8
 27  purpose_credit_card                  396030 non-null  uint8
 28  purpose_debt_consolidation           396030 non-null  uint8
 29  purpose_educational                  396030 non-null  uint8
 30  purpose_home_improvement             396030 non-null  uint8
 31  purpose_house                        396030 non-null  uint8
```

```
 32  purpose_major_purchase               396030 non-null  uint8
 33  purpose_medical                      396030 non-null  uint8
 34  purpose_moving                       396030 non-null  uint8
 35  purpose_other                        396030 non-null  uint8
 36  purpose_renewable_energy             396030 non-null  uint8
 37  purpose_small_business               396030 non-null  uint8
 38  purpose_vacation                     396030 non-null  uint8
 39  purpose_wedding                      396030 non-null  uint8
 40  initial_list_status_w                396030 non-null  uint8
 41  application_type_INDIVIDUAL          396030 non-null  uint8
 42  application_type_JOINT               396030 non-null  uint8
 43  issue_year                           396030 non-null  int64
 44  issue_month                          396030 non-null  int64
 45  earliest_cr_line_year                396030 non-null  int64
dtypes: float64(12), int64(5), object(5), uint8(24)
memory usage: 75.5+ MB
```

4.3: Feature Scaling

Random Forest is not sensitive to scaling, but if we were using distance-based algorithms, we would scale the features.

Step 5: Feature Engineering

```
data.columns
```

```
Index(['loan_amnt', 'int_rate', 'installment', 'grade', 'sub_grade',
       'emp_title', 'emp_length', 'annual_inc', 'loan_status',
'title', 'dti',
       'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'mort_acc', 'pub_rec_bankruptcies', 'address', 'term_ 60
months',
       'home_ownership_MORTGAGE', 'home_ownership_NONE',
       'home_ownership_OTHER', 'home_ownership_OWN',
'home_ownership_RENT',
       'verification_status_Source Verified',
'verification_status_Verified',
       'purpose_credit_card', 'purpose_debt_consolidation',
       'purpose_educational', 'purpose_home_improvement',
'purpose_house',
       'purpose_major_purchase', 'purpose_medical', 'purpose_moving',
       'purpose_other', 'purpose_renewable_energy',
'purpose_small_business',
       'purpose_vacation', 'purpose_wedding', 'initial_list_status_w',
       'application_type_INDIVIDUAL', 'application_type_JOINT',
'issue_year',
       'issue_month', 'earliest_cr_line_year'],
      dtype='object')
```

```
# Income-Loan Ratio: Calculated as the ratio of loan amount to annual
income
 # Calculate the mean of annual_inc, ignoring zero values
mean_annual_inc = data.loc[data['annual_inc'] != 0,
'annual_inc'].mean()

# Replace zero annual_inc with the mean annual_inc
data['annual_inc'] = data['annual_inc'].replace(0, mean_annual_inc)

# Calculate income_loan_ratio
data['income_loan_ratio'] = data['loan_amnt'] / data['annual_inc']

data.info()
```

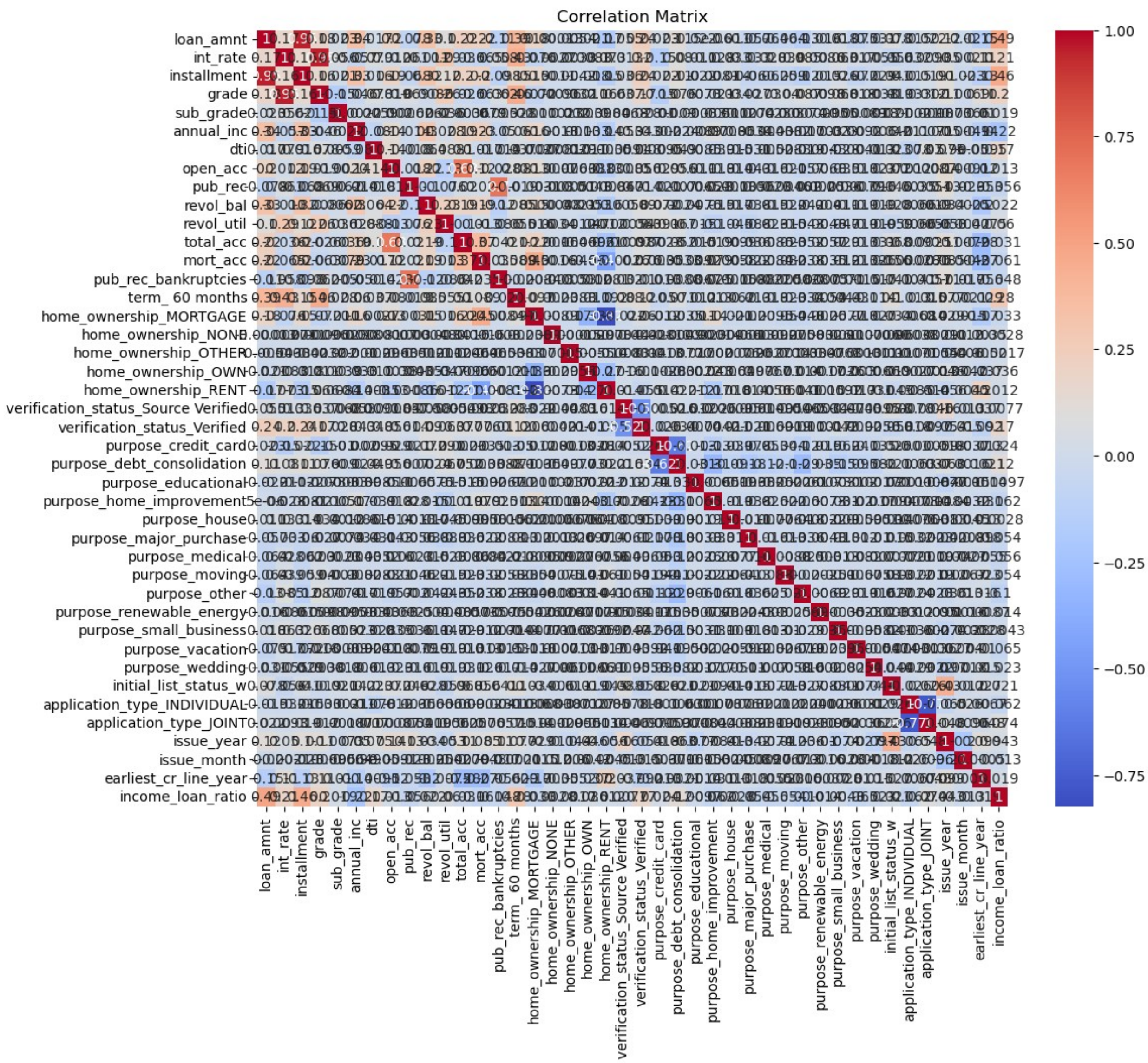
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 47 columns):
 #   Column                               Non-Null Count   Dtype
---  ------                               --------------   -----
 0   loan_amnt                            396030 non-null  float64
 1   int_rate                             396030 non-null  float64
 2   installment                          396030 non-null  float64
 3   grade                                396030 non-null  int64
 4   sub_grade                            396030 non-null  int64
 5   emp_title                            396030 non-null  object
 6   emp_length                           396030 non-null  object
 7   annual_inc                           396030 non-null  float64
 8   loan_status                          396030 non-null  object
 9   title                                396030 non-null  object
 10  dti                                  396030 non-null  float64
 11  open_acc                             396030 non-null  float64
 12  pub_rec                              396030 non-null  float64
 13  revol_bal                            396030 non-null  float64
 14  revol_util                           396030 non-null  float64
 15  total_acc                            396030 non-null  float64
 16  mort_acc                             396030 non-null  float64
 17  pub_rec_bankruptcies                 396030 non-null  float64
 18  address                              396030 non-null  object
 19  term_ 60 months                      396030 non-null  uint8
 20  home_ownership_MORTGAGE              396030 non-null  uint8
 21  home_ownership_NONE                  396030 non-null  uint8
 22  home_ownership_OTHER                 396030 non-null  uint8
 23  home_ownership_OWN                   396030 non-null  uint8
 24  home_ownership_RENT                  396030 non-null  uint8
 25  verification_status_Source Verified  396030 non-null  uint8
 26  verification_status_Verified         396030 non-null  uint8
 27  purpose_credit_card                  396030 non-null  uint8
 28  purpose_debt_consolidation           396030 non-null  uint8
 29  purpose_educational                  396030 non-null  uint8
 30  purpose_home_improvement             396030 non-null  uint8
```

```
 31  purpose_house                         396030 non-null  uint8
 32  purpose_major_purchase                396030 non-null  uint8
 33  purpose_medical                       396030 non-null  uint8
 34  purpose_moving                        396030 non-null  uint8
 35  purpose_other                         396030 non-null  uint8
 36  purpose_renewable_energy              396030 non-null  uint8
 37  purpose_small_business                396030 non-null  uint8
 38  purpose_vacation                      396030 non-null  uint8
 39  purpose_wedding                       396030 non-null  uint8
 40  initial_list_status_w                 396030 non-null  uint8
 41  application_type_INDIVIDUAL           396030 non-null  uint8
 42  application_type_JOINT                396030 non-null  uint8
 43  issue_year                            396030 non-null  int64
 44  issue_month                           396030 non-null  int64
 45  earliest_cr_line_year                 396030 non-null  int64
 46  income_loan_ratio                     396030 non-null  float64
dtypes: float64(13), int64(5), object(5), uint8(24)
memory usage: 78.6+ MB
```

Step 6: Feature Selection

6.1 Correlation Matrix

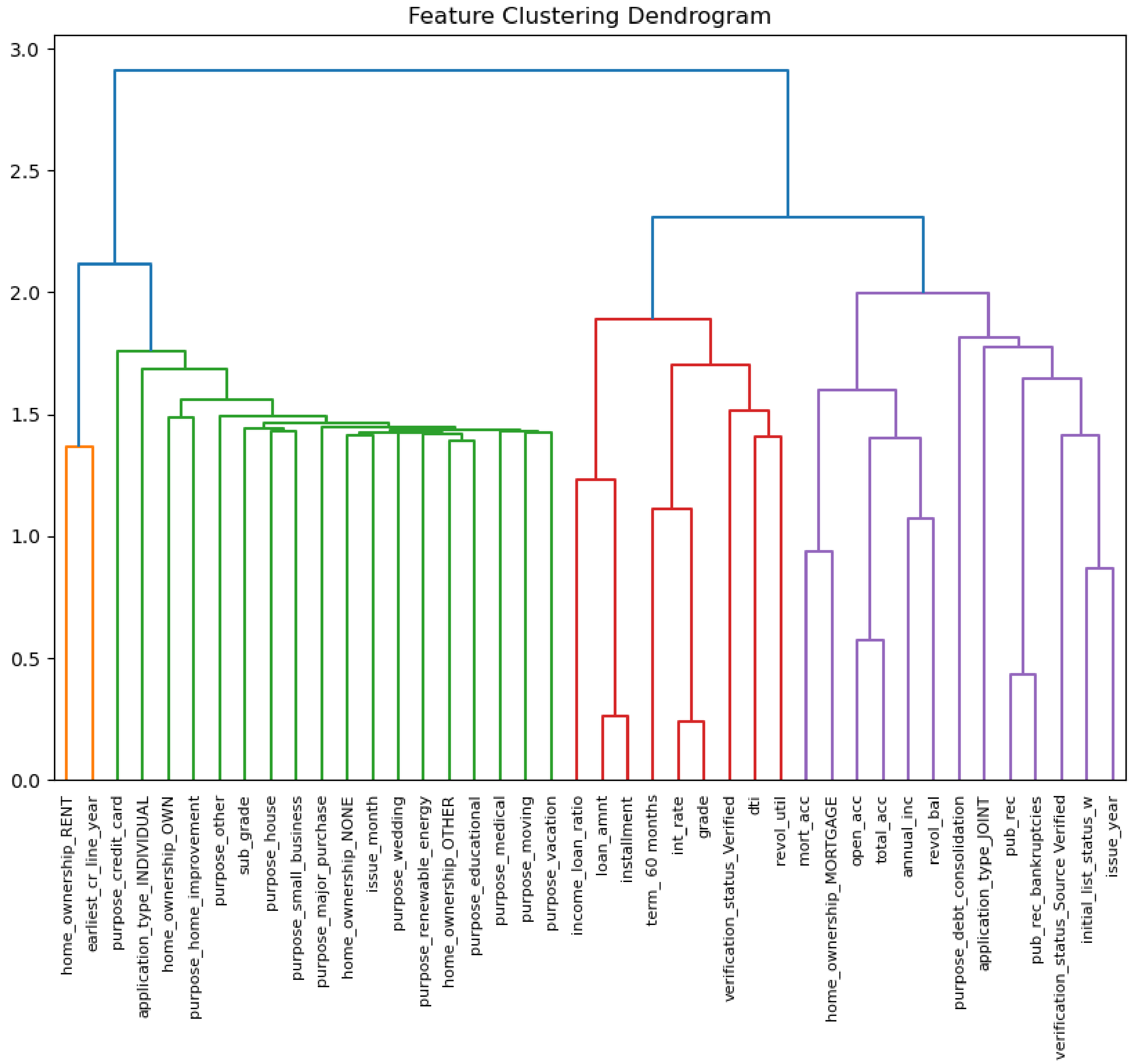
```
plt.figure(figsize=(12, 10))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



Correlation Clustering:

```
Cluster features based on their correlations. Group similar features
into clusters and retain only one representative feature from each
cluster.

import scipy.cluster.hierarchy as sch

# Perform hierarchical clustering on the correlation matrix
corr_matrix = data.corr()
corr_linkage = sch.linkage(sch.distance.pdist(corr_matrix),
method='complete')

# Plot the dendrogram to visualize clusters
plt.figure(figsize=(10, 7))
dendrogram = sch.dendrogram(corr_linkage, labels=corr_matrix.columns,
```

```
leaf_rotation=90)
plt.title('Feature Clustering Dendrogram')
plt.show()

# You can then manually select features from each cluster
```



6.2 Feature Importance from Random Forest

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 47 columns):
 #   Column                                    Non-Null Count   Dtype
```

```
---  ------                               --------------   -----
 0   loan_amnt                            396030 non-null  float64
 1   int_rate                             396030 non-null  float64
 2   installment                          396030 non-null  float64
 3   grade                                396030 non-null  int64
 4   sub_grade                            396030 non-null  int64
 5   emp_title                            396030 non-null  object
 6   emp_length                           396030 non-null  object
 7   annual_inc                           396030 non-null  float64
 8   loan_status                          396030 non-null  object
 9   title                                396030 non-null  object
 10  dti                                  396030 non-null  float64
 11  open_acc                             396030 non-null  float64
 12  pub_rec                              396030 non-null  float64
 13  revol_bal                            396030 non-null  float64
 14  revol_util                           396030 non-null  float64
 15  total_acc                            396030 non-null  float64
 16  mort_acc                             396030 non-null  float64
 17  pub_rec_bankruptcies                 396030 non-null  float64
 18  address                              396030 non-null  object
 19  term_ 60 months                      396030 non-null  uint8
 20  home_ownership_MORTGAGE              396030 non-null  uint8
 21  home_ownership_NONE                  396030 non-null  uint8
 22  home_ownership_OTHER                 396030 non-null  uint8
 23  home_ownership_OWN                   396030 non-null  uint8
 24  home_ownership_RENT                  396030 non-null  uint8
 25  verification_status_Source Verified  396030 non-null  uint8
 26  verification_status_Verified         396030 non-null  uint8
 27  purpose_credit_card                  396030 non-null  uint8
 28  purpose_debt_consolidation           396030 non-null  uint8
 29  purpose_educational                  396030 non-null  uint8
 30  purpose_home_improvement             396030 non-null  uint8
 31  purpose_house                        396030 non-null  uint8
 32  purpose_major_purchase               396030 non-null  uint8
 33  purpose_medical                      396030 non-null  uint8
 34  purpose_moving                       396030 non-null  uint8
 35  purpose_other                        396030 non-null  uint8
 36  purpose_renewable_energy             396030 non-null  uint8
 37  purpose_small_business               396030 non-null  uint8
 38  purpose_vacation                     396030 non-null  uint8
 39  purpose_wedding                      396030 non-null  uint8
 40  initial_list_status_w                396030 non-null  uint8
 41  application_type_INDIVIDUAL          396030 non-null  uint8
 42  application_type_JOINT               396030 non-null  uint8
 43  issue_year                           396030 non-null  int64
 44  issue_month                          396030 non-null  int64
 45  earliest_cr_line_year                396030 non-null  int64
 46  income_loan_ratio                    396030 non-null  float64
```

```
dtypes: float64(13), int64(5), object(5), uint8(24)
memory usage: 78.6+ MB
```

```
data = data.drop(columns=['emp_title','emp_length','address','title'],
axis=1)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 43 columns):
 #   Column                               Non-Null Count   Dtype
---  ------                               --------------   -----
 0   loan_amnt                            396030 non-null  float64
 1   int_rate                             396030 non-null  float64
 2   installment                          396030 non-null  float64
 3   grade                                396030 non-null  int64
 4   sub_grade                            396030 non-null  int64
 5   annual_inc                           396030 non-null  float64
 6   loan_status                          396030 non-null  object
 7   dti                                  396030 non-null  float64
 8   open_acc                             396030 non-null  float64
 9   pub_rec                              396030 non-null  float64
 10  revol_bal                            396030 non-null  float64
 11  revol_util                           396030 non-null  float64
 12  total_acc                            396030 non-null  float64
 13  mort_acc                             396030 non-null  float64
 14  pub_rec_bankruptcies                 396030 non-null  float64
 15  term_ 60 months                      396030 non-null  uint8
 16  home_ownership_MORTGAGE              396030 non-null  uint8
 17  home_ownership_NONE                  396030 non-null  uint8
 18  home_ownership_OTHER                 396030 non-null  uint8
 19  home_ownership_OWN                   396030 non-null  uint8
 20  home_ownership_RENT                  396030 non-null  uint8
 21  verification_status_Source Verified  396030 non-null  uint8
 22  verification_status_Verified         396030 non-null  uint8
 23  purpose_credit_card                  396030 non-null  uint8
 24  purpose_debt_consolidation           396030 non-null  uint8
 25  purpose_educational                  396030 non-null  uint8
 26  purpose_home_improvement             396030 non-null  uint8
 27  purpose_house                        396030 non-null  uint8
 28  purpose_major_purchase               396030 non-null  uint8
 29  purpose_medical                      396030 non-null  uint8
 30  purpose_moving                       396030 non-null  uint8
 31  purpose_other                        396030 non-null  uint8
 32  purpose_renewable_energy             396030 non-null  uint8
 33  purpose_small_business               396030 non-null  uint8
 34  purpose_vacation                     396030 non-null  uint8
 35  purpose_wedding                      396030 non-null  uint8
 36  initial_list_status_w                396030 non-null  uint8
 37  application_type_INDIVIDUAL          396030 non-null  uint8
```

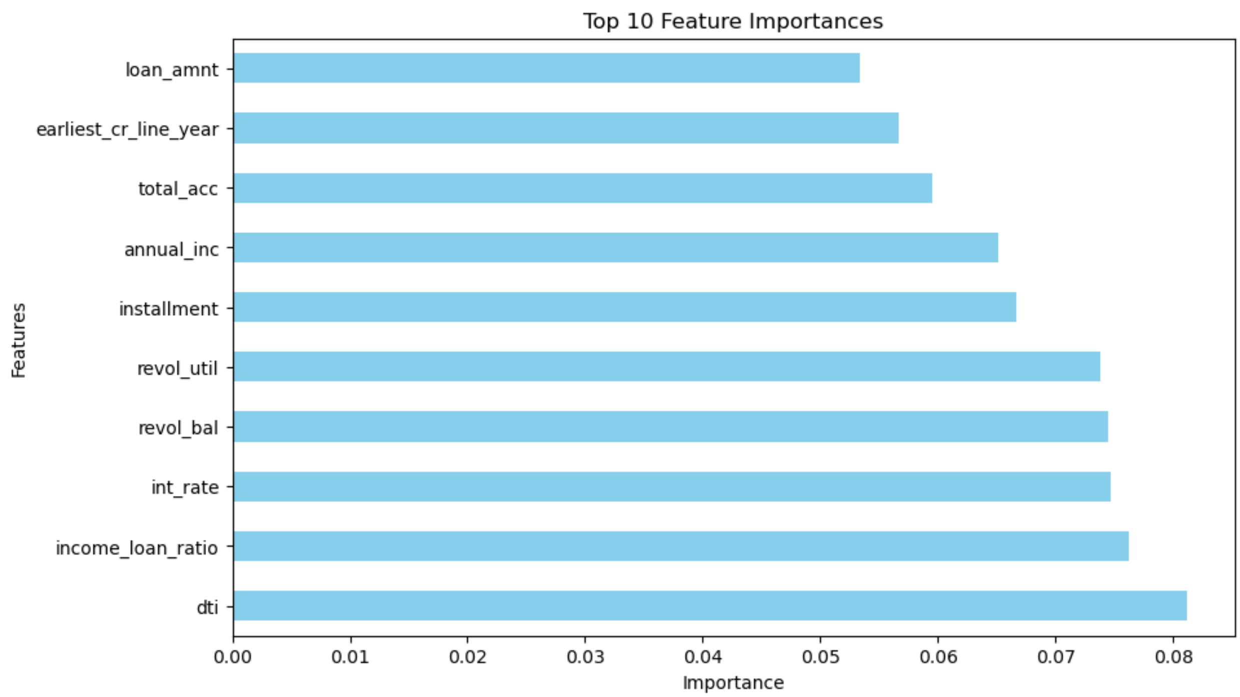
```
 38  application_type_JOINT                    396030 non-null  uint8
 39  issue_year                                396030 non-null  int64
 40  issue_month                               396030 non-null  int64
 41  earliest_cr_line_year                     396030 non-null  int64
 42  income_loan_ratio                         396030 non-null  float64
dtypes: float64(13), int64(5), object(1), uint8(24)
memory usage: 66.5+ MB
```

```python
# Check for infinite values
print(data.isin([np.inf, -np.inf]).sum())
```

```
loan_amnt                                0
int_rate                                 0
installment                              0
grade                                    0
sub_grade                                0
annual_inc                               0
loan_status                              0
dti                                      0
open_acc                                 0
pub_rec                                  0
revol_bal                                0
revol_util                               0
total_acc                                0
mort_acc                                 0
pub_rec_bankruptcies                     0
term_ 60 months                          0
home_ownership_MORTGAGE                  0
home_ownership_NONE                      0
home_ownership_OTHER                     0
home_ownership_OWN                       0
home_ownership_RENT                      0
verification_status_Source Verified      0
verification_status_Verified             0
purpose_credit_card                      0
purpose_debt_consolidation               0
purpose_educational                      0
purpose_home_improvement                 0
purpose_house                            0
purpose_major_purchase                   0
purpose_medical                          0
purpose_moving                           0
purpose_other                            0
purpose_renewable_energy                 0
purpose_small_business                   0
purpose_vacation                         0
purpose_wedding                          0
initial_list_status_w                    0
application_type_INDIVIDUAL              0
application_type_JOINT                   0
```

```
issue_year                                       0
issue_month                                      0
earliest_cr_line_year                            0
income_loan_ratio                                0
dtype: int64
```

```python
X = data.drop(['loan_status'], axis=1)
y = data['loan_status']

# Fit a preliminary model
rf = RandomForestClassifier(random_state=42)
rf.fit(X, y)


importances = pd.Series(rf.feature_importances_, index=X.columns)

# Sort the importances in descending order and select the top 10
features
top_10_importances = importances.sort_values(ascending=False).head(10)

# Plot the top 10 feature importances
plt.figure(figsize=(10, 6))
top_10_importances.plot(kind='barh', color='skyblue')
plt.title('Top 10 Feature Importances')
plt.xlabel('Importance')
plt.ylabel('Features')
plt.show()

# Display the top 10 feature importances
print(top_10_importances)
```



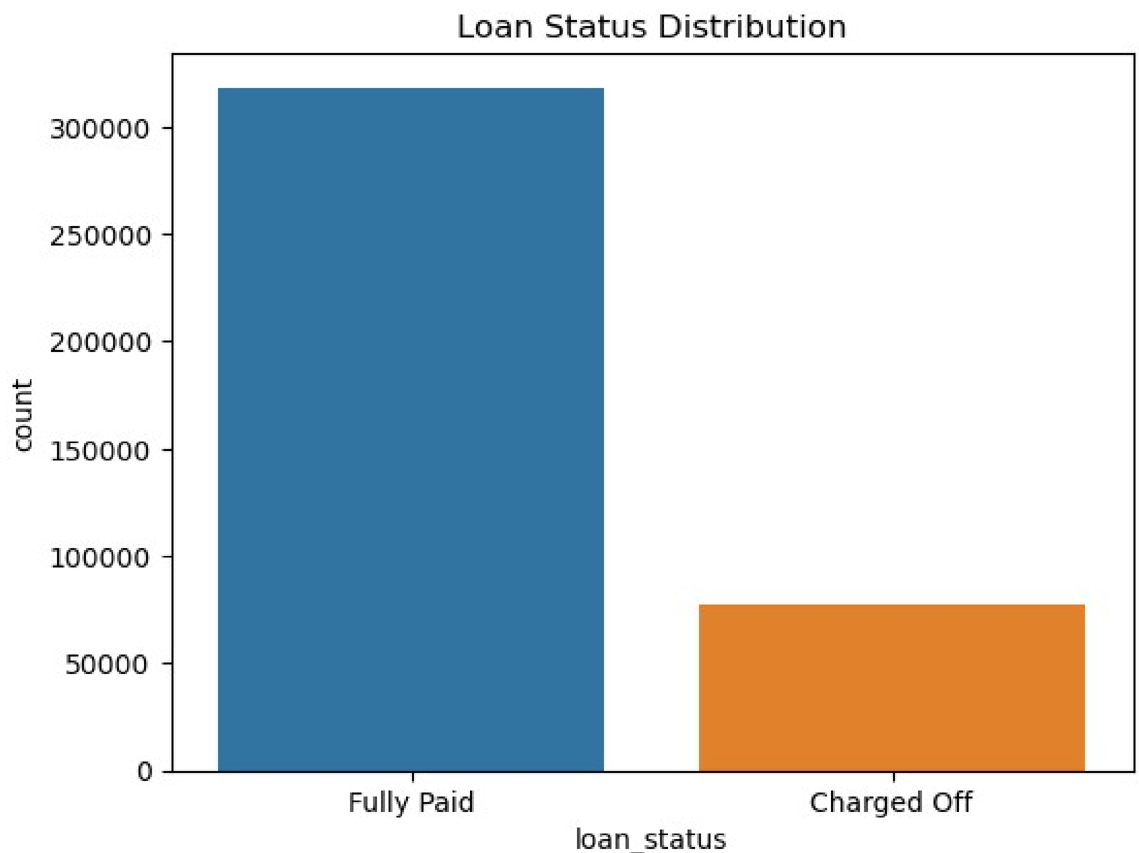

```
dti                       0.081124
income_loan_ratio         0.076196
int_rate                  0.074690
revol_bal                 0.074499
revol_util                0.073837
installment               0.066670
annual_inc                0.065163
total_acc                 0.059488
earliest_cr_line_year     0.056665
loan_amnt                 0.053313
dtype: float64
```

Step 7: Handling Imbalanced Data

7.1 Checking Imbalance

```python
print(data['loan_status'].value_counts())

sns.countplot(x='loan_status', data=data)
plt.title('Loan Status Distribution')
plt.show()
```

```
Fully Paid     318357
Charged Off     77673
Name: loan_status, dtype: int64
```



7.2 Applying SMOTE (Synthetic Minority Over-sampling Technique)

```
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X, y)

# Check the new class distribution
print(pd.Series(y_res).value_counts())

Fully Paid     318357
Charged Off    318357
Name: loan_status, dtype: int64
```

Step 8: Model Building

8.1 Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res,
test_size=0.2, random_state=42)
```

8.2. Building the Random Forest Model

```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```
# Predict the target labels for the test set
y_pred = rf_model.predict(X_test)
```

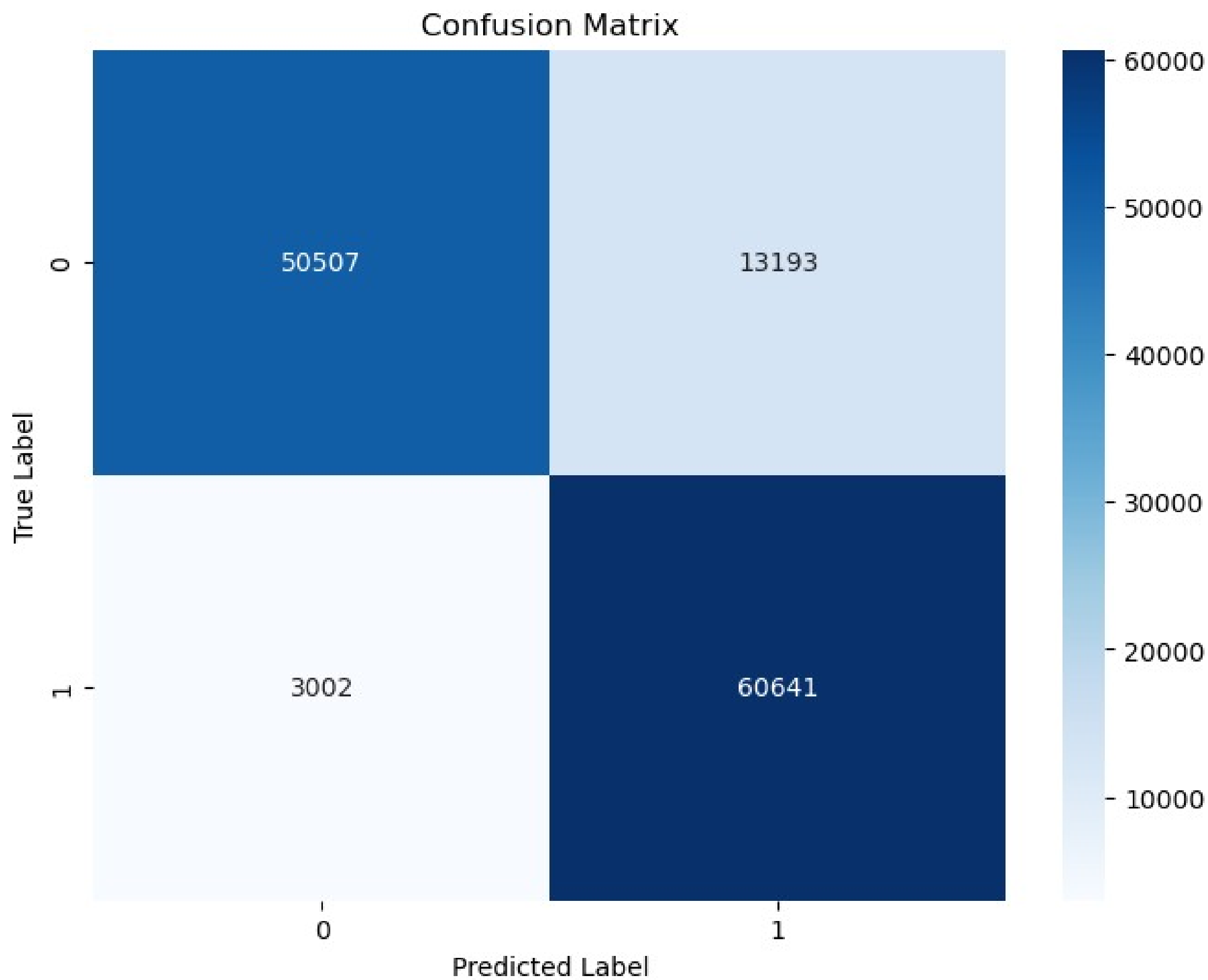
```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.8728237908640444
```

```
# Generate the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[50507 13193]
 [ 3002 60641]]
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```



Accuracy: 0.8728

```
The accuracy is approximately 87.28%, which means that the model
correctly predicted whether a loan was "Fully Paid" or "Charged Off"
about 87.3% of the time.
Out of all the predictions made, 87.28% were correct.
```

The confusion matrix shows how well the model performed by breaking down the correct and incorrect predictions for each class:

```
50507 (Top-left):
These are the True Positives (TP). The model correctly predicted
"Fully Paid" loans 50,507 times.

13193 (Top-right):
These are the False Positives (FP). The model incorrectly predicted
13,193 loans as "Fully Paid", but they were actually "Charged Off".

3002 (Bottom-left):
```

```
These are the False Negatives (FN). The model incorrectly predicted
3,002 loans as "Charged Off", but they were actually "Fully Paid".

60641 (Bottom-right):
These are the True Negatives (TN). The model correctly predicted
"Charged Off" loans 60,641 times.
```

Step 9: Hyperparameter Tuning

9.1 Setting up the Parameter Grid

```
param_grid = {
    'n_estimators': [100, 200, 500],
    'max_depth': [4, 6, 8, None],
    'max_features': ['sqrt', 'log2'],
    'min_samples_split': [2, 5, 10],
     'bootstrap': [True, False]
}

param_grid = {
    'n_estimators': [100, 200],  # Fewer options
    'max_depth': [6, None],  # Fewer depths to explore
    'max_features': ['sqrt'],  # Use just 'sqrt'
    'min_samples_split': [2, 5],  # Fewer split points
    'bootstrap': [True]  # Only one option for bootstrap
}
```

9.2 Using RandomizedSearchCV

```
random_search = RandomizedSearchCV(estimator=rf_model,
param_distributions=param_grid, n_iter=8, cv=3, scoring='roc_auc',
n_jobs=-1, random_state=42)
random_search.fit(X_train, y_train)

print("Best Parameters:", random_search.best_params_)

Best Parameters: {'n_estimators': 200, 'min_samples_split': 2,
'max_features': 'sqrt', 'max_depth': None, 'bootstrap': True}
```

9.3 Training the Model with Best Parameters

```
best_rf = random_search.best_estimator_
best_rf.fit(X_train, y_train)

RandomForestClassifier(n_estimators=200, random_state=42)
```

Step 10: Model Evaluation

10.1 Predictions on Validation Set

```
y_pred = best_rf.predict(X_valid)
y_proba = best_rf.predict_proba(X_valid)[:, 1]
```

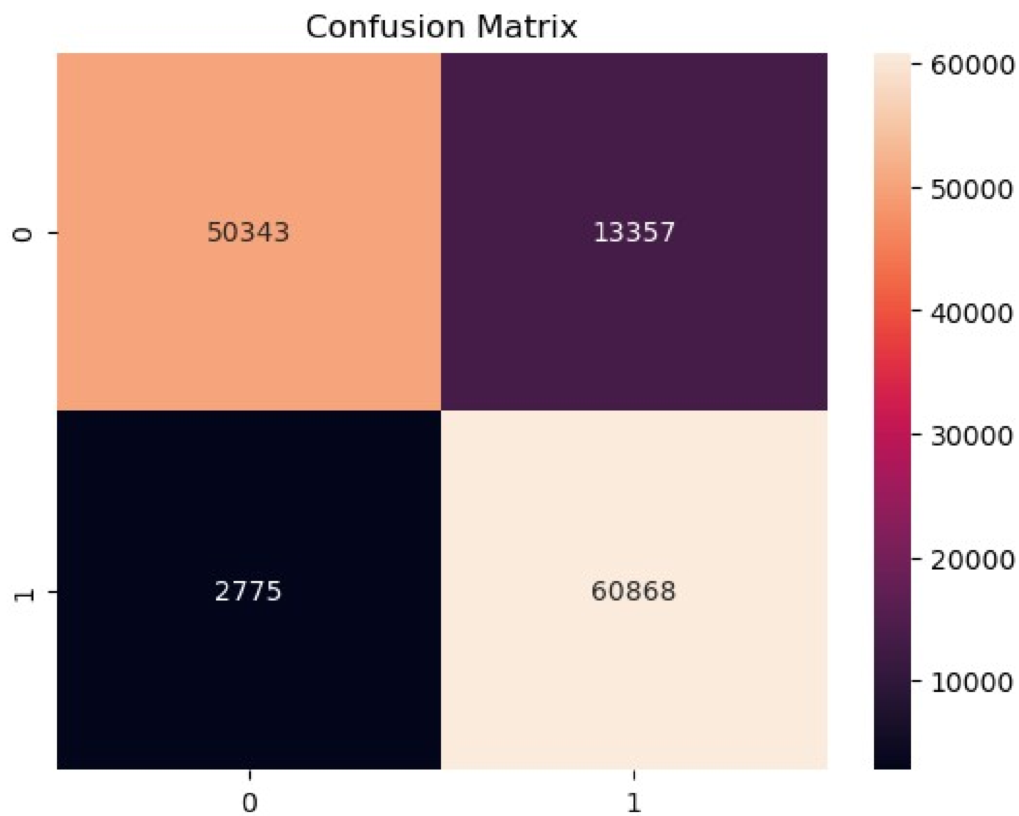
10.2 Classification Report

```
print(classification_report(y_valid, y_pred))
```

```
              precision    recall  f1-score   support

 Charged Off       0.95      0.79      0.86     63700
  Fully Paid       0.82      0.96      0.88     63643

    accuracy                           0.87    127343
   macro avg       0.88      0.87      0.87    127343
weighted avg       0.88      0.87      0.87    127343
```

10.3 Confusion Matrix

```
cm = confusion_matrix(y_valid, y_pred)
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.show()
```



10.5 Cross-Validation Score

```
cv_scores = cross_val_score(best_rf, X_res, y_res, cv=5,
scoring='roc_auc')
print("Cross-Validation AUC Scores:", cv_scores)
print("Mean CV AUC Score:", cv_scores.mean())

Cross-Validation AUC Scores: [0.71404065 0.93998657 0.99620884
0.99620909 0.99611502]
Mean CV AUC Score: 0.9285120327121297
```

Step 11: Conclusion

In this project, we developed a comprehensive Random Forest model to predict loan approval status. We covered extensive data preprocessing, feature engineering, handling imbalanced data with SMOTE, hyperparameter tuning with RandomizedSearchCV, and thorough model evaluation. Additionally, we interpreted the model using feature importances and SHAP values to understand the impact of each feature on the predictions.