loantap

September 30, 2024

0.1 About LoanTap

LoanTap is an online platform committed to delivering customized loan products to millennials. They innovate in an otherwise dull loan segment, to deliver instant, flexible loans on consumer-friendly terms to salaried professionals and businessmen.

0.1.1 Business Problem

The data science team at LoanTap is building an underwriting layer to determine the creditworthiness of MSMEs as well as individuals.

LoanTap deploys formal credit to salaried individuals and businesses through 4 main financial instruments:

- Personal Loan
- EMI Free Loan
- Personal Overdraft
- Advance Salary Loan

This case study will focus on the underwriting process behind Personal Loan only.

0.1.2 Dataset

Column Name	Description
`loan_amnt`	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
`term`	The number of payments on the loan. Values are in months and can be either 36 or 60.
`int_rate`	Interest Rate on the loan.
`installment`	The monthly payment owed by the borrower if the loan originates.
`grade`	LoanTap assigned loan grade.
`sub_grade`	LoanTap assigned loan subgrade.
`emp_title`	The job title supplied by the Borrower when applying for the loan.
`emp_length`	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
`home_ownership`	The home ownership status provided by the borrower during registration or obtained from the credit report.
`annual_inc`	The self-reported annual income provided by the borrower during registration.

Column Name	Description
verification_status	Indicates if income was verified by LoanTap, not verified, or if the income source was verified.
issue_d	The month which the loan was funded.
loan_status	Current status of the loan - Target Variable.
purpose	A category provided by the borrower for the loan request.
title	The loan title provided by the borrower.
dti	A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LoanTap loan, divided by the borrower's self-reported monthly income.
earliest_cr_line	The month the borrower's earliest reported credit line was opened.
open_acc	The number of open credit lines in the borrower's credit file.
pub_rec	Number of derogatory public records.
revol_bal	Total credit revolving balance.
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
total_acc	The total number of credit lines currently in the borrower's credit file.
initial_list_status	The initial listing status of the loan. Possible values are – W, F.
application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers.
mort_acc	Number of mortgage accounts.
pub_rec_bankruptcies	Number of public record bankruptcies.
Address	Address of the individual.

Importing Required Libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

palette = ['#963f8d', '#624496']
```

```
sns.set(style='ticks', palette=palette)
```

Read Dataset

[5]:
```
df = pd.read_csv('../data/LoanTapData.csv')
df.sample(5)
```

[5]:
```
        loan_amnt        term  int_rate  installment grade sub_grade  \
263539    20500.0   60 months     20.99       554.48     E        E4
25692     10000.0   36 months     10.99       327.34     B        B4
18277      6500.0   36 months     10.15       210.20     B        B2
327083     5350.0   36 months     12.12       178.01     B        B3
306382     1500.0   36 months      7.88        46.93     A        A5

                              emp_title emp_length home_ownership  \
263539              Behavior Specialist  10+ years       MORTGAGE
25692                               Rca  10+ years       MORTGAGE
18277                     Payroll Admin    7 years           RENT
327083  peacock interiors and gallery inc    4 years           RENT
306382          Lawrence public schools    9 years           RENT

        annual_inc verification_status   issue_d  loan_status  \
263539     57000.0     Source Verified  May-2014  Charged Off
25692      40000.0     Source Verified  Jul-2015   Fully Paid
18277      46000.0     Source Verified  Jul-2014   Fully Paid
327083     40000.0        Not Verified  Dec-2012   Fully Paid
306382     35000.0        Not Verified  Apr-2010   Fully Paid

                   purpose                    title    dti earliest_cr_line  \
263539  debt_consolidation       Debt consolidation   9.41         Sep-1990
25692   debt_consolidation       Debt consolidation  15.30         May-2001
18277          credit_card  Credit card refinancing  32.92         Oct-1995
327083         credit_card      debit consolidation  13.50         Dec-2004
306382            vacation                   kitten  11.69         Dec-1988

        open_acc  pub_rec  revol_bal  revol_util  total_acc  \
263539       9.0      5.0     4766.0        54.2       15.0
25692       13.0      2.0    10602.0        35.6       27.0
18277       13.0      0.0    14818.0        42.5       30.0
327083       7.0      0.0    16607.0        40.3       15.0
306382       9.0      0.0     3162.0        22.4       27.0

       initial_list_status application_type  mort_acc  pub_rec_bankruptcies  \
263539                   w       INDIVIDUAL       1.0                   0.0
25692                    w       INDIVIDUAL       5.0                   2.0
18277                    f       INDIVIDUAL       0.0                   0.0
327083                   f       INDIVIDUAL       0.0                   0.0
306382                   f       INDIVIDUAL       NaN                   0.0
```

```
                                                   address
263539         01246 Carrie Passage\r\nNew Kyle, ND 11650
25692   00296 Kirk Bypass Suite 650\r\nPerezburgh, LA …
18277            81136 Beth Hollow\r\nRhondaland, RI 00813
327083  2423 Hines Spring Apt. 664\r\nNew Pamela, IL 0…
306382       302 Schmidt Avenue\r\nJeffreyville, NM 00813
```

[6]:
```
print("Shape of the data: ", df.shape)
print("The Given Dataset has {} rows and {} columns".format(df.shape[0], df.
 ↪shape[1]))
print("Columns: ", df.columns.to_list())
```

```
Shape of the data:  (396030, 27)
The Given Dataset has 396030 rows and 27 columns
Columns:  ['loan_amnt', 'term', 'int_rate', 'installment', 'grade', 'sub_grade',
'emp_title', 'emp_length', 'home_ownership', 'annual_inc',
'verification_status', 'issue_d', 'loan_status', 'purpose', 'title', 'dti',
'earliest_cr_line', 'open_acc', 'pub_rec', 'revol_bal', 'revol_util',
'total_acc', 'initial_list_status', 'application_type', 'mort_acc',
'pub_rec_bankruptcies', 'address']
```

0.1.3 Shape

- The dataset comprises 396030 rows and 27 columns, representing a volume of data.
- Each row corresponds to each loan distribution by the company.

0.1.4 Data Structure

[7]:
```
df.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                 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
```

[8]:
```
df.isnull().sum()
```

[8]:
```
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
```

[9]:
```
df_missing = df.isnull().sum()
df_missing = df_missing[df_missing > 0]
df_missing = df_missing.sort_values(ascending=False)
df_missing = df_missing.to_frame()
df_missing.columns = ['count']
df_missing
```

[9]:

	count
mort_acc	37795
emp_title	22927
emp_length	18301
title	1756
pub_rec_bankruptcies	535
revol_util	276

[10]:
```
numeric_features = ['mort_acc', 'pub_rec_bankruptcies', 'revol_util']
df[numeric_features] = df[numeric_features].fillna(df[numeric_features].
 ↪median())

categorical_features = ['emp_title', 'emp_length', 'title']
df[categorical_features] = df[categorical_features].
 ↪fillna(df[categorical_features].mode().iloc[0])

df.isnull().sum()
```

[10]:
```
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
```

[11]:
```
df.duplicated().sum()
```

[11]:
```
np.int64(0)
```

[12]:
```
df.describe().T
```

[12]:
```
                         count          mean           std     min       25%  \
loan_amnt             396030.0  14113.888089   8357.441341  500.00   8000.00
int_rate              396030.0     13.639400      4.472157    5.32     10.49
installment           396030.0    431.849698    250.727790   16.08    250.33
annual_inc            396030.0  74203.175798  61637.621158    0.00  45000.00
dti                   396030.0     17.379514     18.019092    0.00     11.28
open_acc              396030.0     11.311153      5.137649    0.00      8.00
pub_rec               396030.0      0.178191      0.530671    0.00      0.00
revol_bal             396030.0  15844.539853  20591.836109    0.00   6025.00
revol_util            396030.0     53.792451     24.443685    0.00     35.90
total_acc             396030.0     25.414744     11.886991    2.00     17.00
mort_acc              396030.0      1.736308      2.056819    0.00      0.00
pub_rec_bankruptcies  396030.0      0.121483      0.355962    0.00      0.00

                           50%       75%         max
loan_amnt             12000.00  20000.00    40000.00
int_rate                 13.33     16.49       30.99
installment             375.43    567.30     1533.81
annual_inc            64000.00  90000.00  8706582.00
dti                      16.91     22.98     9999.00
open_acc                 10.00     14.00       90.00
pub_rec                   0.00      0.00       86.00
revol_bal             11181.00  19620.00  1743266.00
revol_util               54.80     72.90      892.30
total_acc                24.00     32.00      151.00
mort_acc                  1.00      3.00       34.00
pub_rec_bankruptcies      0.00      0.00        8.00
```

[13]:
```
df.describe(include='object').T
```

[13]:
```
                       count  unique                          top    freq
term                  396030       2                    36 months  302005
grade                 396030       7                            B  116018
sub_grade             396030      35                           B3   26655
```

```
emp_title             396030  173105                    Teacher   27316
emp_length            396030      11                  10+ years  144342
home_ownership        396030       6                   MORTGAGE  198348
verification_status   396030       3                   Verified  139563
issue_d               396030     115                   Oct-2014   14846
loan_status           396030       2                 Fully Paid  318357
purpose               396030      14         debt_consolidation  234507
title                 396030   48816         Debt consolidation  154228
earliest_cr_line      396030     684                   Oct-2000    3017
initial_list_status   396030       2                          f  238066
application_type      396030       3                 INDIVIDUAL  395319
address               396030  393700  USS Smith\r\nFPO AP 70466       8
```

0.1.5 Dataset Information:

- **Data Consistency**: Few columns has missing values in the dataset.
- **Data Types**: Columns are classified into object and float types.
- **Duplicate**: There is no duplicate rows identified

0.1.6 Preprocessing

[14]:
```python
df['pincode'] = df['address'].apply(lambda x: x.split()[-1])
df['pincode'] = df['pincode'].astype('int')
```

[15]:
```python
df['state'] = df['address'].apply(lambda x: x.split()[-2])
df['state'] = df['state'].astype('str')
```

[16]:
```python
# Create flags
df['pub_rec_flag'] = df['pub_rec'].apply(lambda x: 1 if x > 1.0 else 0)
df['mort_acc_flag'] = df['mort_acc'].apply(lambda x: 1 if x > 1.0 else 0)
df['pub_rec_bankruptcies_flag'] = df['pub_rec_bankruptcies'].apply(lambda x: 1␣
↪if x > 1.0 else 0)
```

[17]:
```python
df_ = df[df['loan_status']=='Fully Paid']
df_['emp_title'] = df_['emp_title'].str.lower()
(df_['emp_title'].value_counts(normalize=True) * 100).to_frame().head(10)
```

[17]:
```
                  proportion
emp_title
teacher             6.702852
manager             1.356967
registered nurse    0.658066
supervisor          0.618174
sales               0.555980
rn                  0.517344
driver              0.506350
owner               0.465201
project manager     0.463002
```

```
office manager      0.404891
```

Name the top 2 afforded job titles. - Teacher - Manager

```
[18]: # Columns Might not be useful
useless_columns = ['emp_title', 'issue_d', 'title', 'earliest_cr_line',␣
 ↪'address', 'pub_rec', 'mort_acc', 'pub_rec_bankruptcies']
df = df.drop(useless_columns, axis=1)
```

```
[19]: # Object columns to Category
for col in df.select_dtypes(include='object').columns:
    df[col] = df[col].astype('category')
```

0.1.7 Insight

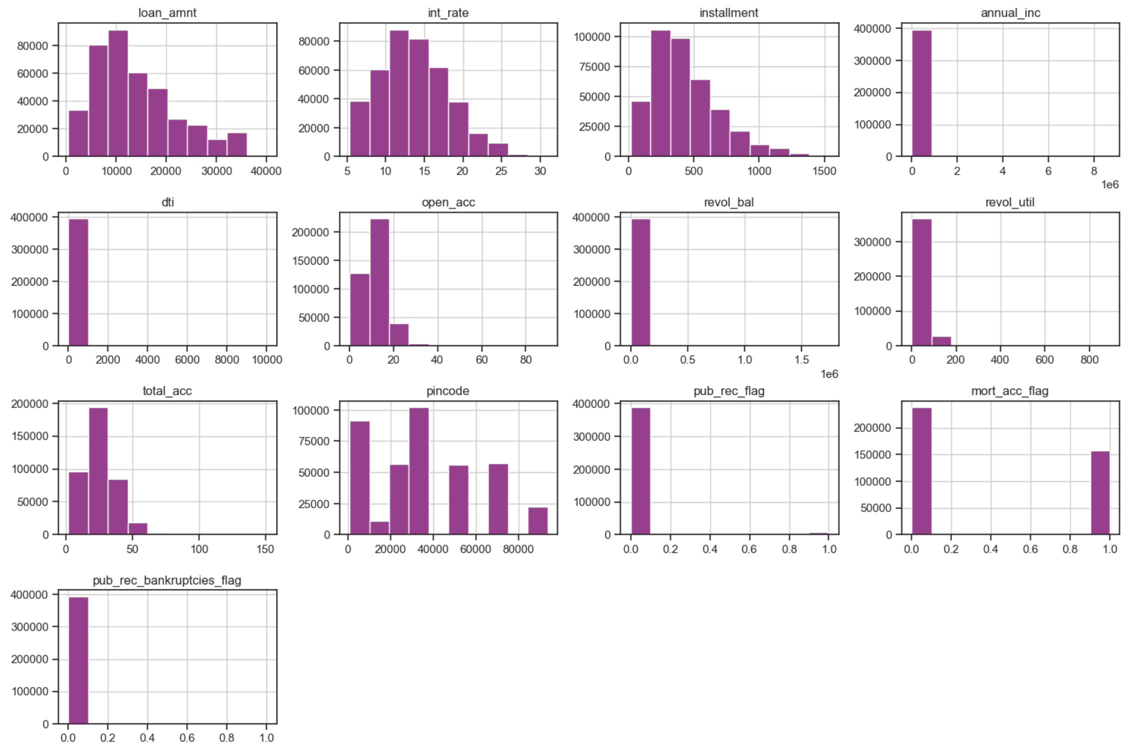
High cardinality columns are removed * emp_title (173105) * title (48816) * address (393700)

Other columns like * issue_d * earliest_cr_line

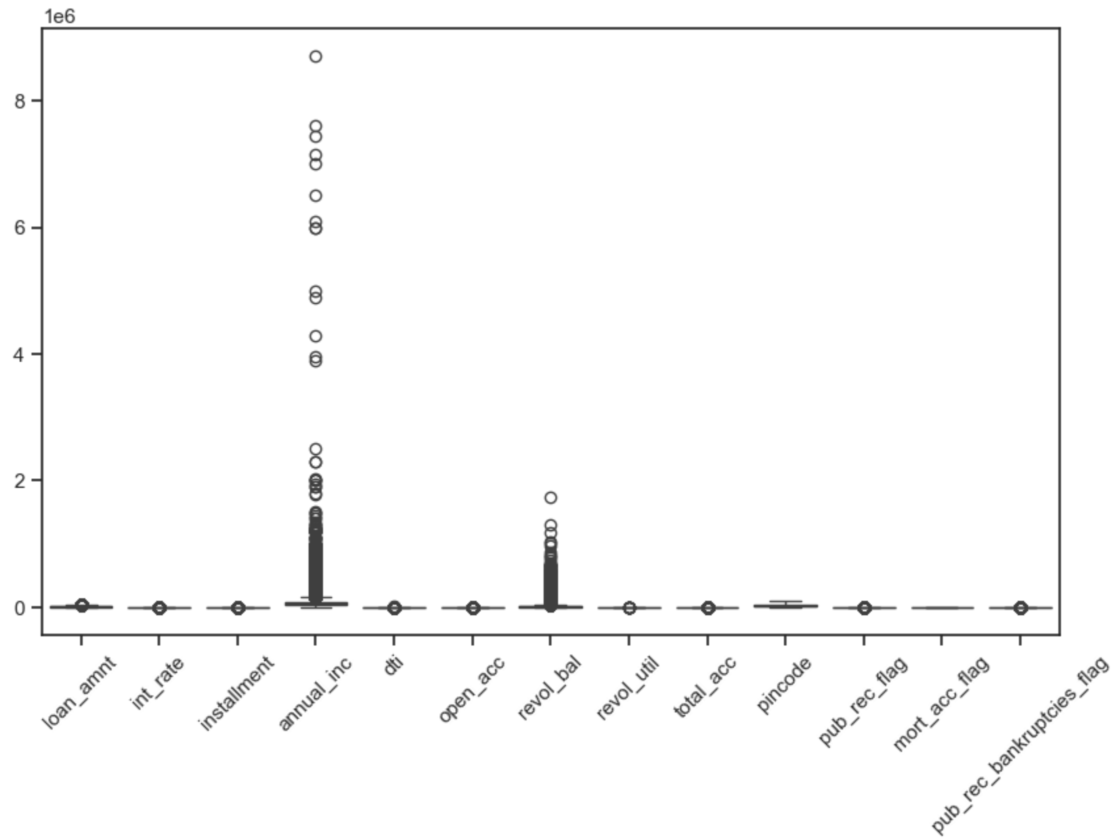
and likely not useful for predicting loan status.

0.1.8 Exploratory Data Analysis (EDA)

```
[20]: df.hist(figsize=(15, 10), color=palette[0])
plt.subplots_adjust(hspace=0.5, wspace=0.5)
plt.xticks(fontsize=8)
plt.yticks(fontsize=8)
plt.tight_layout(pad=0.5)
plt.text(0.5, 0.5, 'Histograms', horizontalalignment='center',␣
 ↪verticalalignment='center', fontsize=15, color='red', alpha=0.5)
plt.show()
```



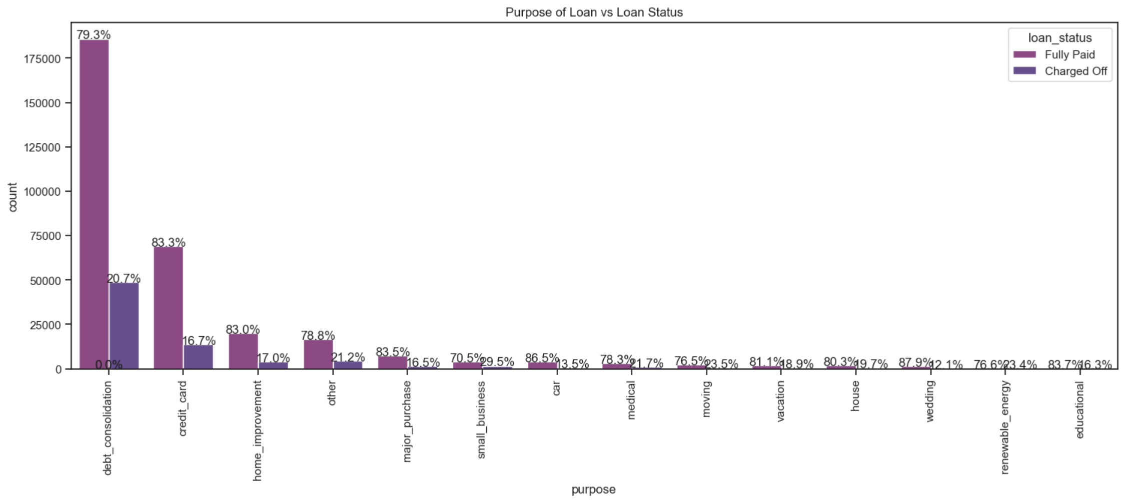

```
[21]: plt.figure(figsize=(10, 6))
      sns.boxplot(data=df)
      plt.xticks(rotation=45)
      plt.show()
```



[22]:
```
plt.figure(figsize=(18, 6))
loan_status_counts = df['loan_status'].value_counts()
sns.countplot(data=df, x='purpose', hue='loan_status',
              order=df['purpose'].value_counts().index,
              hue_order=loan_status_counts.index)
plt.xticks(rotation=90)
plt.title('Purpose of Loan vs Loan Status')

# Calculate percentages
total_counts = df['purpose'].value_counts()
for p in plt.gca().patches:
    height = p.get_height()
    total = total_counts[round(p.get_x())]
    percentage = height / total * 100
    plt.gca().text(p.get_x() + p.get_width() / 2 , height + 20, f'{percentage:.
↪1f}%', ha='center')

plt.show()
```



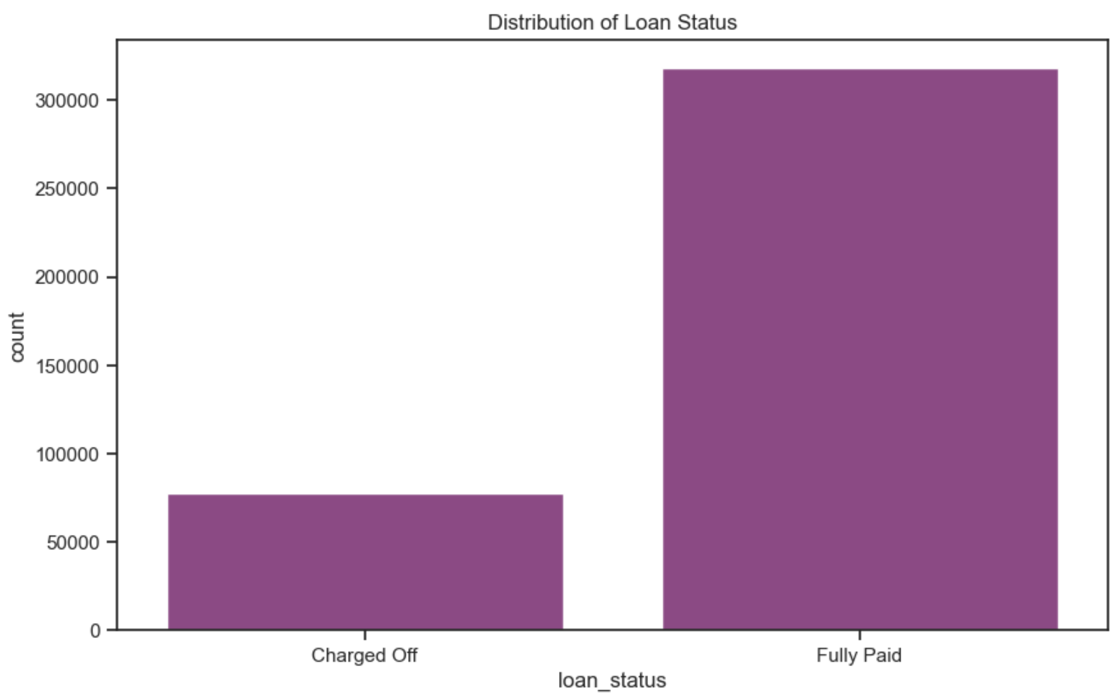
[23]:
```
loan_status_by_purpose = df[['loan_status', 'purpose']].groupby(['purpose',␣
↪'loan_status']).size().unstack().reset_index()
# loan_status_by_purpose.drop(columns=['loan_status'], inplace=True)
loan_status_by_purpose['Total'] = loan_status_by_purpose[['Charged Off','Fully␣
↪Paid']].sum(axis=1)
loan_status_by_purpose['Charged Off Percentage'] =␣
↪round(loan_status_by_purpose['Charged Off'] /␣
↪loan_status_by_purpose['Total'] * 100, 2)
loan_status_by_purpose['Fully Paid Percentage'] =␣
↪round(loan_status_by_purpose['Fully Paid'] / loan_status_by_purpose['Total']␣
↪* 100, 2)
loan_status_by_purpose.sort_values(by='Charged Off Percentage',␣
↪ascending=False, inplace=True)
loan_status_by_purpose
```

[23]:
```
loan_status              purpose  Charged Off  Fully Paid   Total  \
11                small_business         1679        4022    5701
8                         moving          670        2184    2854
10              renewable_energy           77         252     329
7                        medical          911        3285    4196
9                          other         4495       16690   21185
2             debt_consolidation        48640      185867  234507
5                          house          434        1767    2201
12                      vacation          464        1988    2452
4               home_improvement         4087       19943   24030
1                    credit_card        13874       69145   83019
6                 major_purchase         1448        7342    8790
3                    educational           42         215     257
0                            car          633        4064    4697
```

```
13                       wedding         219        1593     1812

loan_status  Charged Off Percentage  Fully Paid Percentage
11                            29.45                  70.55
8                             23.48                  76.52
10                            23.40                  76.60
7                             21.71                  78.29
9                             21.22                  78.78
2                             20.74                  79.26
5                             19.72                  80.28
12                            18.92                  81.08
4                             17.01                  82.99
1                             16.71                  83.29
6                             16.47                  83.53
3                             16.34                  83.66
0                             13.48                  86.52
13                            12.09                  87.91
```

[24]:
```python
plt.figure(figsize=(10, 6))
sns.countplot(x='loan_status', data=df)
plt.title('Distribution of Loan Status')
plt.show()
```

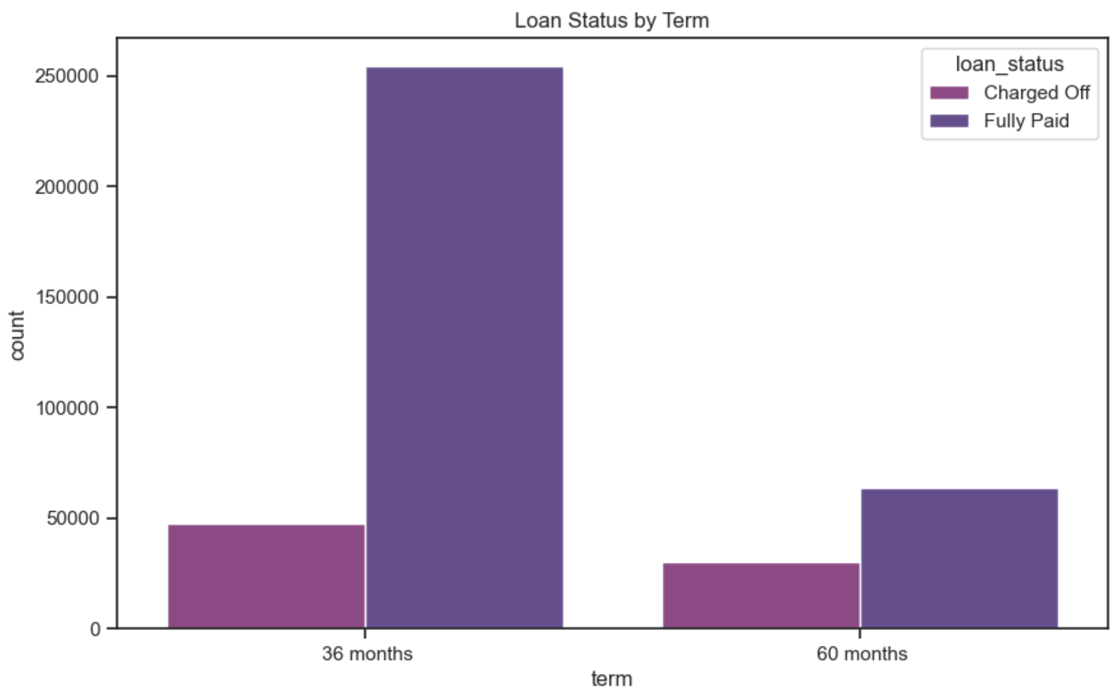


[25]:
```python
df['loan_status'].value_counts(normalize=True) * 100
```

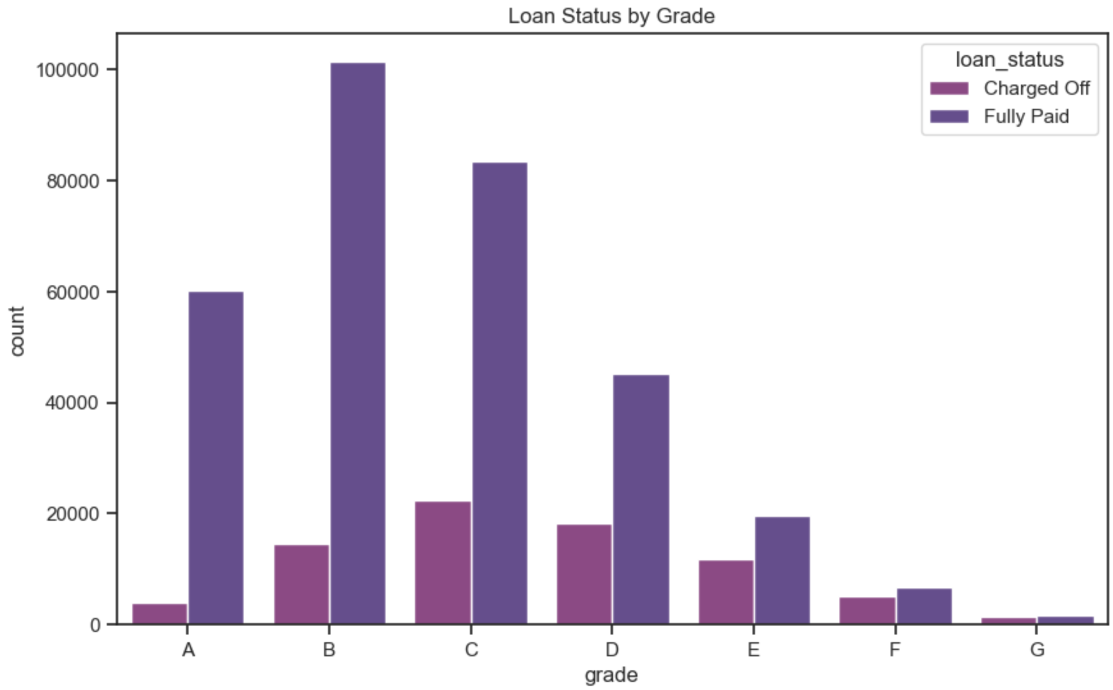
[25]:
```
loan_status
Fully Paid     80.387092
Charged Off    19.612908
Name: proportion, dtype: float64
```

What percentage of customers have fully paid their Loan Amount? - 80.38% of loans are fully paid

[26]:
```
plt.figure(figsize=(10, 6))
sns.countplot(x='term', hue='loan_status', data=df)
plt.title('Loan Status by Term')
plt.show()
```



[27]:
```
plt.figure(figsize=(10, 6))
sns.countplot(x='grade', hue='loan_status', data=df)
plt.title('Loan Status by Grade')
plt.show()
```

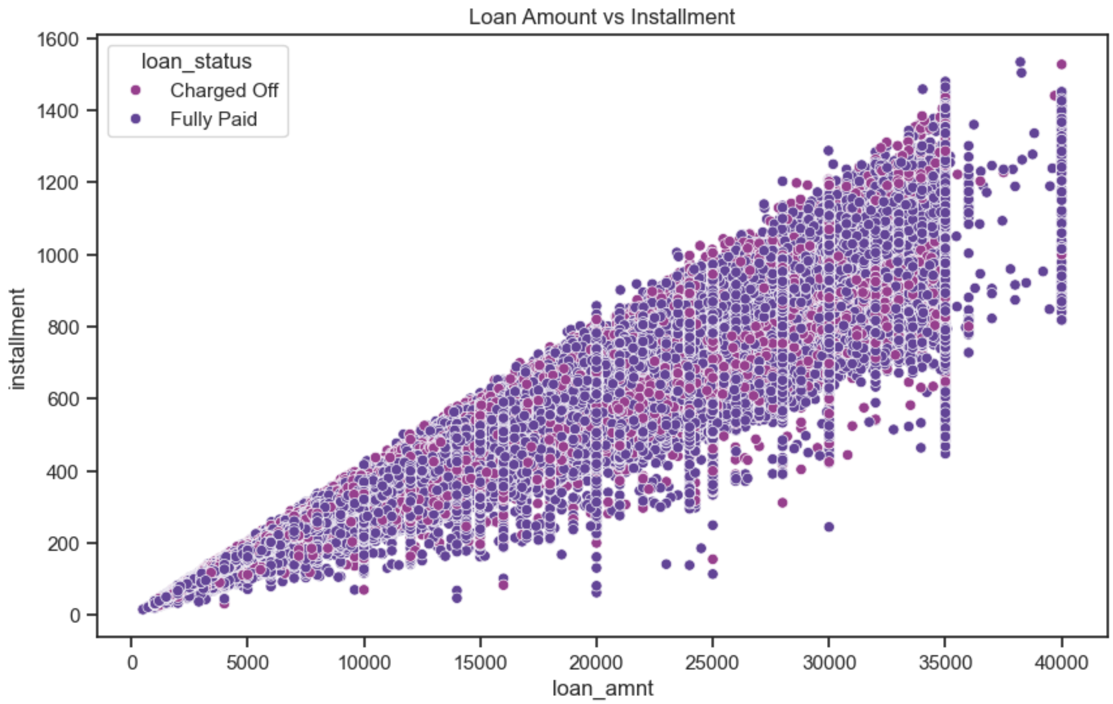


[28]:
```
df_ = df[df['grade']=='A']
df_['loan_status'].value_counts(normalize=True) * 100
```

[28]:
```
loan_status
Fully Paid      93.712122
Charged Off      6.287878
Name: proportion, dtype: float64
```

People with grades 'A' are more likely to fully pay their loan. (T/F) - True, ~94% people with 'A' Grade paid their loan

[29]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='loan_amnt', y='installment', data=df, hue='loan_status')
plt.title('Loan Amount vs Installment')
plt.show()
```

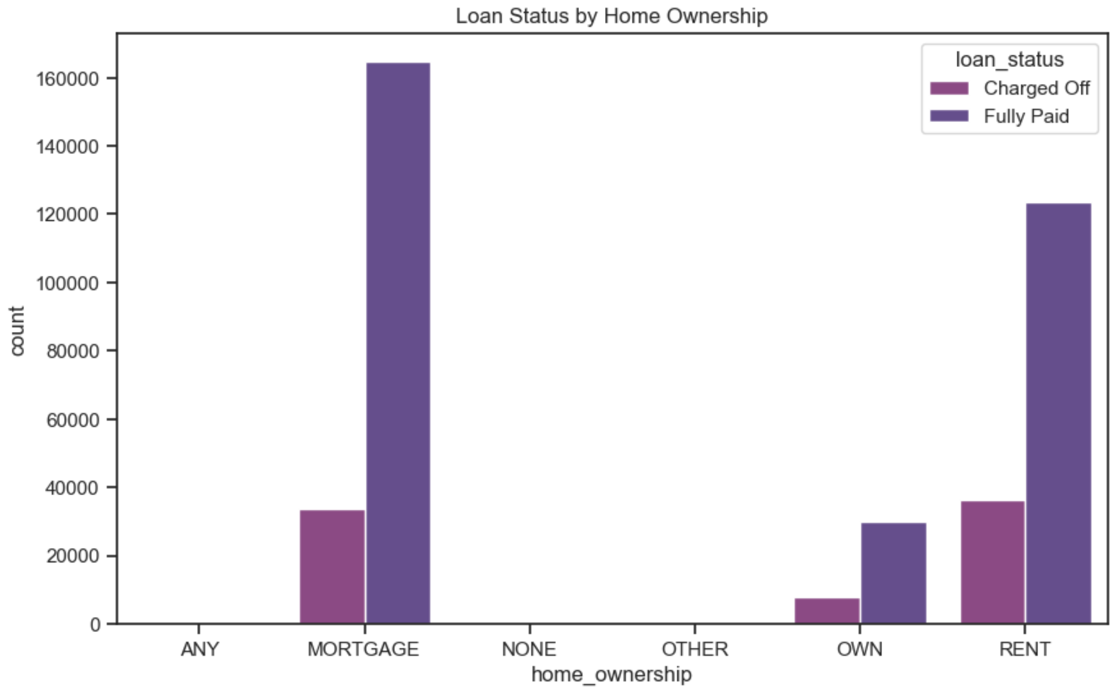


[30]:
```
correlation = df['loan_amnt'].corr(df['installment'])
print(f"correlation between Loan Amount and Installment features: ", np.
 ↪round(correlation, 2))
```

```
correlation between Loan Amount and Installment features:  0.95
```

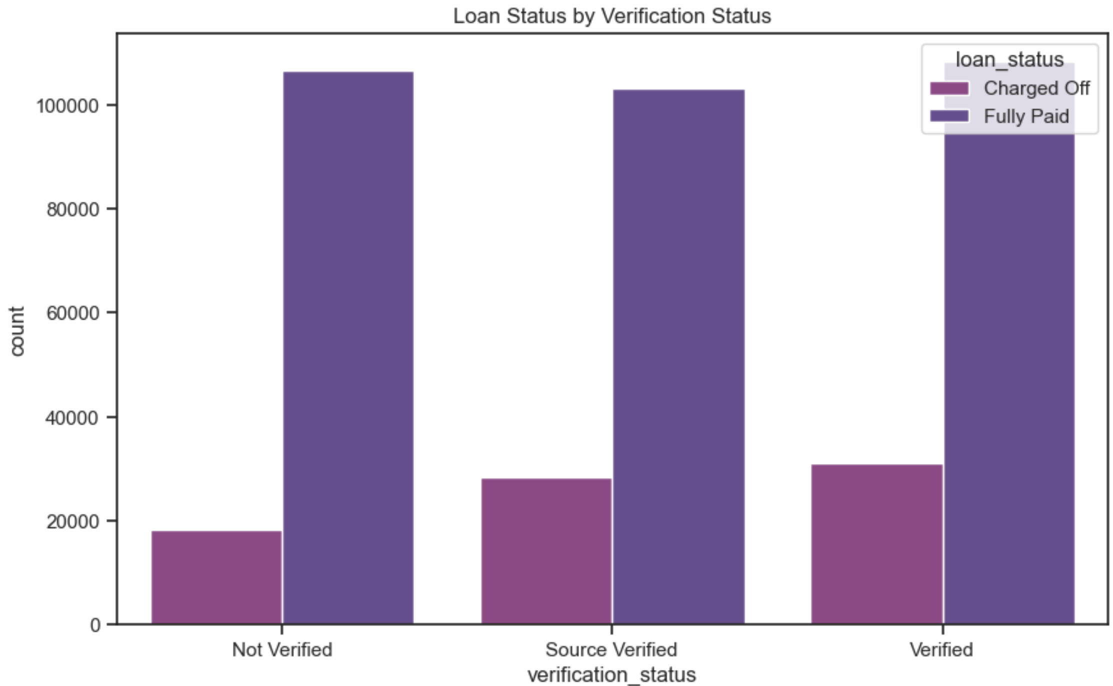
Correlation between Loan Amount and Installment features: - 0.95

[31]:
```
plt.figure(figsize=(10, 6))
sns.countplot(x='home_ownership', hue='loan_status', data=df)
plt.title('Loan Status by Home Ownership')
plt.show()
```

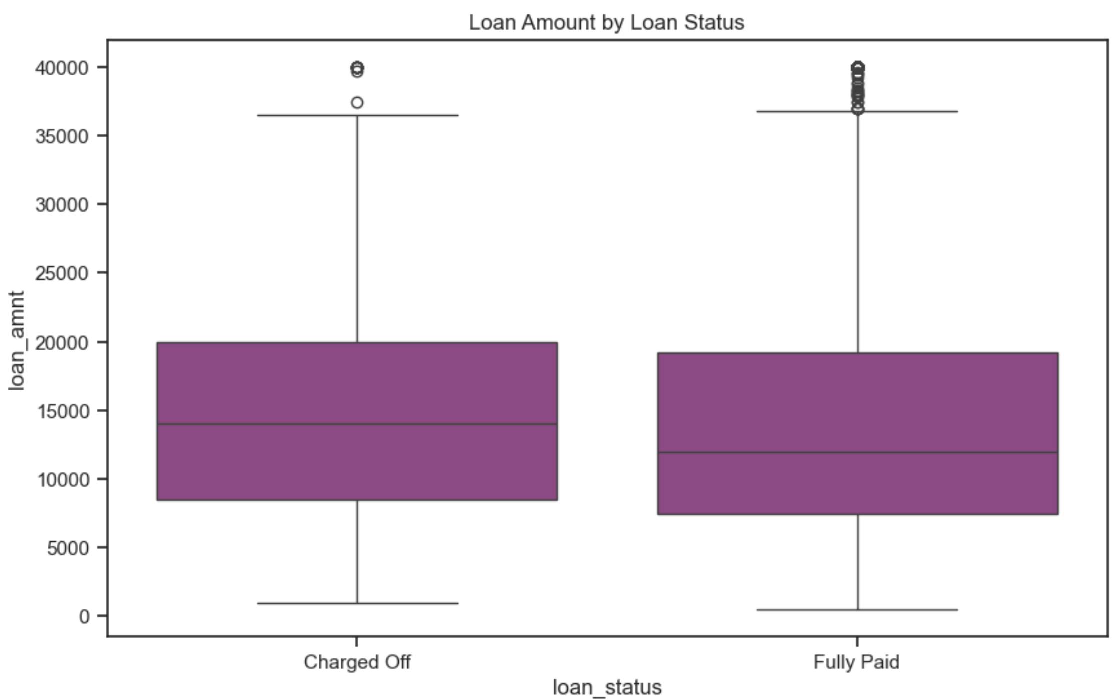


The majority of people have home ownership as - Mortage

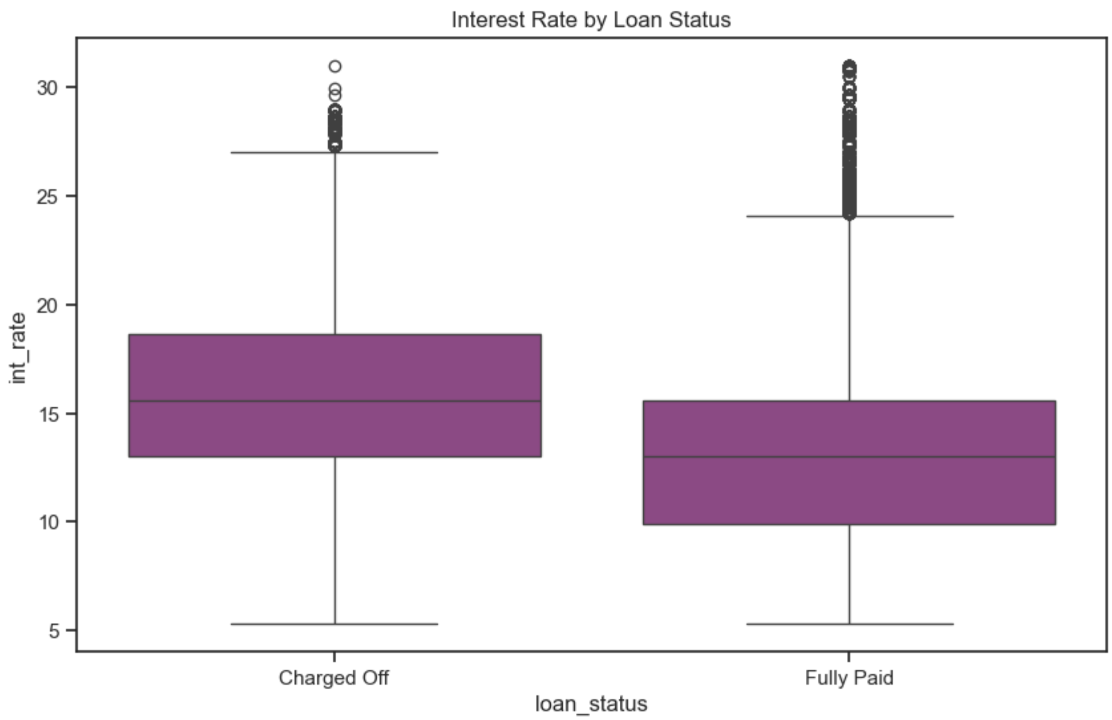
[32]:
```
plt.figure(figsize=(10, 6))
sns.countplot(x='verification_status', hue='loan_status', data=df)
plt.title('Loan Status by Verification Status')
plt.show()
```



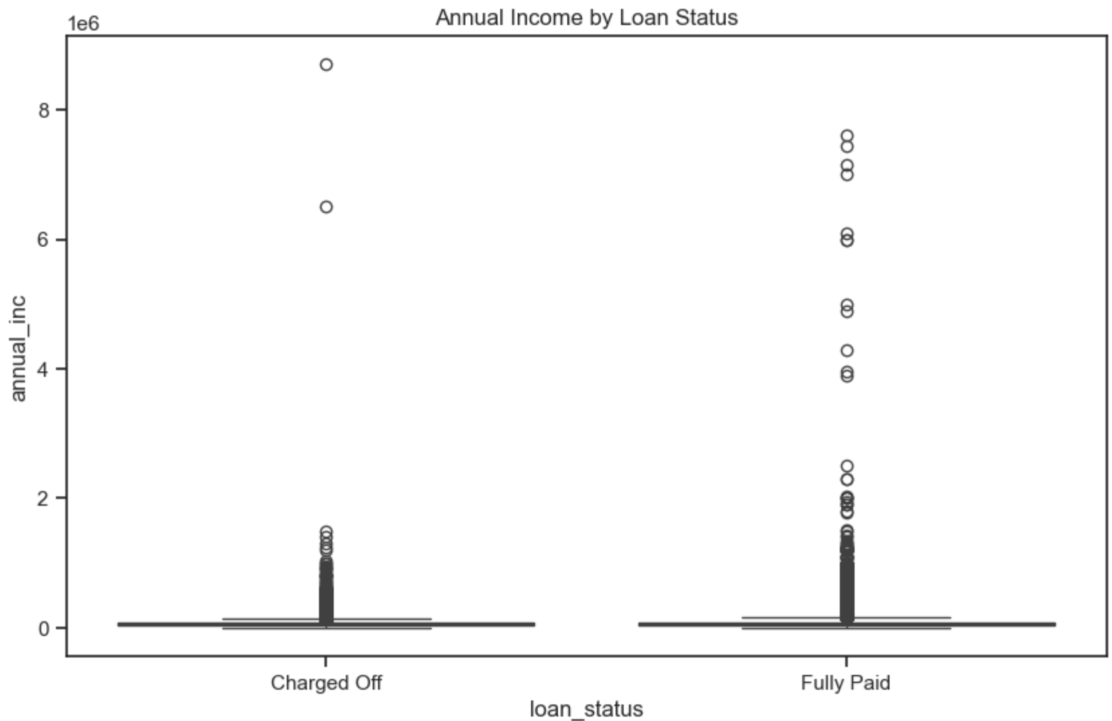
[33]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='loan_status', y='loan_amnt', data=df)
plt.title('Loan Amount by Loan Status')
plt.show()
```



[34]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='loan_status', y='int_rate', data=df)
plt.title('Interest Rate by Loan Status')
plt.show()
```



[35]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='loan_status', y='annual_inc', data=df)
plt.title('Annual Income by Loan Status')
plt.show()
```



[36]:
```
numeric_df = df.select_dtypes(include=[np.number])
corr_matrix = numeric_df.corr()

plt.figure(figsize=(14, 10))
sns.heatmap(corr_matrix, annot=True, cmap=palette, linewidths=0.5)
plt.title('Correlation Matrix Heat Map')
plt.show()
```

[37]:
```python
def detect_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] < lower_bound) | (df[column] > upper_bound)]
```

0.1.9 Insights:

- Annual income columns has higher outliers
- More loans are availed for "Debt Consolidation"
- ~80% of loans opted for "Debt Consolidation" are Fully paid, ~20% are Charged off
- Second highest reason for loan for "Credit_card"
- 0.95 correlation between Loan Amount and Installment features, Hence one can be removed during modelling

0.2 Data preparation for Modelling

[38]:
```
df = df.drop(columns=['installment'])
```

[39]:
```
X = df.drop(columns=['loan_status'])
# y = df['loan_status']
y = df['loan_status'].apply(lambda x: 1 if x == 'Fully Paid' else 0)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[40]:
```
# preprocessing steps
numeric_features = ['loan_amnt', 'int_rate', 'annual_inc', 'dti', 'open_acc',␣
↪'revol_bal', 'revol_util', 'total_acc','pub_rec_flag', 'mort_acc_flag',␣
↪'pub_rec_bankruptcies_flag', 'pincode']

numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_features = ['term', 'grade', 'sub_grade', 'home_ownership',␣
↪'verification_status', 'purpose', 'initial_list_status', 'application_type',␣
↪'state']
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(sparse_output=False, drop='first'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

X_train = preprocessor.fit_transform(X_train)
```

[41]:
```
model = LogisticRegression()
model.fit(X_train, y_train)
```

[41]:
```
LogisticRegression()
```

[42]:
```
X_test = preprocessor.transform(X_test)
y_pred = model.predict(X_test)
```

[43]:
```
# Model Evaluation
from sklearn.metrics import accuracy_score, classification_report
```

```python
print("Accuracy: ", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy:  0.8361235259955054
              precision    recall  f1-score   support

           0       0.69      0.31      0.43     15577
           1       0.85      0.97      0.90     63629

    accuracy                           0.84     79206
   macro avg       0.77      0.64      0.66     79206
weighted avg       0.82      0.84      0.81     79206
```

[44]:

```python
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score

print("Precision: ", precision_score(y_test, y_pred))
print("Recall: ", recall_score(y_test, y_pred))
print("F1 Score: ", f1_score(y_test, y_pred))
print("ROC AUC Score: ", roc_auc_score(y_test, y_pred))
```

```
Precision:  0.8507208442394781
Recall:  0.9654088544531582
F1 Score:  0.9044435937453988
ROC AUC Score:  0.6367135432309445
```

Thinking from a bank's perspective, which metric should our primary focus be on: - ROC AUC - Precision - Recall - F1 Score

Bank's Perspective From a bank's perspective, the primary focus should typically be on minimizing financial risk while maximizing opportunities for profit. This involves a delicate balance:

- *Minimizing False Negatives (Recall)*:
  - Missing a defaulter can lead to financial losses, so high recall is important.
- *Minimizing False Positives (Precision)*:
  - Incorrectly predicting a non-defaulter as a defaulter can lead to lost business opportunities, so high precision is also important.

Recommended Primary Metric: *F1 Score*

- F1 Score is recommended as the primary metric because it balances precision and recall, which is essential for a bank's loan approval model. By focusing on the F1 Score, the bank can ensure that the model is effective at identifying defaulters while also minimizing the rejection of good loan applicants.

0.2.1 Model Performance Metrics:

- Precision: 0.8507

- Recall: 0.9655
- F1 Score: 0.9045
- ROC AUC Score: 0.6368

Understanding Precision and Recall

Precision (0.8507): This indicates that 85.07% of the loans predicted as defaulters are actually defaulters. A precision of 0.8507 means that there is a 14.93% rate of false positives (non-defaulters incorrectly predicted as defaulters).

Recall (0.9655): This indicates that 96.55% of actual defaulters are correctly identified by the model. A recall of 0.9655 means that there is a 3.45% rate of false negatives (defaulters incorrectly predicted as non-defaulters).

How does the gap in precision and recall affect the bank?

The gap between precision and recall indicates that while your model is very good at identifying defaulters (high recall), it also incorrectly flags some non-defaulters as defaulters (lower precision).

[45]:
```
# Feature Importance
importance = model.coef_[0]
numeric_features_list = preprocessor.transformers_[0][2]
categorical_features_list = preprocessor.named_transformers_['cat']['onehot'].
 ↪get_feature_names_out(categorical_features)
all_features = list(numeric_features_list) + list(categorical_features_list)
feature_importance = pd.DataFrame({'Feature': all_features, 'Importance':␣
 ↪importance})
feature_importance = feature_importance.sort_values(by='Importance',␣
 ↪ascending=False)
feature_importance
```

[45]:

	Feature	Importance
75	application_type_JOINT	1.754345
53	home_ownership_MORTGAGE	1.009141
56	home_ownership_OWN	0.859076
57	home_ownership_RENT	0.749367
55	home_ownership_OTHER	0.492414
72	purpose_wedding	0.425202
1	int_rate	0.355426
2	annual_inc	0.185882
54	home_ownership_NONE	0.135111
7	total_acc	0.118117
101	state_MN	0.108354
64	purpose_house	0.090313
23	sub_grade_B1	0.069224
5	revol_bal	0.060948
112	state_NY	0.057535
86	state_DE	0.054669
124	state_VT	0.054387
82	state_CA	0.047774

83	state_CO	0.047229
93	state_IN	0.046346
108	state_NH	0.038223
120	state_TN	0.031719
9	mort_acc_flag	0.029253
113	state_OH	0.027078
114	state_OK	0.023852
77	state_AK	0.023454
81	state_AZ	0.021813
123	state_VA	0.021500
115	state_OR	0.019864
89	state_HI	0.019571
85	state_DC	0.018238
99	state_ME	0.017817
100	state_MI	0.013318
106	state_ND	0.011249
96	state_LA	0.010096
105	state_NC	0.008725
118	state_SC	0.007135
73	initial_list_status_w	0.006770
102	state_MO	0.006451
87	state_FL	0.005309
121	state_TX	0.004956
10	pub_rec_bankruptcies_flag	0.004553
107	state_NE	0.000334
90	state_IA	-0.000048
109	state_NJ	-0.000378
117	state_RI	-0.000395
78	state_AL	-0.000963
92	state_IL	-0.001433
94	state_KS	-0.002406
125	state_WA	-0.009146
119	state_SD	-0.009293
98	state_MD	-0.009819
111	state_NV	-0.011719
110	state_NM	-0.012393
122	state_UT	-0.012841
126	state_WI	-0.013895
104	state_MT	-0.019889
8	pub_rec_flag	-0.023633
80	state_AR	-0.024443
95	state_KY	-0.027286
71	purpose_vacation	-0.028110
79	state_AP	-0.031839
91	state_ID	-0.035145
97	state_MA	-0.042179
76	state_AE	-0.043678

60	purpose_credit_card	-0.048063
84	state_CT	-0.050708
116	state_PA	-0.059509
103	state_MS	-0.063033
88	state_GA	-0.066932
127	state_WV	-0.067203
6	revol_util	-0.067940
28	sub_grade_C1	-0.070853
24	sub_grade_B2	-0.071530
0	loan_amnt	-0.072828
68	purpose_other	-0.076083
61	purpose_debt_consolidation	-0.083264
65	purpose_major_purchase	-0.089890
128	state_WY	-0.114570
59	verification_status_Verified	-0.116311
4	open_acc	-0.118873
63	purpose_home_improvement	-0.139503
58	verification_status_Source Verified	-0.161955
67	purpose_moving	-0.210400
66	purpose_medical	-0.213280
62	purpose_educational	-0.228592
25	sub_grade_B3	-0.254894
74	application_type_INDIVIDUAL	-0.259876
29	sub_grade_C2	-0.264055
69	purpose_renewable_energy	-0.271468
33	sub_grade_D1	-0.277595
19	sub_grade_A2	-0.331528
38	sub_grade_E1	-0.345315
43	sub_grade_F1	-0.368631
12	term_ 60 months	-0.375852
34	sub_grade_D2	-0.400465
26	sub_grade_B4	-0.406340
30	sub_grade_C3	-0.414956
35	sub_grade_D3	-0.455671
39	sub_grade_E2	-0.457699
3	dti	-0.474969
70	purpose_small_business	-0.488873
48	sub_grade_G1	-0.494554
44	sub_grade_F2	-0.506619
31	sub_grade_C4	-0.508829
49	sub_grade_G2	-0.526377
40	sub_grade_E3	-0.559836
36	sub_grade_D4	-0.572010
27	sub_grade_B5	-0.572913
51	sub_grade_G4	-0.592506
20	sub_grade_A3	-0.597591
32	sub_grade_C5	-0.611243

```
45                          sub_grade_F3   -0.653658
37                          sub_grade_D5   -0.656049
41                          sub_grade_E4   -0.660050
21                          sub_grade_A4   -0.702664
46                          sub_grade_F4   -0.721955
42                          sub_grade_E5   -0.738930
52                          sub_grade_G5   -0.763358
11                               pincode   -0.854771
47                          sub_grade_F5   -0.882899
22                          sub_grade_A5   -0.957883
50                          sub_grade_G3   -1.017774
13                               grade_B   -1.236452
14                               grade_C   -1.869935
15                               grade_D   -2.361790
16                               grade_E   -2.761831
17                               grade_F   -3.133763
18                               grade_G   -3.394571
```

Which were the features that heavily affected the outcome? * application_type * home_ownership * purpose * int_rate * annual_inc * total_acc * grade * sub_grade * state

Will the results be affected by geographical location? - Yes. States features are listed in the top features

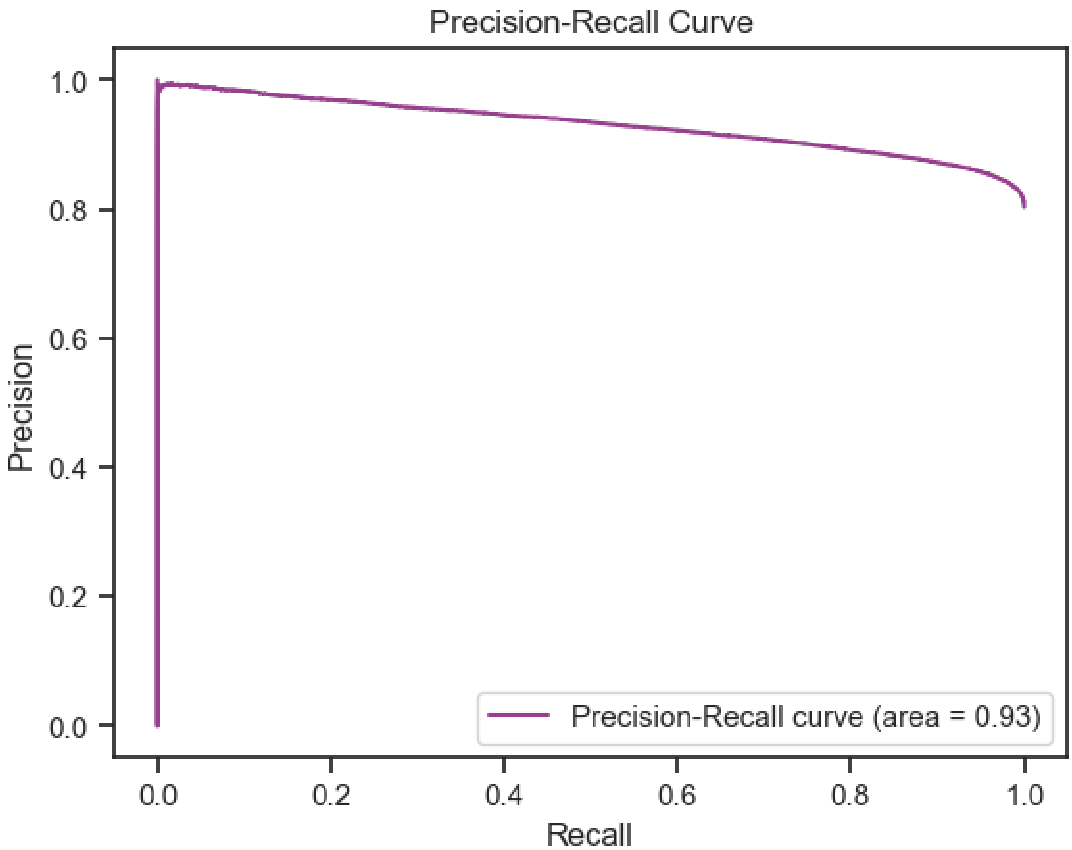
[46]:
```python
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve, auc

# Predict probabilities
y_probs = model.predict_proba(X_test)[:, 1]

# Calculate precision-recall curve
precision, recall, _ = precision_recall_curve(y_test, y_probs)

# Calculate AUC
pr_auc = auc(recall, precision)

# Plot the precision-recall curve
plt.figure()
plt.plot(recall, precision, label=f'Precision-Recall curve (area = {pr_auc:.
↪2f})')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc='best')
plt.show()
```



0.2.2 Actionable Insights

High Recall, Lower Precision:

Insight: Your model is highly effective at identifying defaulters (high recall) but has a moderate rate of false positives (lower precision).

Impact: While the bank minimizes the risk of defaults, it may also be rejecting a significant number of creditworthy applicants, leading to lost business opportunities.

Geographical Variability:

Insight: Geographical differences can significantly impact loan default rates due to varying economic conditions, cost of living, and access to financial services.

Impact: Ignoring geographical factors may lead to suboptimal loan approval decisions and higher default rates in certain regions.

0.2.3 Recommendations

Segmented Modeling:

Develop separate models for different geographical regions or clusters with similar characteristics. This allows for more tailored predictions and better handling of regional variations.

Post-Processing Rules and Manual Review:

Implement post-processing rules or manual reviews for borderline cases to reduce false positives without significantly impacting recall.

[ ]:

```

```