LoanTap Logistic Regression Business Case

LoanTap is at the forefront of offering tailored financial solutions to millennials.

- Their innovative approach seeks to harness data science for refining their credit underwriting process.
- The focus here is the Personal Loan segment. A deep dive into the dataset can reveal patterns in borrower behavior and creditworthiness.

Dataset Explanation: LoanTapData.csv

1. loan_amnt: Amount borrower applied for.
2. term: Loan duration (36 or 60 months).
3. int_rate: Interest rate on loan.
4. installment: Monthly repayment amount.
5. grade: LoanTap assigned loan grade (Risk ratings by LoanTap.)
6. sub_grade: LoanTap assigned loan grade (Risk ratings by LoanTap.)
7. emp_title: Borrower's job title.
8. emp_length: Duration of borrower's employment (0-10 years).
9. home_ownership: Borrower's housing situation (own, rent, etc.).
10. annual_inc: Borrower's yearly income.
11. verification_status: Whether borrower's income was verified.
12. issue_d: Loan issuance month.
13. loan_status: Current status of the loan.
14. purpose: Borrower's reason for the loan.
15. title: The loan's title provided by the borrower.
16. dti (Debt-to-Income ratio): Monthly debt vs. monthly income ratio.
17. earliest_cr_line: Date of borrower's oldest credit account.
18. open_acc: Number of borrower's active credit lines.
19. pub_rec: Negative records on borrower's public credit profile.
20. revol_bal: Total credit balance.
21. revol_util: Usage percentage of 'revolving' accounts like credit cards.
22. total_acc: Total number of borrower's credit lines.
23. initial_list_status: Loan's first category ('W' or 'F').
24. application_type: Individual or joint application.
25. mort_acc: Number of borrower's mortgages.
26. pub_rec_bankruptcies: Bankruptcy records for borrower.
27. Address: Borrower's location.

1. Define Problem Statement and perform Exploratory Data Analysis

In [316]:
```
# Importing Libraries
#Data processing
import numpy as np
import pandas as pd

#Data Visualisation
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
%matplotlib inline

#Seting option for full column view of Data
pd.set_option('display.max_columns', None)

#Stats & model building
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler , MinMaxScaler,LabelEncoder
from sklearn.linear_model import LinearRegression
from imblearn.over_sampling import SMOTE
from sklearn.metrics import (accuracy_score, confusion_matrix,
                             roc_curve, auc, ConfusionMatrixDisplay,
                             f1_score, recall_score,
                             precision_score, precision_recall_curve,
                             average_precision_score, classification_report)

#Hide warnings
import warnings
warnings.filterwarnings("ignore")
```

In [389]:
```
import category_encoders as ce
```

1.a Definition of problem

- **Credit Underwriting** is the process by which a financial organization decides to accept the risk of lending to a particular person or company.Our objective is to determine the creditworthiness of MSMEs as well as individuals to grant Personal Loan
- Given a set of attributes for an Individual, determine if a credit line should be extended to them.
- The main challenge is to minimise the risk of NPAs by flagging defaulters while maximising opportunity to earn interest by disbursing loans to as many customers as possible.

1.b Observations on Data

In [279]:
```
df = pd.read_csv(r"C:\Users\krama\Downloads\logistic_regression.csv")
```

In [280]:
```
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                 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
```

- ~4M records with 27 features are present in the given data

- dtypes: float64(12)(numerical), object(15)(categorical)

In [281]:
```
df.isna().sum().sort_values(ascending = False)
```

Out[281]:
```
mort_acc                37795
emp_title               22927
emp_length              18301
title                    1755
pub_rec_bankruptcies      535
revol_util                276
loan_amnt                   0
dti                         0
application_type            0
initial_list_status         0
total_acc                   0
revol_bal                   0
pub_rec                     0
open_acc                    0
earliest_cr_line            0
purpose                     0
term                        0
loan_status                 0
issue_d                     0
verification_status         0
annual_inc                  0
home_ownership              0
sub_grade                   0
grade                       0
installment                 0
int_rate                    0
address                     0
dtype: int64
```

- 6 columns have null values:
  - mort_acc 37795
  - emp_title 22927
  - emp_length 18301
  - title 1755
  - pub_rec_bankruptcies 535
  - revol_util 276

In [283]:
```
numeric_data = df.select_dtypes(include=[np.number])
categorical_data = df.select_dtypes(exclude=[np.number])
```

In [284]:
```
numeric_data.head()
```

Out[284]:

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util	total
0	10000.0	11.44	329.48	117000.0	26.24	16.0	0.0	36369.0	41.8	
1	8000.0	11.99	265.68	65000.0	22.05	17.0	0.0	20131.0	53.3	
2	15600.0	10.49	506.97	43057.0	12.79	13.0	0.0	11987.0	92.2	
3	7200.0	6.49	220.65	54000.0	2.60	6.0	0.0	5472.0	21.5	
4	24375.0	17.27	609.33	55000.0	33.95	13.0	0.0	24584.0	69.8	

In [285]:
```
categorical_data.head()
```

Out[285]:

	term	grade	sub_grade	emp_title	emp_length	home_ownership	verification_status	issue_c
0	36 months	B	B4	Marketing	10+ years	RENT	Not Verified	Jan 201
1	36 months	B	B5	Credit analyst	4 years	MORTGAGE	Not Verified	Jan 201
2	36 months	B	B3	Statistician	< 1 year	RENT	Source Verified	Jan 201
3	36 months	A	A2	Client Advocate	6 years	RENT	Not Verified	Nov 201
4	60 months	C	C5	Destiny Management Inc.	9 years	MORTGAGE	Verified	Apr 201

In [286]:
```
## number of unique categories in each categorical columns
df.select_dtypes(exclude=[np.number]).nunique().sort_values()
```

Out[286]:
```
term                        2
loan_status                 2
initial_list_status         2
verification_status         3
application_type            3
home_ownership              6
grade                       7
emp_length                 11
purpose                    14
sub_grade                  35
issue_d                   115
earliest_cr_line          684
title                   48817
emp_title              173105
address                393700
dtype: int64
```

In [287]:
```
## number of unique categories in each categorical columns
df.select_dtypes(include=[np.number]).nunique().sort_values()
```

Out[287]:
```
pub_rec_bankruptcies        9
pub_rec                    20
mort_acc                   33
open_acc                   61
total_acc                 118
int_rate                  566
revol_util               1226
loan_amnt                1397
dti                      4262
annual_inc              27197
revol_bal               55622
installment             55706
dtype: int64
```

In [288]:
```
##statistical summary
df.describe()
```

Out[288]:

e	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revo
)	396030.000000	3.960300e+05	396030.000000	396030.000000	396030.000000	3.960300e+05	395754.0
)	431.849698	7.420318e+04	17.379514	11.311153	0.178191	1.584454e+04	53.7
7	250.727790	6.163762e+04	18.019092	5.137649	0.530671	2.059184e+04	24.4
)	16.080000	0.000000e+00	0.000000	0.000000	0.000000	0.000000e+00	0.0
)	250.330000	4.500000e+04	11.280000	8.000000	0.000000	6.025000e+03	35.8
)	375.430000	6.400000e+04	16.910000	10.000000	0.000000	1.118100e+04	54.8
)	567.300000	9.000000e+04	22.980000	14.000000	0.000000	1.962000e+04	72.9
)	1533.810000	8.706582e+06	9999.000000	90.000000	86.000000	1.743266e+06	892.3

In [289]:
```
df.describe(include="object")
```

Out[289]:

	term	grade	sub_grade	emp_title	emp_length	home_ownership	verification_status	iss
count	396030	396030	396030	373103	377729	396030	396030	39
unique	2	7	35	173105	11	6	3	
top	36 months	B	B3	Teacher	10+ years	MORTGAGE	Verified	
freq	302005	116018	26655	4389	126041	198348	139563	1

- A charge-off means a lender or creditor has written the account off as a loss, and the account is closed to future charges. It may be sold to a debt buyer or transferred to a collection agency.

- Most of the loans are for 36 months
- Most of the applicants are teachers and have 10+ years of experience
- Most of the homes are mortgaged
- Most of the loans are issued in Oct 2014
- Most of the loans are fully paid
- Most of the loans are taken for debt consolidation
- Most of the loan applications are from individual applicants

1.b.1 Data Cleaning

In [290]:
```
## Splitting the addrees column into address , city, state, zip code
```

In [291]:
```
df.tail()
```

Out[291]:

i	earliest_cr_line	open_acc	pub_rec	revol_bal	revol_util	total_acc	initial_list_status	application_typ
3	Nov-2004	6.0	0.0	1990.0	34.3	23.0	w	INDIVIDUA
5	Feb-2006	6.0	0.0	43263.0	95.7	8.0	f	INDIVIDUA
6	Mar-1997	15.0	0.0	32704.0	66.9	23.0	f	INDIVIDUA
3	Nov-1990	9.0	0.0	15704.0	53.8	20.0	f	INDIVIDUA
2	Sep-1998	3.0	0.0	4292.0	91.3	19.0	f	INDIVIDUA

In [303]:
```
new = df["address"].str.split("\r\n", n=1, expand=True)
df["address1"] = new[0]
df["address2"] = new[1]
new1 = df["address2"].str.split(",", n=1, expand=True)
new2= new1[1].str.split(expand=True)
df["state"]= new2[0]
df["city"] = new1[0]
df["zip_code"]= new2[1]
df.drop(columns=["address","address2"],inplace =True)
```

In [304]:
```
df.tail()
```

Out[304]:

pen_acc	pub_rec	revol_bal	revol_util	total_acc	initial_list_status	application_type	mort_acc	pub_r
6.0	0.0	1990.0	34.3	23.0	w	INDIVIDUAL	0.0	
6.0	0.0	43263.0	95.7	8.0	f	INDIVIDUAL	1.0	
15.0	0.0	32704.0	66.9	23.0	f	INDIVIDUAL	0.0	
9.0	0.0	15704.0	53.8	20.0	f	INDIVIDUAL	5.0	
3.0	0.0	4292.0	91.3	19.0	f	INDIVIDUAL	NaN	

In [305]:
```
##Converting textual employment lengths into numerical values
df.loc[df["emp_length"] =="< 1 year","emp_length"]=1
df.loc[df["emp_length"] =="10+ years","emp_length"]=10
df.loc[df["emp_length"] =="2 years","emp_length"]=2
df.loc[df["emp_length"] =="3 years","emp_length"]=3
df.loc[df["emp_length"] =="5 years","emp_length"]=5
df.loc[df["emp_length"] =="1 year","emp_length"]=1
df.loc[df["emp_length"] =="4 years","emp_length"]=4
df.loc[df["emp_length"] =="6 years","emp_length"]=6
df.loc[df["emp_length"] =="7 years","emp_length"]=7
df.loc[df["emp_length"] =="8 years","emp_length"]=8
df.loc[df["emp_length"] =="9 years","emp_length"]=9
df.loc[df["emp_length"] =="Not Provided","emp_length"]=0
```

In [294]:
```
df.info()
```

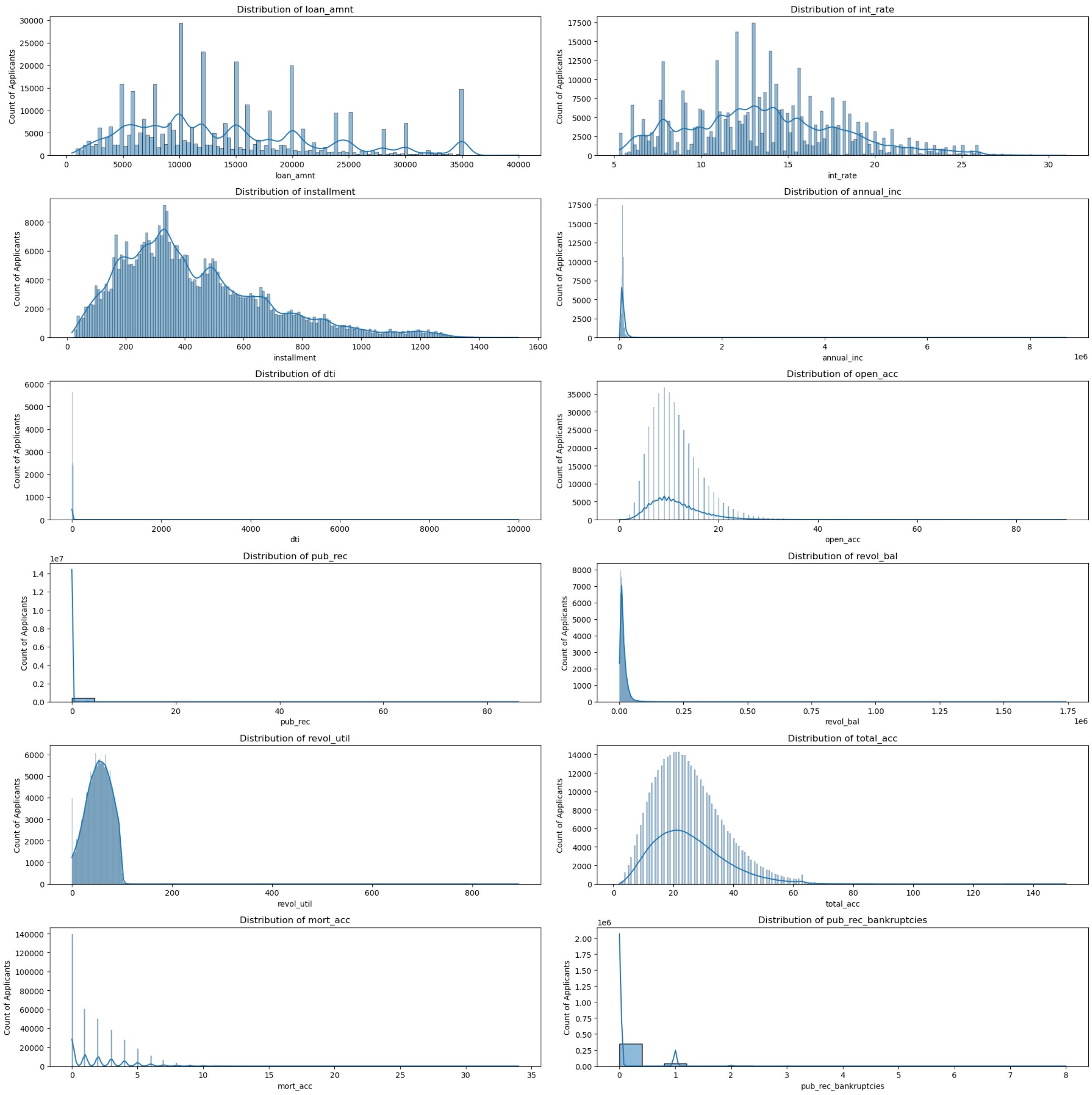
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 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                 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
```

1.c Univariate Analysis

In [295]:
```
# plt.figure(figsize=(20,20))
# sns.pairplot(df)
# plt.title('Relationship b/w Features')
# plt.show();
```

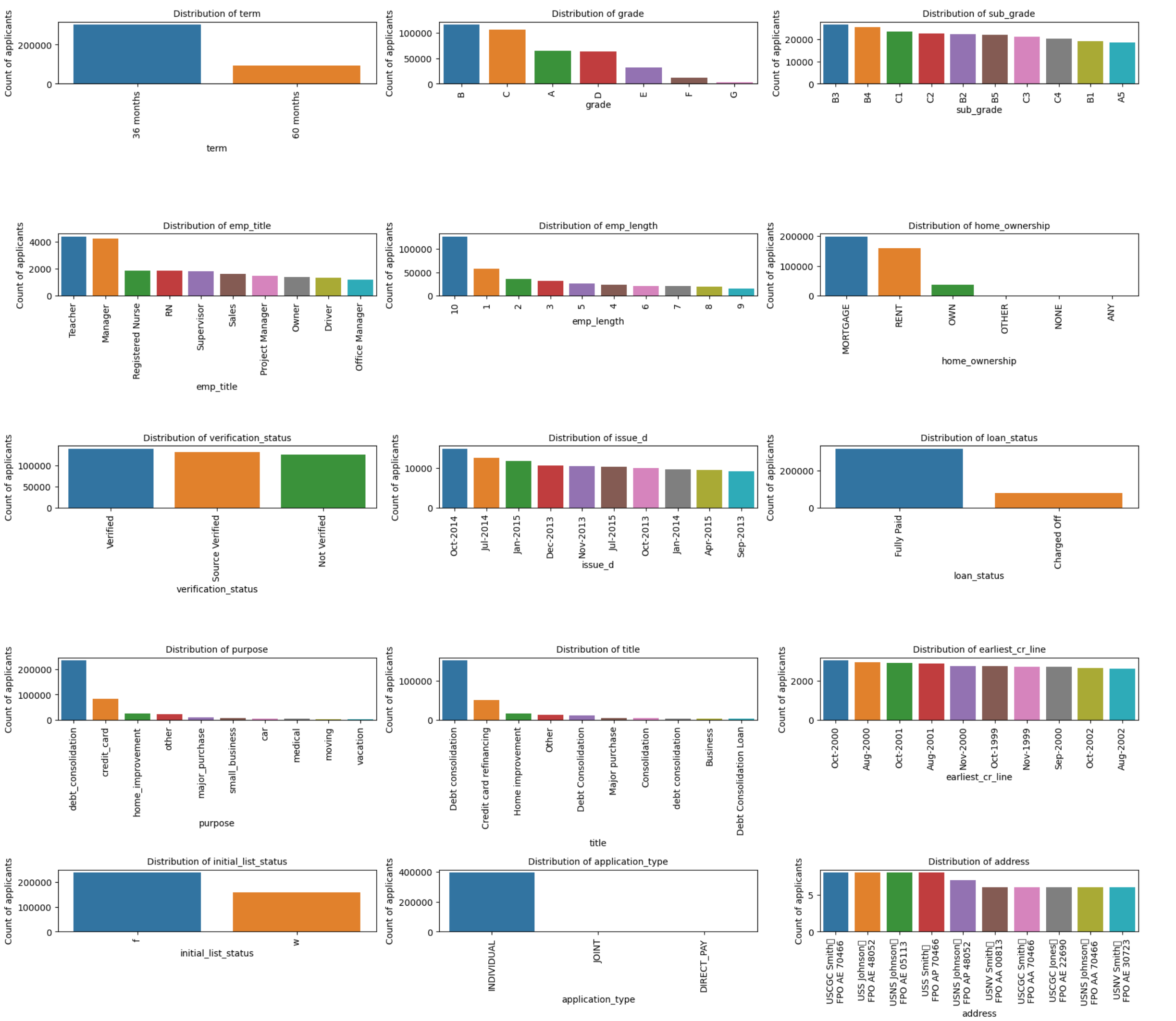
- loan_amt and installment are positively correlated to each other
- open_acc and total_acc are positively correlated to each other
- total_acc and mort_acc are positively correlated to each other

In [306]:
```
# Distribution of continuous numerical features
numeric_cols = df.select_dtypes(include=['float','int']).columns.tolist()

plt.figure(figsize=(20,20))
i=1
for col in numeric_cols:
  ax=plt.subplot(6,2,i)
  sns.histplot(data=df[col], kde=True)
  plt.title(f'Distribution of {col}')
  plt.xlabel(col)
  plt.ylabel('Count of Applicants')
  i += 1

plt.tight_layout()
plt.show();
```



- Only total_acc and open_acc follow normal distribution , rest of the distribution are not uniform and are positively skewed.

In [297]:
```
# Distribution of categorical variables
# cat_cols=['term',
#   'grade',
#   'emp_length',
#   'home_ownership',
#   'verification_status',
#   'loan_status',
#   'purpose',
#   'initial_list_status',
#   'application_type']

cat_cols = df.select_dtypes(include=['bool','category','object']).columns.tolist(
plt.figure(figsize=(18,18))
i=1

for col in cat_cols:
    ax = plt.subplot(6,3,i)
    sns.countplot(x=df[col],order=df[col].value_counts().iloc[:10].index)
    plt.title(f'Distribution of {col}', fontsize=10)
    plt.xlabel(col)
    plt.xticks(rotation='vertical')
    plt.ylabel('Count of applicants')
    i+=1

plt.tight_layout()
plt.show();
```



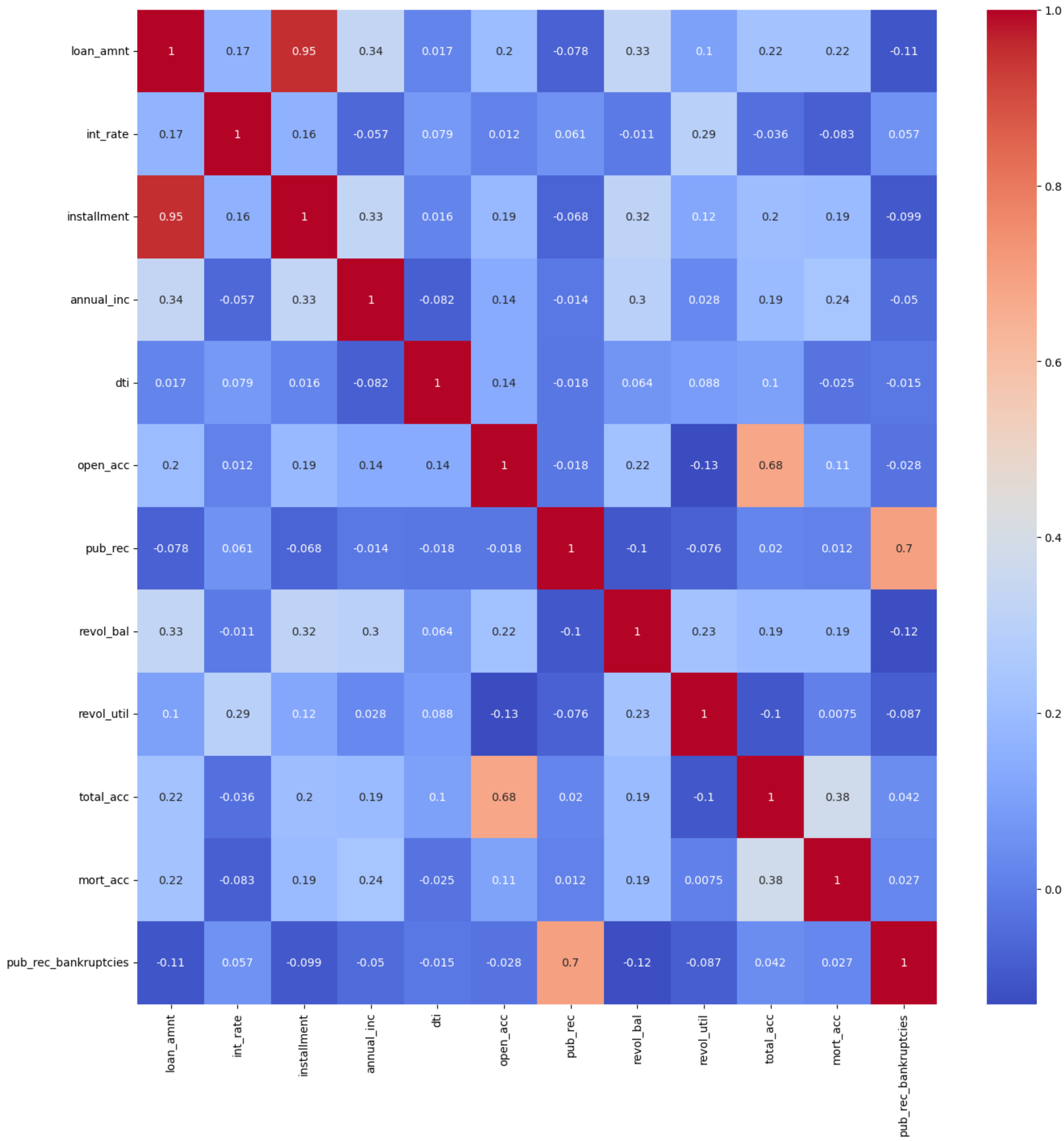
- Most of the applicants have 36 month as term

- Grade B & C , subgrade B3 and B4 are most common
- Teacher and Manager are the top emp_title and have been working for more than 10 years
- Most of the homes are mortgages
- Most of the loans are for debt consolidation
- Most of the loans are fully paid
- NJ , WI, LA , NV ,AK are top states which have most of the loan applicants
- 70466,30723,22690,48052 are the most common zip codes

1.d BivariateAnalysis

Relationship between numerical feature

In [307]:
```
plt.figure(figsize=(16,16))
sns.heatmap(df.corr(),annot=True,cmap="coolwarm")
plt.show()
```



1. loan_amnt and installment are perfectly correlated
2. total_acc is highly correlated with open_acc

3. total_acc is moderately correlated with mort_acc We can remove some of these correlated features to avoid multicolinearity

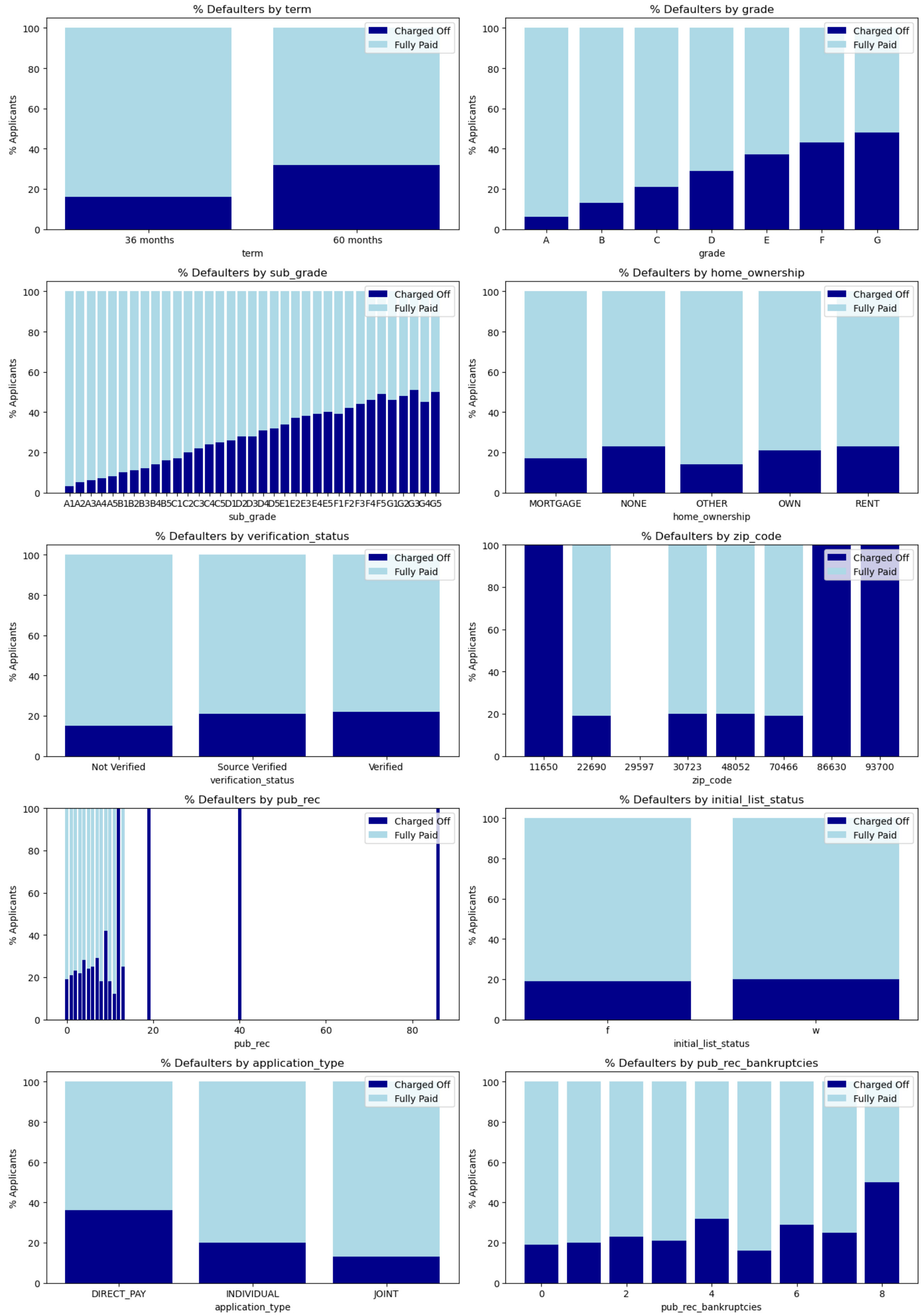
Relationship between categorical and numrerical featues

In [302]:
```
df.head()
```

Out[302]:

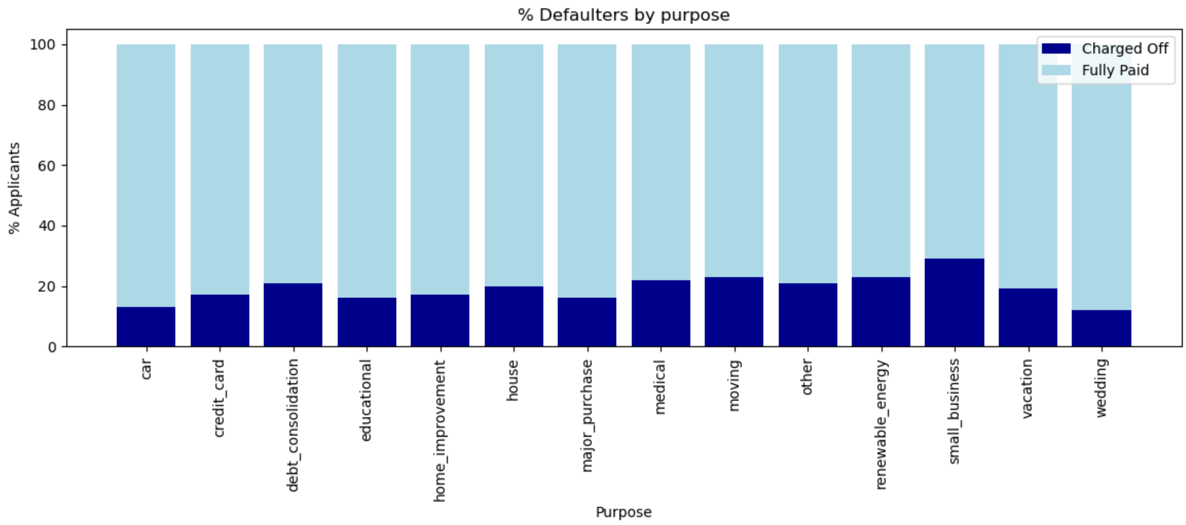
earliest_cr_line	open_acc	pub_rec	revol_bal	revol_util	total_acc	initial_list_status	application_type
Jun-1990	16.0	0.0	36369.0	41.8	25.0	w	INDIVIDUAL
Jul-2004	17.0	0.0	20131.0	53.3	27.0	f	INDIVIDUAL
Aug-2007	13.0	0.0	11987.0	92.2	26.0	f	INDIVIDUAL
Sep-2006	6.0	0.0	5472.0	21.5	13.0	f	INDIVIDUAL
Mar-1999	13.0	0.0	24584.0	69.8	43.0	f	INDIVIDUAL

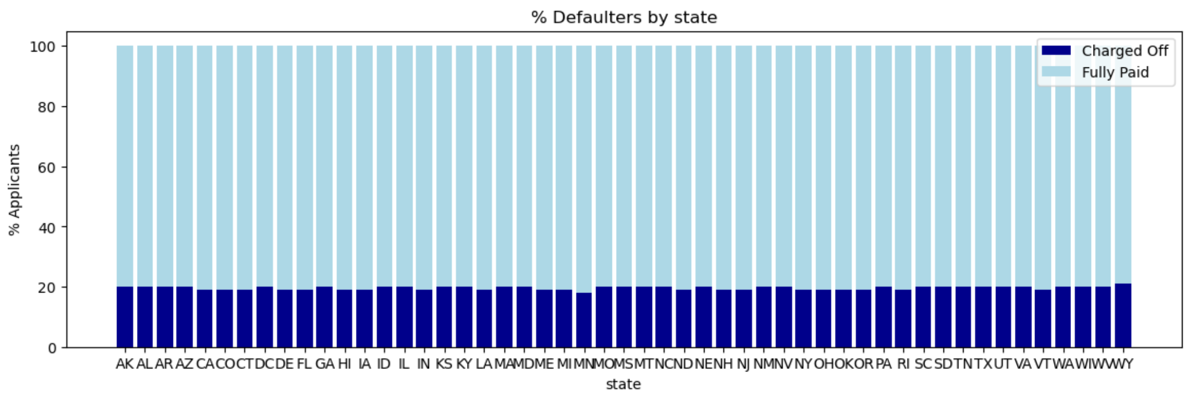
In [308]:
```
# Impact of categorical factors on loan status

plot = ['term', 'grade', 'sub_grade', 'home_ownership', 'verification_status',
       'zip_code', 'pub_rec', 'initial_list_status',
       'application_type', 'pub_rec_bankruptcies']

plt.figure(figsize=(14,20))
i=1
for col in plot:
  ax=plt.subplot(5,2,i)

  data = df.pivot_table(index=col, columns='loan_status', aggfunc='count', values
  data = data.div(data.sum(axis=1), axis=0).multiply(100).round()
  data.reset_index(inplace=True)

  plt.bar(data[col],data['Charged Off'], color='#00008b')
  plt.bar(data[col],data['Fully Paid'], color='#add8e6', bottom=data['Charged Off
  plt.xlabel(f'{col}')
  plt.ylabel('% Applicants')
  plt.title(f'% Defaulters by {col}')
  plt.legend(['Charged Off','Fully Paid'])
  i += 1

plt.tight_layout()
plt.show()
```



% Defaulters by term
% Applicants
36 months
60 months
term
Charged Off
Fully Paid
% Defaulters by grade
A B C D E F G
grade
% Defaulters by sub_grade
A1A2A3A4A5B1B2B3B4B5C1C2C3C4C5D1D2D3D4D5E1E2E3E4E5F1F2F3F4F5G1G2G3G4G5
sub_grade
% Defaulters by home_ownership
MORTGAGE NONE OTHER OWN RENT
home_ownership
% Defaulters by verification_status
Not Verified
Source Verified
Verified
verification_status
% Defaulters by zip_code
11650 22690 29597 30723 48052 70466 86630 93700
zip_code
% Defaulters by pub_rec
pub_rec
% Defaulters by initial_list_status
f
w
initial_list_status
% Defaulters by application_type
DIRECT_PAY
INDIVIDUAL
JOINT
application_type
% Defaulters by pub_rec_bankruptcies
pub_rec_bankruptcies

In [309]:
```
# Impact of Purpose/state on Loan status

purpose = df.pivot_table(index='purpose', columns='loan_status', aggfunc='count',
purpose = purpose.div(purpose.sum(axis=1), axis=0).multiply(100).round()
purpose.reset_index(inplace=True)

plt.figure(figsize=(14,4))
plt.bar(purpose['purpose'],purpose['Charged Off'], color='#00008b')
plt.bar(purpose['purpose'],purpose['Fully Paid'], color='#add8e6', bottom=purpose
plt.xlabel('Purpose')
plt.ylabel('% Applicants')
plt.title('% Defaulters by purpose')
plt.legend(['Charged Off','Fully Paid'])
plt.xticks(rotation=90)
plt.show()

state = df.pivot_table(index='state', columns='loan_status', aggfunc='count', val
state = state.div(state.sum(axis=1), axis=0).multiply(100).round()
state.reset_index(inplace=True)

plt.figure(figsize=(14,4))
plt.bar(state['state'],state['Charged Off'], color='#00008b')
plt.bar(state['state'],state['Fully Paid'], color='#add8e6', bottom=state['Charge
plt.xlabel('state')
plt.ylabel('% Applicants')
plt.title('% Defaulters by state')
plt.legend(['Charged Off','Fully Paid'])
plt.show()
```





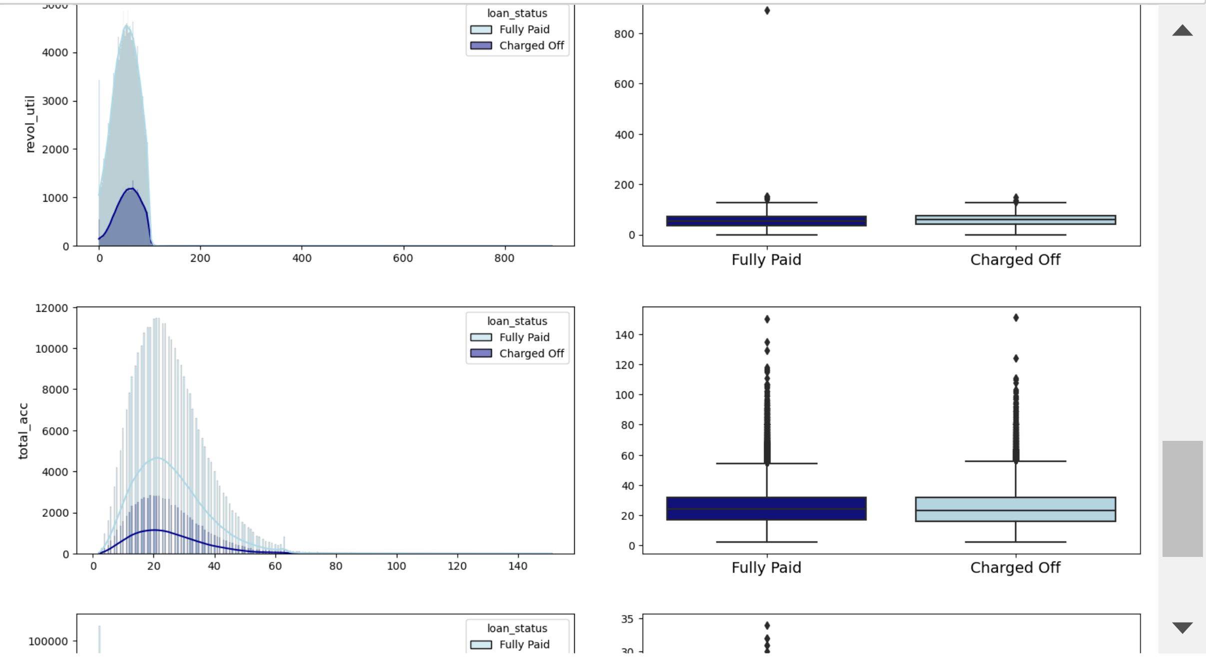
Observations:

- The % of defaulters is much higher for longer (60-month) term

- As expected, grade/sub-grade has the maximum impact on loan_status with highest grade having maximum defaulters
- Zip codes such as 11650, 86630 and 93700 have 100% defaulters
- We can remove initial_list_status and state as they have no impact on loan_status
- public records also don't seem to have any impact on loan_status surprisingly
- Direct pay application type has higher default rate compared to individual/joint
- Loan taken for the purpose of small business has the highest rate of default

In [322]:

```
# Impact of numerical features on loan_status

num_cols = ['loan_amnt', 'int_rate', 'emp_length', 'annual_inc', 'dti', 'open_acc
       'revol_bal', 'revol_util', 'total_acc', 'mort_acc']

fig, ax = plt.subplots(len(num_cols),2,figsize=(15,40))
i=0
color_dict = {'Fully Paid': matplotlib.colors.to_rgba('#add8e6', 0.5),
              'Charged Off': matplotlib.colors.to_rgba('#00008b', 1)}
for col in num_cols:
    sns.histplot(data=df, x=col, hue='loan_status', ax=ax[i, 0], legend=True,
                palette=color_dict, kde=True, fill=True)
    sns.boxplot(data=df, y=col, x='loan_status', ax=ax[i,1],
               palette=('#00008b', '#add8e6'))
    ax[i,0].set_ylabel(col, fontsize=12)
    ax[i,0].set_xlabel(' ')
    ax[i,1].set_xlabel(' ')
    ax[i,1].set_ylabel(' ')
    ax[i,1].xaxis.set_tick_params(labelsize=14)
    i += 1

plt.tight_layout()
plt.show()
```



Observations:

- From the boxplots, it can be observed that the mean loan_amnt, int_rate, dti, open_acc and revol_util are slightly higher for defaulters while annual income is lower

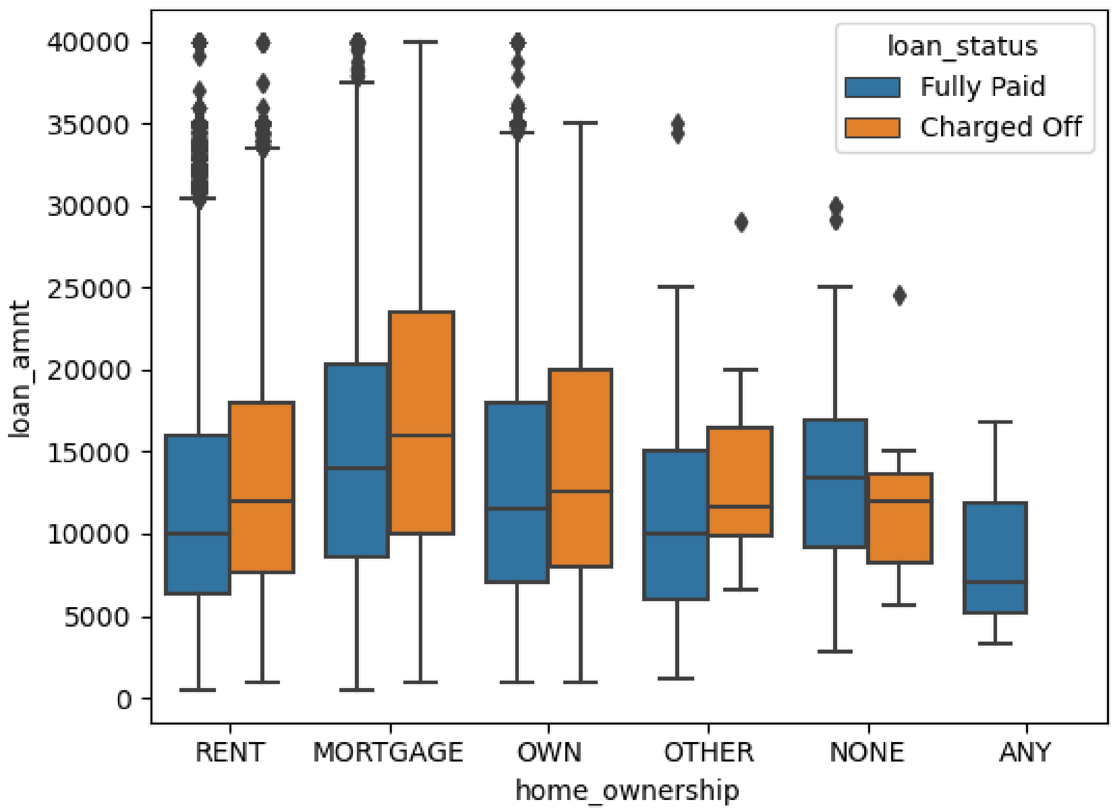
In [28]:
```
df.head()
```

me_ownership	annual_inc	...	open_acc	pub_rec	revol_bal	revol_util	total_acc	initial_list_status	application_typ
RENT	117000.0	...	16.0	0.0	36369.0	41.8	25.0	w	INDIVIDUA
MORTGAGE	65000.0	...	17.0	0.0	20131.0	53.3	27.0	f	INDIVIDUA
RENT	43057.0	...	13.0	0.0	11987.0	92.2	26.0	f	INDIVIDUA
RENT	54000.0	...	6.0	0.0	5472.0	21.5	13.0	f	INDIVIDUA
MORTGAGE	55000.0	...	13.0	0.0	24584.0	69.8	43.0	f	INDIVIDUA

Multivariate Analysis

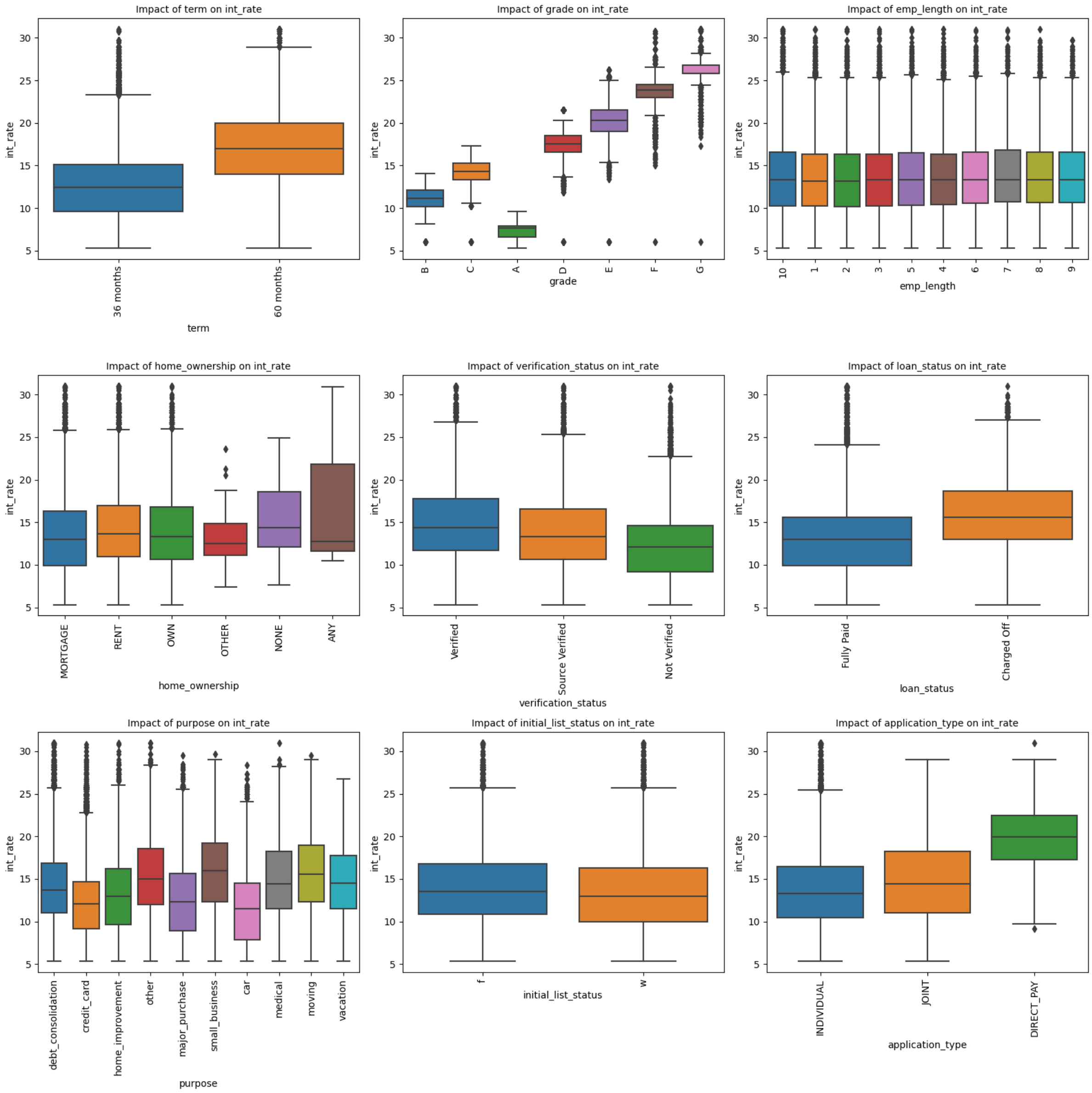
In [325]:
```
##Cat-Cat-Num
sns.boxplot(data = df, x="home_ownership", y='loan_amnt',hue="loan_status")
```

Out[325]: <Axes: xlabel='home_ownership', ylabel='loan_amnt'>

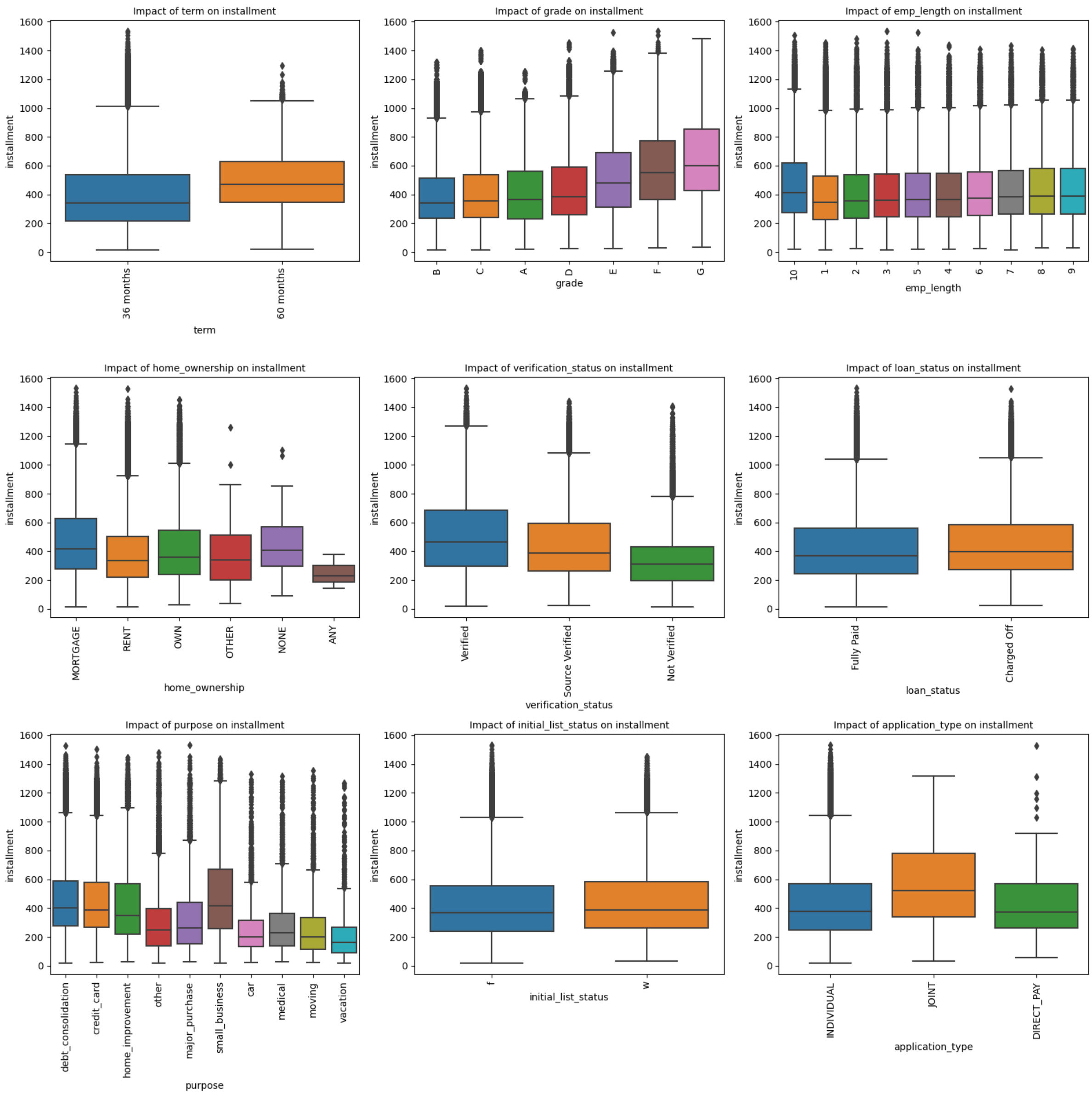


- Major part of loan amount for Mortageg is charged off
- Loan amot for Rent has major outliers

In [326]:
```
cat_cols=['term',
 'grade',
 'emp_length',
 'home_ownership',
 'verification_status',
 'loan_status',
 'purpose',
 'initial_list_status',
 'application_type']


#cat_cols = df.select_dtypes(include=['bool','category','object']).columns.tolist
plt.figure(figsize=(16,16))
i=1
for col in cat_cols:
  ax = plt.subplot(3,3,i)
  sns.boxplot(data = df, x=col, y='int_rate',order=df[col].value_counts().iloc[:1
  plt.title(f"Impact of {col} on int_rate", fontsize=10)
  plt.xlabel(col)
  plt.xticks(rotation='vertical')
  plt.ylabel('int_rate')
  i+=1

plt.tight_layout()
plt.show();
```


Impact of term on int_rate
int_rate
36 months
60 months
term
Impact of grade on int_rate
B
C
A
D
E
F
G
grade
Impact of emp_length on int_rate
10
1
2
3
5
4
6
7
8
9
emp_length
Impact of home_ownership on int_rate
MORTGAGE
RENT
OWN
OTHER
NONE
ANY
home_ownership
Impact of verification_status on int_rate
Verified
Source Verified
Not Verified
verification_status
Impact of loan_status on int_rate
Fully Paid
Charged Off
loan_status
Impact of purpose on int_rate
debt_consolidation
credit_card
home_improvement
other
major_purchase
small_business
car
medical
moving
vacation
purpose
Impact of initial_list_status on int_rate
f
w
initial_list_status
Impact of application_type on int_rate
INDIVIDUAL
JOINT
DIRECT_PAY
application_type

In [327]:
```
cat_cols=['term',
 'grade',
 'emp_length',
 'home_ownership',
 'verification_status',
 'loan_status',
 'purpose',
 'initial_list_status',
 'application_type']


#cat_cols = df.select_dtypes(include=['bool','category','object']).columns.tolist
plt.figure(figsize=(16,16))
i=1
for col in cat_cols:
  ax = plt.subplot(3,3,i)
  sns.boxplot(data = df, x=col, y='installment',order=df[col].value_counts().iloc
  plt.title(f"Impact of {col} on installment", fontsize=10)
  plt.xlabel(col)
  plt.xticks(rotation='vertical')
  plt.ylabel('installment')
  i+=1

plt.tight_layout()
plt.show();
```



2. Data Preprocessing

2.a Duplicate value check

In [331]:
```
# Check for Duplicate rows
df[df.duplicated()].shape[0]
```

Out[331]: 0

In [56]:
```
df[df.duplicated(subset=["emp_title","address1","loan_amnt"],keep=False)]
```

Out[56]:

cr_line	open_acc	pub_rec	revol_bal	revol_util	total_acc	initial_list_status	application_type	mort_ac
p-1999	4.0	1.0	2496.0	39.6	31.0	w	INDIVIDUAL	4.
n-1979	13.0	0.0	3434.0	9.8	37.0	w	INDIVIDUAL	2.
r-1970	9.0	0.0	30851.0	90.5	30.0	f	INDIVIDUAL	6.
b-1973	7.0	0.0	7870.0	37.8	17.0	f	INDIVIDUAL	2.
r-1977	9.0	1.0	3870.0	52.0	39.0	w	INDIVIDUAL	7.
n-2003	9.0	0.0	10573.0	42.6	25.0	w	INDIVIDUAL	2.
g-2002	5.0	0.0	14085.0	92.1	13.0	f	INDIVIDUAL	1.
n-1979	9.0	0.0	86016.0	81.2	17.0	f	INDIVIDUAL	Na
il-1986	10.0	0.0	33188.0	83.4	26.0	f	INDIVIDUAL	2.
b-1990	5.0	0.0	1219.0	7.9	18.0	f	INDIVIDUAL	1.
n-1976	13.0	0.0	12030.0	36.9	14.0	f	INDIVIDUAL	0.
c-1999	15.0	0.0	18257.0	40.2	21.0	f	INDIVIDUAL	1.
y-1986	16.0	0.0	14592.0	10.5	33.0	w	INDIVIDUAL	0.
n-2002	4.0	0.0	10218.0	41.2	7.0	w	INDIVIDUAL	1.
ir-1989	10.0	0.0	10704.0	67.3	42.0	f	INDIVIDUAL	1.
ir-2002	14.0	0.0	3862.0	13.5	32.0	f	INDIVIDUAL	Na
b-1992	27.0	0.0	17036.0	18.6	48.0	w	INDIVIDUAL	2.
p-2007	5.0	0.0	6505.0	83.4	7.0	f	INDIVIDUAL	Na
p-1995	7.0	0.0	19801.0	79.2	21.0	w	INDIVIDUAL	6.

:r_line	open_acc	pub_rec	revol_bal	revol_util	total_acc	initial_list_status	application_type	mort_ac
c-2005	3.0	0.0	3659.0	66.5	5.0	w	INDIVIDUAL	0.

- There are no duplicate rows

2.b Missing value treatment

In [332]:
```
df.isna().sum().sort_values(ascending=False)
```

Out[332]:
```
zip_code                42384
state                   42384
mort_acc                37795
emp_title               22927
emp_length              18301
title                    1755
pub_rec_bankruptcies      535
revol_util                276
open_acc                    0
city                        0
address1                    0
application_type            0
initial_list_status         0
total_acc                   0
revol_bal                   0
pub_rec                     0
loan_amnt                   0
earliest_cr_line            0
term                        0
purpose                     0
loan_status                 0
issue_d                     0
verification_status         0
annual_inc                  0
home_ownership              0
sub_grade                   0
grade                       0
installment                 0
int_rate                    0
dti                         0
dtype: int64
```

- There are few columns which have null values like mort_acct,emp_title,emp_length

In [333]:
```
df.isnull().sum().sort_values(ascending=False)
```

Out[333]:
```
zip_code                42384
state                   42384
mort_acc                37795
emp_title               22927
emp_length              18301
title                    1755
pub_rec_bankruptcies      535
revol_util                276
open_acc                    0
city                        0
address1                    0
application_type            0
initial_list_status         0
total_acc                   0
revol_bal                   0
pub_rec                     0
loan_amnt                   0
earliest_cr_line            0
term                        0
purpose                     0
loan_status                 0
issue_d                     0
verification_status         0
annual_inc                  0
home_ownership              0
sub_grade                   0
grade                       0
installment                 0
int_rate                    0
dti                         0
dtype: int64
```

In [334]:
```
df["mort_acc"].fillna(0,inplace=True)
df["pub_rec_bankruptcies"].fillna(0,inplace=True)
df["revol_util"].fillna(0,inplace=True)
```

In [335]:
```
df["emp_title"].fillna(value="Not Provided",inplace=True,axis=0)
df["emp_length"].fillna(value="Not Provided",inplace=True,axis=0)
df["title"].fillna(value="Not Provided",inplace=True,axis=0)
df["state"].fillna(value="Not Provided",inplace=True,axis=0)
df["zip_code"].fillna(value="Not Provided",inplace=True,axis=0)
```

In [336]:
```
df.isna().sum().sort_values(ascending=False)
```

Out[336]:
```
loan_amnt               0
term                    0
city                    0
state                   0
address1                0
pub_rec_bankruptcies    0
mort_acc                0
application_type        0
initial_list_status     0
total_acc               0
revol_util              0
revol_bal               0
pub_rec                 0
open_acc                0
earliest_cr_line        0
dti                     0
title                   0
purpose                 0
loan_status             0
issue_d                 0
verification_status     0
annual_inc              0
home_ownership          0
emp_length              0
emp_title               0
sub_grade               0
grade                   0
installment             0
int_rate                0
zip_code                0
dtype: int64
```

- All null record are handled

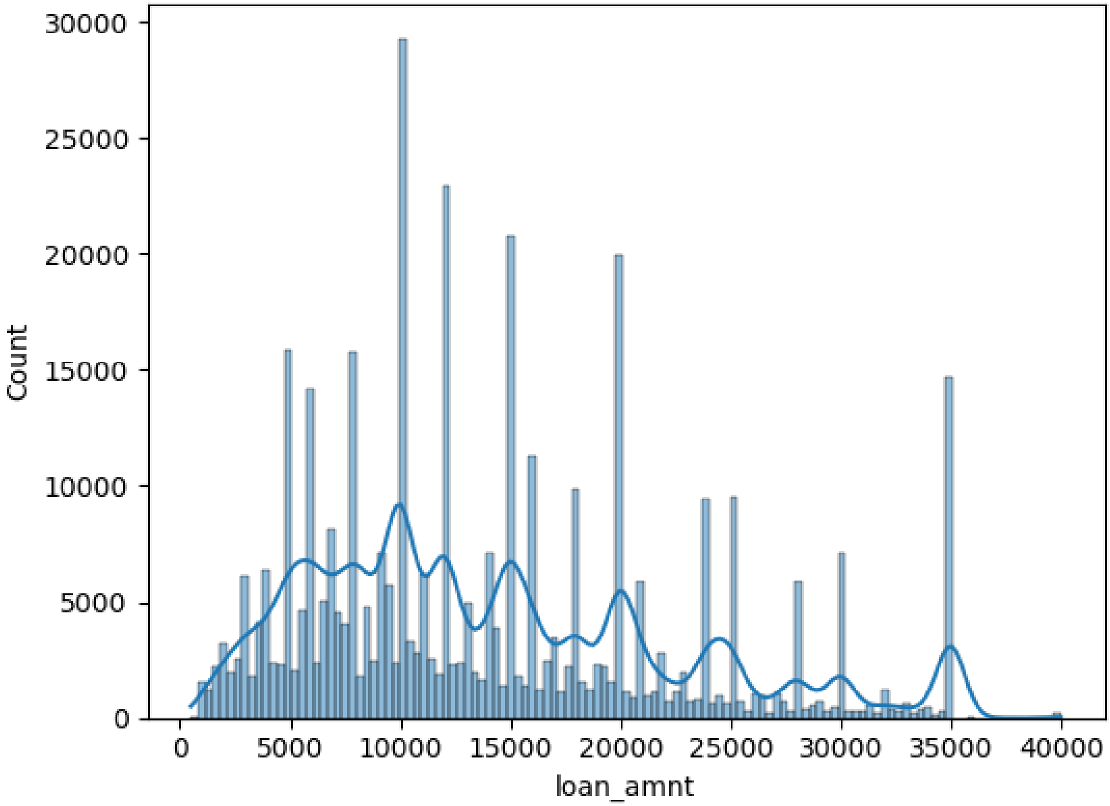
2.c Outlier treatment

In [337]:
```
numeric_cols
```

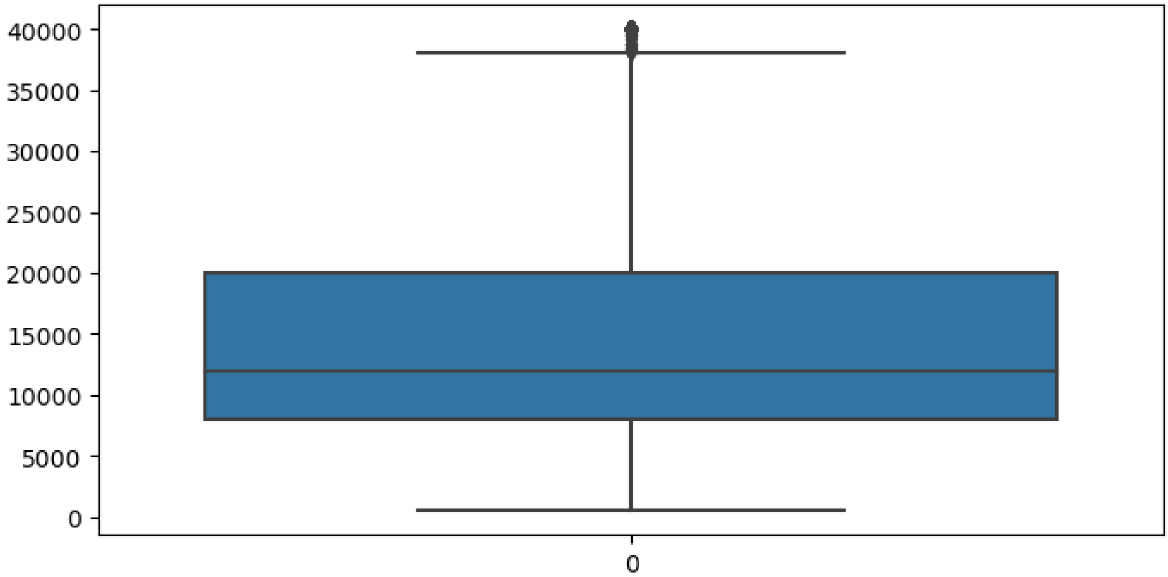
Out[337]:
```
['loan_amnt',
 'int_rate',
 'installment',
 'annual_inc',
 'dti',
 'open_acc',
 'pub_rec',
 'revol_bal',
 'revol_util',
 'total_acc',
 'mort_acc',
 'pub_rec_bankruptcies']
```

In [338]:
```
sns.histplot(df["loan_amnt"],kde=True)
```

Out[338]: <Axes: xlabel='loan_amnt', ylabel='Count'>



In [341]:
```
plt.figure(figsize=(8,4))
sns.boxplot(df["loan_amnt"])
plt.show()
```



In [342]:
```
q1_loan_amnt = df["loan_amnt"].quantile(0.25)
q3_loan_amnt = df["loan_amnt"].quantile(0.75)
iqr_loan_amt = q3_loan_amnt - q1_loan_amnt
lower_limit_loan_amnt = max((q1_loan_amnt - 1.5*iqr_loan_amt),0)
upper_limit_loan_amnt = q3_loan_amnt + 1.5*iqr_loan_amt
outliers_low = (df["loan_amnt"] < lower_limit_loan_amnt)
outliers_upper = (df["loan_amnt"] > upper_limit_loan_amnt)
total_oulliers_loan_amnt = len(df["loan_amnt"][outliers_upper]) + len(df["loan_am
```

In [343]:
```
for col in numeric_cols:
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3-q1
    lower_lim = max((q1-1.5*iqr),0)
    upper_lim = q3 + 1.5*iqr
    outliers_low = (df[col] < lower_lim)
    outliers_upper = (df[col] > upper_lim)
    total_outliers = len(df[col][outliers_upper]) + len(df[col][outliers_low])
    print(f" Total Outliers in {col} : {total_outliers}")
```

```
 Total Outliers in loan_amnt : 191
 Total Outliers in int_rate : 3777
 Total Outliers in installment : 11250
 Total Outliers in annual_inc : 16700
 Total Outliers in dti : 275
 Total Outliers in open_acc : 10307
 Total Outliers in pub_rec : 57758
 Total Outliers in revol_bal : 21259
 Total Outliers in revol_util : 12
 Total Outliers in total_acc : 8499
 Total Outliers in mort_acc : 6843
 Total Outliers in pub_rec_bankruptcies : 45115
```

In [345]:
```
#Removing outliers using standard deviation
for col in numeric_cols:
  mean=df[col].mean()
  std=df[col].std()
  upper = mean + (3*std)
  df = df[~(df[col]>upper)]
```

In [346]:
```
df.shape
```

Out[346]:
```
(361343, 30)
```

2.d Feature Engineering

In [ ]:
```
##Creation of Flags: For attributes like Pub_rec,Mort_acc, and Pub_rec_bankruptci
```

In [347]:
```
df["pub_rec"].value_counts()
```

Out[347]:
```
0.0    314335
1.0     47008
Name: pub_rec, dtype: int64
```

In [77]:
```
df["pub_rec"]=df["pub_rec"].astype("bool")
```

In [78]:
```
df["pub_rec"].value_counts()
```

Out[78]:
```
False    338272
True      57758
Name: pub_rec, dtype: int64
```

In [348]:
```
df["mort_acc"].value_counts()
```

Out[348]:
```
0.0    169400
1.0     56079
2.0     45507
3.0     34322
4.0     24950
5.0     16130
6.0      9696
7.0      5259
Name: mort_acc, dtype: int64
```

In [349]:
```
df["pub_rec_bankruptcies"].value_counts()
```

Out[349]:
```
0.0    323053
1.0     38290
Name: pub_rec_bankruptcies, dtype: int64
```

In [350]:
```
df["pub_rec_bankruptcies"]=df["pub_rec_bankruptcies"].astype("int")
df["pub_rec"]=df["pub_rec"].astype("int")
```

In [351]:
```
df.head()
```

Out[351]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_owne
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10	
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4	MORT(
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	1	
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6	
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9	MORT(

In [352]:
```
df["issue_year"] = pd.to_datetime(df["issue_d"]).dt.year
df["issue_month"] = pd.to_datetime(df["issue_d"]).dt.month
```

In [353]:
```
df["earliest_cr_line_year"] = pd.to_datetime(df["earliest_cr_line"]).dt.year
df["earliest_cr_line_month"] = pd.to_datetime(df["earliest_cr_line"]).dt.month
```

In [354]:
```
df.head()
```

Out[354]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_owne
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10	
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4	MORT(
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	1	
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6	
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9	MORT(

2.e Multicollinearity and Feature Selection

In [357]:
```
df.head()
```

Out[357]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_owne
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10	
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4	MORT(
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	1	
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6	
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9	MORT(

In [358]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 361343 entries, 0 to 396029
Data columns (total 34 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   loan_amnt                 361343 non-null  float64
 1   term                      361343 non-null  object 
 2   int_rate                  361343 non-null  float64
 3   installment               361343 non-null  float64
 4   grade                     361343 non-null  object 
 5   sub_grade                 361343 non-null  object 
 6   emp_title                 361343 non-null  object 
 7   emp_length                361343 non-null  object 
 8   home_ownership            361343 non-null  object 
 9   annual_inc                361343 non-null  float64
 10  verification_status       361343 non-null  object 
 11  issue_d                   361343 non-null  object 
 12  loan_status               361343 non-null  object 
 13  purpose                   361343 non-null  object 
 14  title                     361343 non-null  object 
 15  dti                       361343 non-null  float64
 16  earliest_cr_line          361343 non-null  object 
 17  open_acc                  361343 non-null  float64
 18  pub_rec                   361343 non-null  int32  
 19  revol_bal                 361343 non-null  float64
 20  revol_util                361343 non-null  float64
 21  total_acc                 361343 non-null  float64
 22  initial_list_status       361343 non-null  object 
 23  application_type          361343 non-null  object 
 24  mort_acc                  361343 non-null  float64
 25  pub_rec_bankruptcies      361343 non-null  int32  
 26  address1                  361343 non-null  object 
 27  state                     361343 non-null  object 
 28  city                      361343 non-null  object 
 29  zip_code                  361343 non-null  object 
 30  issue_year                361343 non-null  int64  
 31  issue_month               361343 non-null  int64  
 32  earliest_cr_line_year     361343 non-null  int64  
 33  earliest_cr_line_month    361343 non-null  int64  
dtypes: float64(10), int32(2), int64(4), object(18)
memory usage: 93.7+ MB
```

In [361]:
```
df["loan_status"].value_counts()
```

Out[361]:
```
Fully Paid     290359
Charged Off     70984
Name: loan_status, dtype: int64
```

In [373]:
```
df.loc[df["loan_status"] =="Fully Paid","loan_status"]=1
df.loc[df["loan_status"] =="Charged Off","loan_status"]=0
df["loan_status"] = df["loan_status"].astype(int)
```

In [374]:
```
df["emp_length"].value_counts()
```

Out[374]:
```
10              112403
1                53403
2                33004
3                29197
5                24447
4                22099
6                19233
7                19173
8                17577
Not Provided     16758
9                14049
Name: emp_length, dtype: int64
```

In [375]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 361343 entries, 0 to 396029
Data columns (total 34 columns):
 #   Column                   Non-Null Count   Dtype  
---  ------                   --------------   -----  
 0   loan_amnt                361343 non-null  float64
 1   term                     361343 non-null  object 
 2   int_rate                 361343 non-null  float64
 3   installment              361343 non-null  float64
 4   grade                    361343 non-null  object 
 5   sub_grade                361343 non-null  object 
 6   emp_title                361343 non-null  object 
 7   emp_length               361343 non-null  object 
 8   home_ownership           361343 non-null  object 
 9   annual_inc               361343 non-null  float64
 10  verification_status      361343 non-null  object 
 11  issue_d                  361343 non-null  object 
 12  loan_status              361343 non-null  int32  
 13  purpose                  361343 non-null  object 
 14  title                    361343 non-null  object 
 15  dti                      361343 non-null  float64
 16  earliest_cr_line         361343 non-null  object 
 17  open_acc                 361343 non-null  float64
 18  pub_rec                  361343 non-null  int32  
 19  revol_bal                361343 non-null  float64
 20  revol_util               361343 non-null  float64
 21  total_acc                361343 non-null  float64
 22  initial_list_status      361343 non-null  object 
 23  application_type         361343 non-null  object 
 24  mort_acc                 361343 non-null  float64
 25  pub_rec_bankruptcies     361343 non-null  int32  
 26  address1                 361343 non-null  object 
 27  state                    361343 non-null  object 
 28  city                     361343 non-null  object 
 29  zip_code                 361343 non-null  object 
 30  issue_year               361343 non-null  int64  
 31  issue_month              361343 non-null  int64  
 32  earliest_cr_line_year    361343 non-null  int64  
 33  earliest_cr_line_month   361343 non-null  int64  
dtypes: float64(10), int32(3), int64(4), object(17)
memory usage: 92.4+ MB
```

In [376]:
```
cat_cols = df.select_dtypes(include=['bool','category','object']).columns.tolist(
```

In [377]:
```
cat_cols
```

Out[377]:
```
['term',
 'grade',
 'sub_grade',
 'emp_title',
 'emp_length',
 'home_ownership',
 'verification_status',
 'issue_d',
 'purpose',
 'title',
 'earliest_cr_line',
 'initial_list_status',
 'application_type',
 'address1',
 'state',
 'city',
 'zip_code']
```

In [378]:
```
for col in cat_cols:
    print(f"{col} --> {df[col].value_counts()}",end="\n")
```

```
term -->  36 months    276243
 60 months     85100
Name: term, dtype: int64
grade --> B    107943
C     96047
A     59396
D     57192
E     28176
F     10499
G      2090
Name: grade, dtype: int64
sub_grade --> B3    24863
B4    23849
C1    21760
B2    20933
C2    20616
B5    20478
C3    19043
C4    18260
```

- Most of the categorical variables have more than 2 categories so we will use target encoding here

In [379]:
```
cat_cols
```

Out[379]:
```
['term',
 'grade',
 'sub_grade',
 'emp_title',
 'emp_length',
 'home_ownership',
 'verification_status',
 'issue_d',
 'purpose',
 'title',
 'earliest_cr_line',
 'initial_list_status',
 'application_type',
 'address1',
 'state',
 'city',
 'zip_code']
```

In [380]:
```
# Separate predictor and target variables
x = df.drop(["loan_status","pub_rec","pub_rec_bankruptcies"], axis=1)
y = df[['loan_status']]
```

In [381]:
```
x.head()
```

Out[381]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_owne
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10	
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4	MORT(
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	1	
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6	
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9	MORT(

In [382]:
```
y.head()
```

Out[382]:

	loan_status
0	1
1	1
2	1
3	1
4	0

In [383]:
```
# Split the data into training and test data

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
                                                    random_state=42)

print(f'Shape of x_train: {x_train.shape}')
print(f'Shape of x_test: {x_test.shape}')
print(f'Shape of y_train: {y_train.shape}')
print(f'Shape of y_test: {y_test.shape}')
```

```
Shape of x_train: (289074, 31)
Shape of x_test: (72269, 31)
Shape of y_train: (289074, 1)
Shape of y_test: (72269, 1)
```

In [384]:
```
import category_encoders as ce
te = ce.TargetEncoder()
x_train = te.fit_transform(x_train, y_train)
```

In [385]:
```
x_test = te.transform(x_test)
```

In [386]:
```
#Initialising object of class StandardScaler() for Standardisation
scaler_x = StandardScaler()
#Transforming numeric columns of x_train and x_test
all_cols = x_train.columns
x_train[all_cols]=scaler_x.fit_transform(x_train[all_cols])
x_test[all_cols]=scaler_x.transform(x_test[all_cols])
```

In [387]:
```
x_train.head()
```

Out[387]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	l
246119	-0.580162	0.555274	0.463710	-0.431614	-0.931135	-0.648720	-1.590412	0.619026	
389833	-1.087179	0.555274	-0.356169	-1.079232	0.675849	0.679405	0.210733	-0.339642	
217137	-0.301303	-1.800913	1.127314	-0.549164	-1.753970	-1.482284	0.569058	0.244334	
118015	2.537991	-1.800913	0.617721	1.779935	-0.166934	-0.388289	0.656135	0.619026	
33812	0.307117	-1.800913	0.463710	-0.121834	-0.166934	-0.208726	-0.825047	0.236994	

In [388]:
```
x_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 289074 entries, 246119 to 133644
Data columns (total 31 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   loan_amnt              289074 non-null  float64
 1   term                   289074 non-null  float64
 2   int_rate               289074 non-null  float64
 3   installment            289074 non-null  float64
 4   grade                  289074 non-null  float64
 5   sub_grade              289074 non-null  float64
 6   emp_title              289074 non-null  float64
 7   emp_length             289074 non-null  float64
 8   home_ownership         289074 non-null  float64
 9   annual_inc             289074 non-null  float64
 10  verification_status    289074 non-null  float64
 11  issue_d                289074 non-null  float64
 12  purpose                289074 non-null  float64
 13  title                  289074 non-null  float64
 14  dti                    289074 non-null  float64
 15  earliest_cr_line       289074 non-null  float64
 16  open_acc               289074 non-null  float64
 17  revol_bal              289074 non-null  float64
 18  revol_util             289074 non-null  float64
 19  total_acc              289074 non-null  float64
 20  initial_list_status    289074 non-null  float64
 21  application_type       289074 non-null  float64
 22  mort_acc               289074 non-null  float64
 23  address1               289074 non-null  float64
 24  state                  289074 non-null  float64
 25  city                   289074 non-null  float64
 26  zip_code               289074 non-null  float64
 27  issue_year             289074 non-null  float64
 28  issue_month            289074 non-null  float64
 29  earliest_cr_line_year  289074 non-null  float64
 30  earliest_cr_line_month 289074 non-null  float64
dtypes: float64(31)
memory usage: 70.6 MB
```

In [390]:
```
# Statmodels implementation
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# X = df[df.columns.drop('selling_price')]
# y = df["selling_price"]

# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

scaler = StandardScaler()
X_tr_scaled = scaler.fit_transform(x_train)

X_sm = sm.add_constant(X_tr_scaled)  #Statmodels default is without intercept, to

sm_model = sm.OLS(y_train, X_sm).fit()

print(sm_model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:            loan_status   R-squared:                       0.972
Model:                            OLS   Adj. R-squared:                  0.972
Method:                 Least Squares   F-statistic:                 3.192e+05
Date:                Fri, 12 Jan 2024   Prob (F-statistic):               0.00
Time:                        01:14:30   Log-Likelihood:             3.7155e+05
No. Observations:              289074   AIC:                        -7.430e+05
Df Residuals:                  289042   BIC:                        -7.427e+05
Df Model:                          31
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.8037      0.000   6456.639      0.000       0.803       0.804
x1             0.0006      0.001      0.586      0.558      -0.001       0.002
x2             0.0015      0.000      4.493      0.000       0.001       0.002
x3            -0.0010      0.001     -1.713      0.087      -0.002       0.000
x4            -0.0010      0.001     -1.096      0.273      -0.003       0.001
x5            -0.0002      0.001     -0.362      0.717      -0.001       0.001
x6             0.0009      0.001      1.205      0.228      -0.001       0.002
x7             0.0087      0.000     56.758      0.000       0.008       0.009
x8            -0.0010      0.000     -7.326      0.000      -0.001      -0.001
x9             0.0005      0.000      3.385      0.001       0.000       0.001
x10           -0.0003      0.000     -2.059      0.039      -0.001   -1.61e-05
x11         2.386e-05      0.000      0.177      0.860      -0.000       0.000
x12            0.0008      0.000      5.377      0.000       0.001       0.001
x13           -0.0008      0.000     -5.890      0.000      -0.001      -0.001
x14            0.0028      0.000     17.376      0.000       0.002       0.003
x15           -0.0007      0.000     -4.764      0.000      -0.001      -0.000
x16            0.0003      0.000      1.913      0.056   -7.38e-06       0.001
x17           -0.0007      0.000     -3.740      0.000      -0.001      -0.000
x18          -8.26e-06      0.000     -0.049      0.961      -0.000       0.000
x19           -0.0005      0.000     -3.152      0.002      -0.001      -0.000
x20            0.0004      0.000      2.420      0.016    8.55e-05       0.001
x21           -0.0004      0.000     -2.542      0.011      -0.001   -8.22e-05
x22            0.0002      0.000      1.305      0.192   -8.16e-05       0.000
x23            0.0003      0.000      1.866      0.062    -1.5e-05       0.001
x24            0.3752      0.000   1985.874      0.000       0.375       0.376
x25         7.388e-05      0.000      0.593      0.553      -0.000       0.000
x26            0.0186      0.000    140.524      0.000       0.018       0.019
x27            0.0064      0.000     39.817      0.000       0.006       0.007
x28            0.0018      0.000     10.492      0.000       0.001       0.002
x29         9.751e-05      0.000      0.761      0.447      -0.000       0.000
x30           -0.0001      0.000     -0.764      0.445      -0.000       0.000
x31        -5.776e-05      0.000     -0.463      0.643      -0.000       0.000
==============================================================================
Omnibus:                   131522.696   Durbin-Watson:                   2.001
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        123476046.010
Skew:                          -0.686   Prob(JB):                         0.00
Kurtosis:                     104.240   Cond. No.                         21.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [391]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame() # blank dataframe to store VIF Values
X_t = pd.DataFrame(X_tr_scaled, columns=x_train.columns) # add values and columns
vif['Features'] = X_t.columns
vif['VIF'] = [variance_inflation_factor(X_t.values, i) for i in range(X_t.shape[1
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = "VIF", ascending = False)
vif
```

Out[391]:

	Features	VIF
0	loan_amnt	58.60
3	installment	49.05
5	sub_grade	38.59
4	grade	21.45
2	int_rate	20.13
1	term	6.88
23	address1	2.30
19	total_acc	2.23
16	open_acc	2.12
27	issue_year	1.89
17	revol_bal	1.86
9	annual_inc	1.70
29	earliest_cr_line_year	1.68
26	zip_code	1.66
22	mort_acc	1.65
13	title	1.64
15	earliest_cr_line	1.60
18	revol_util	1.53
6	emp_title	1.50
11	issue_d	1.48
14	dti	1.47
8	home_ownership	1.37
12	purpose	1.29
20	initial_list_status	1.29
10	verification_status	1.18
25	city	1.13
7	emp_length	1.11
28	issue_month	1.06
21	application_type	1.00
24	state	1.00
30	earliest_cr_line_month	1.00

- loan_amnt, installment,grade,subgrade,int_rate,term have high multicollinearity

Model Building

In [406]:
```
from sklearn.linear_model import LogisticRegression

lgr = LogisticRegression(random_state=42,penalty=None,class_weight="balanced")

lgr.fit(x_train,y_train )
```

Out[406]:
```
LogisticRegression
LogisticRegression(class_weight='balanced', penalty=None, random_state=42)
```

In [408]:
```
y_pred=lgr.predict(x_test)
y_pred_prob = lgr.predict_proba(x_test)
```

In [409]:
```
lgr.score(x_test, y_test) # accuracy
```

Out[409]:
```
0.8830618937580429
```

In [411]:
```
lgr.score(x_train,y_train) # accuracy
```

Out[411]:
```
0.9968485578087272
```

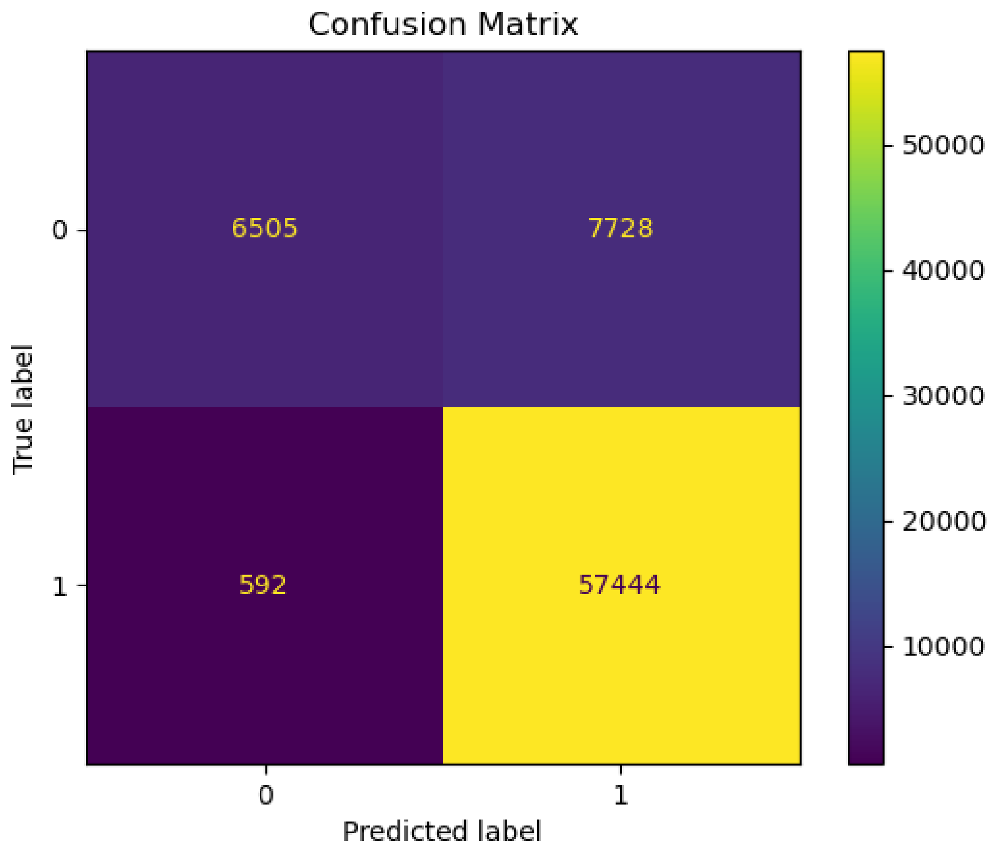
- Model is overfitted

In [413]:
```
# Oversampling to balance the target variable

sm=SMOTE(random_state=42)
x_train_res, y_train_res = sm.fit_resample(x_train,y_train)

print(f"Before OverSampling, count of label 1: {np.sum(y_train == 1)}")
print(f"Before OverSampling, count of label 0: {np.sum(y_train == 0)}")
print(f"After OverSampling, count of label 1: {np.sum(y_train_res == 1)}")
print(f"After OverSampling, count of label 0: {np.sum(y_train_res == 0)}")
```

```
Before OverSampling, count of label 1: loan_status    232323
dtype: int64
Before OverSampling, count of label 0: loan_status    56751
dtype: int64
After OverSampling, count of label 1: loan_status    232323
dtype: int64
After OverSampling, count of label 0: loan_status    232323
dtype: int64
```

In [414]:
```
model = LogisticRegression()
model.fit(x_train_res, y_train_res)
train_preds = model.predict(x_train)
test_preds = model.predict(x_test)

#Model Evaluation
print('Train Accuracy :', model.score(x_train, y_train).round(2))
print('Train F1 Score:',f1_score(np.ravel(y_train),train_preds).round(2))
print('Train Recall Score:',recall_score(y_train,train_preds).round(2))
print('Train Precision Score:',precision_score(y_train,train_preds).round(2))

print('\nTest Accuracy :',model.score(x_test,y_test).round(2))
print('Test F1 Score:',f1_score(np.ravel(y_test),test_preds).round(2))
print('Test Recall Score:',recall_score(y_test,test_preds).round(2))
print('Test Precision Score:',precision_score(y_test,test_preds).round(2))

# Confusion Matrix
cm = confusion_matrix(y_test, test_preds)
disp = ConfusionMatrixDisplay(cm)
disp.plot()
plt.title('Confusion Matrix')
plt.show()
```

```
Train Accuracy : 1.0
Train F1 Score: 1.0
Train Recall Score: 1.0
Train Precision Score: 1.0

Test Accuracy : 0.88
Test F1 Score: 0.93
Test Recall Score: 0.99
Test Precision Score: 0.88
```

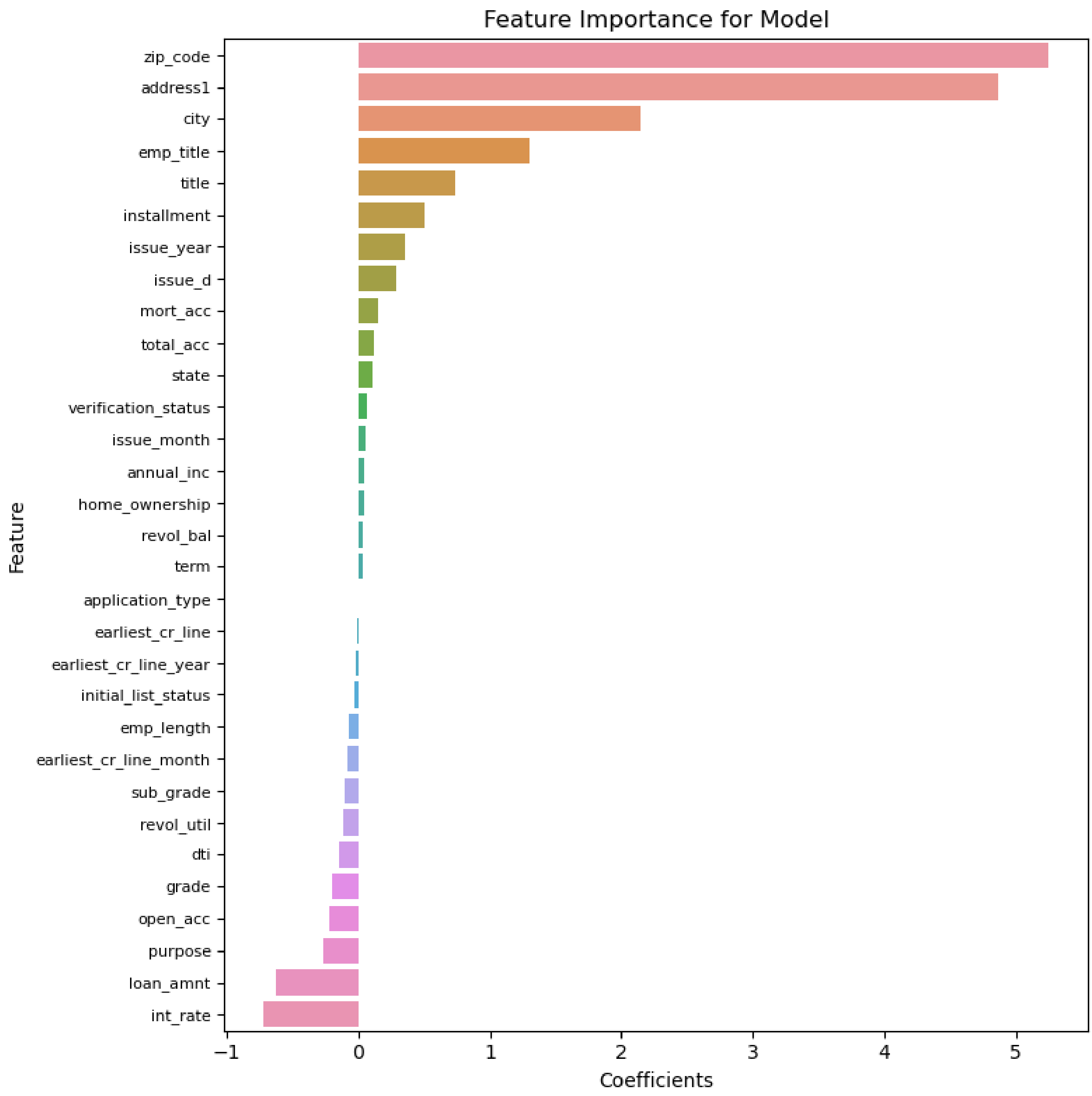


In [415]:
```
print(classification_report(y_test, test_preds))
```

```
              precision    recall  f1-score   support

           0       0.92      0.46      0.61     14233
           1       0.88      0.99      0.93     58036

    accuracy                           0.88     72269
   macro avg       0.90      0.72      0.77     72269
weighted avg       0.89      0.88      0.87     72269
```

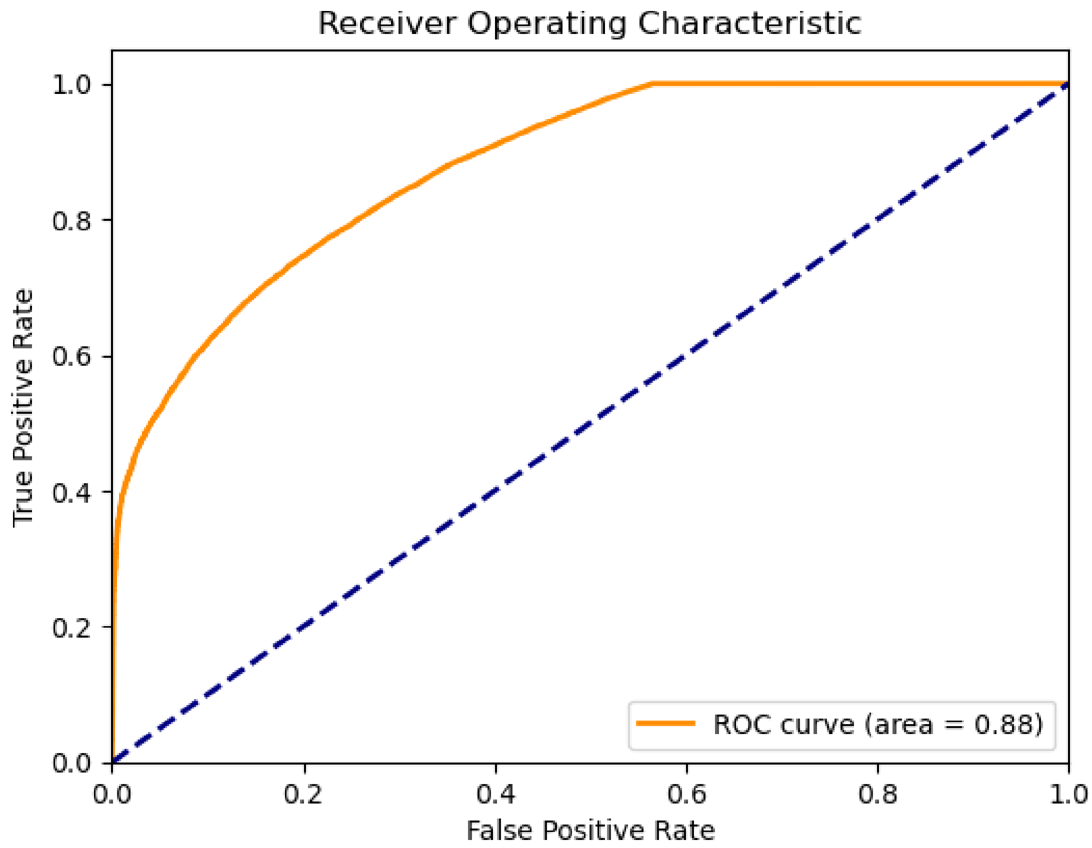
- It can be observed that the precision score is very high (our model is able to identify 80% of applicant with high credit wothiness) but the recall is low for defaulters (of all the predicted defaulters, only 46% are actually defaulters).
- Although this model is effective in extending credits to potential applicant but may not help in reducing NPAs by flagging most of the defaulters, it may cause loantap to give loans to defaulters (false positives)
- Low recall has also caused F1 score to drop to 61% even though accuracy is 88%

In [417]:
```
feature_imp = pd.DataFrame({'Columns':x_train.columns, 'Coefficients':model.coef_

plt.figure(figsize=(8,8))
sns.barplot(y = feature_imp['Columns'],
            x =  feature_imp['Coefficients'])
plt.title("Feature Importance for Model")
plt.yticks(fontsize=8)
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```



- The model has assigned large weightage to zip_code features

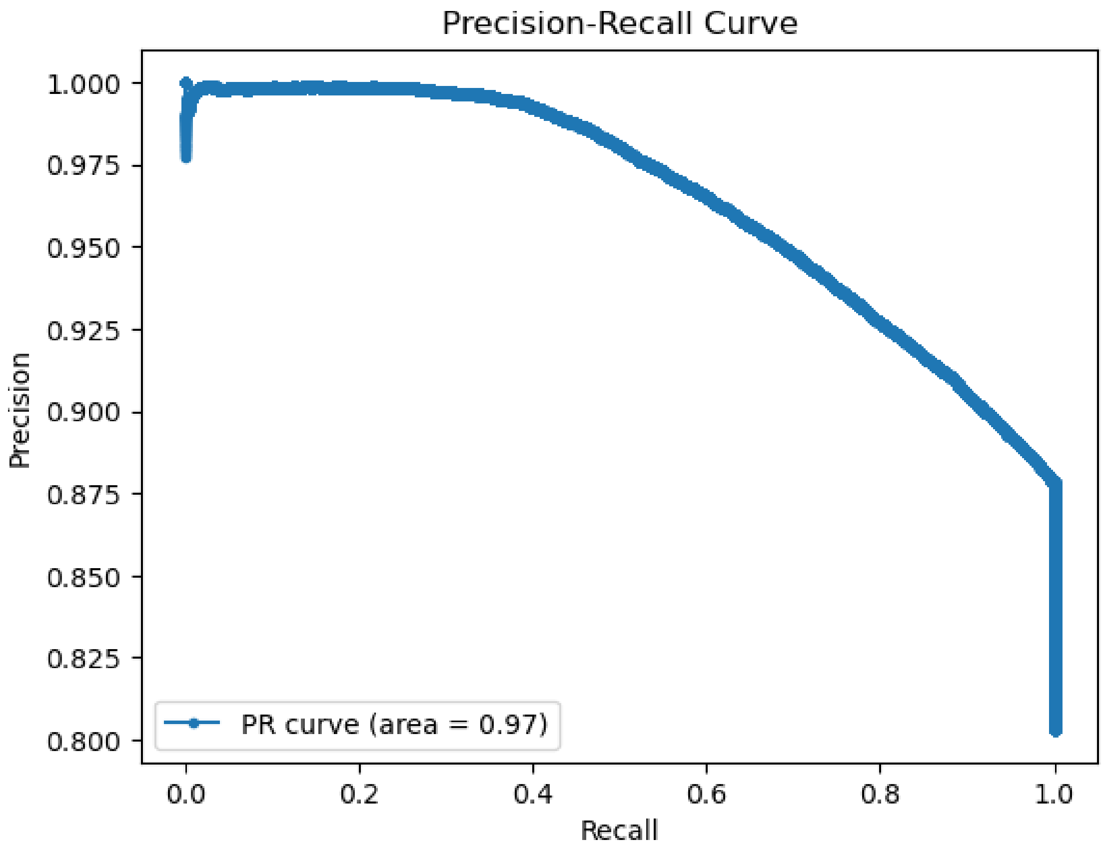
In [419]:
```
# Predict probabilities for the test set
probs = model.predict_proba(x_test)[:,1]

# Compute the false positive rate, true positive rate, and thresholds
fpr, tpr, thresholds = roc_curve(y_test, probs)

# Compute the area under the ROC curve
roc_auc = auc(fpr, tpr)

# Plot the ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % r
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()
```



- AUC of 0.88 signifies that the model is able to discriminate well between the positive and the negative class.
- But it is not a good measure for an imbalanced target variable because it may be high even when the classifier has a poor score on the minority class.
- This can happen when the classifier performs well on the majority class instances, which dominate the dataset. As a result, the AUC may appear high, but the model may not effectively identify the minority class instances.

Lets plot the Precision-Recall curve which is more suited for evaluation of imbalanced data

In [420]:
```
# Compute the false precision and recall at all thresholds
precision, recall, thresholds = precision_recall_curve(y_test, probs)

# Area under Precision Recall Curve
auprc = average_precision_score(y_test, probs)

# Plot the precision-recall curve
plt.plot(recall, precision, marker='.', label='PR curve (area = %0.2f)' % auprc)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc="lower left")
plt.show()
```



AUPRC score is high (close to one) shows better performance

Tradeoff Questions:

- How can we make sure that our model can detect real defaulters and there are less false positives? This is important as we can lose out on an opportunity to finance more individuals and earn interest on it

If priority is to minimise false positives, we can choose a higher threshold, which will result in fewer false positives at the cost of potentially missing some true positives. Precision value should be focussed to be increased

- Since NPA (non-performing asset) is a real problem in this industry, it's important we play safe and shouldn't disburse loans to anyone. We should aim to find a balance between precision ad recall

To balance precision and recall, a number of techniques can be used, such as adjusting the decision threshold or using an ensemble of models. Another approach is to use a metric that

Recommendations

- The optimal strategy to achieve the objective of balancing the risk of increasing NPAs by disbursing loans to defaulters with the opportunity to earn interest by disbursing loans to as many worthy customers as possible: maximise the F1 score along with the area under Precision Recall Curve (precision-recall trade-off) More complex classifiers like random forest would give better results compared to logistic regression because they are not restricted by the linearity of decision boundary

Insights Model is performimg better than a average model with high AUPRC.

- Loan Amt and Installment are highly correlated
- Most of applicant are Teachers or Manager.These people have a stable income and hence very less counts as NPA with these emp titles
- Majority amount of loans are for debt consolidation

In [ ]:

```

```