Lending Club Loan Data

Dataset Link -> https://www.kaggle.com/datasets/wordsforthewise/lending-club

University - STEVENS Institute of Technology

Professor - Hadi Safari Katesari

Presenter - Blessy Vincent Theogaraj

Introduction

Interested in exploring the ability to predict a borrower's likelihood to pay back a loan based on available information at the origination of a loan.

This dataset provides us with the loan information from Lending Club, one of the world's largest peer-to-peer lending platforms (https://www.kaggle.com/wordsforthewise/lending-club).The dataset includes important information about a large amount of specific loans. We will use the features, some describing the loan(e.g., the monthly payment and interest rate) while others are about borrowers(e.g., borrowers' job title and annual income), to **predict whether this loan will be finally paid off**. It is a **binary-classification** task.

Goal

- We will use exploratory data analysis, including statistical analysis and visualization analysis, to extract features that significantly influence the loan status: whether this loan would be paid off. It is a large dataset with millions of loan data and more than 20 features. What's more, not all features are numerical. Some missing values exists. We will show how we deal with this challenging dataset and make data-driven conclusions.
- We will implement several models for our classification, visualize and explain each followed by comparing their methods and results. Our plan is to include:

- PCA and LDA
- Logistic regression and
- Statistical Analysis

- The amount of loans being paid off is far more than the number of loans that were charged off, so we meet data imbalance problem here. We will try to solve this problem from both the insight of data processing and machine learning model development.

- Data processing insight: Develop sampling method. We would try both oversampling and down-sampling.
- Model insight: Adjust the weight of data from different categories in the objective function for model to optimize.

Approach

1. Dataset input
2. Read the csv file - .csv file is provided by kaggle.com
3. Remove entries not known at loan initiation
4. Handle NaN's
    - Columns with a majority of NaN's are removed
    - Entries with NaN are replaced with mean/mode
5. Remove in-process loans and reduce enumerations to good/bad
6. Separate the data into training and testing sets (X_train/y_train, X_test/y_test)
7. Begin testing models - As this is a 2 class problem (defaults or does not default on loan) - After working through the data, we tried a few different techniques than originally proposed:
8. LDA
9. Logistic Regression
10. PCA+Logistic Regression

Compare results and analyze performance vs expectations/understanding

In [1]:

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

```python
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
from sklearn.metrics import roc_auc_score
# ignore the warning if any
import warnings
warnings.filterwarnings("ignore")

# set row/columns
pd.options.display.max_columns= None
pd.options.display.max_rows= None
np.set_printoptions(suppress=True)
```

Read the Dataset

In [2]:

```python
# read the dataset
df = pd.read_csv("accepted_2007_to_2018Q4.csv")
```

In [3]:

```python
df.head()
```

Out[3]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_length
0	68407277	NaN	3600.0	3600.0	3600.0	36 months	13.99	123.03	C	C4	leadman	10+ years
1	68355089	NaN	24700.0	24700.0	24700.0	36 months	11.99	820.28	C	C1	Engineer	10+ years
2	68341763	NaN	20000.0	20000.0	20000.0	60 months	10.78	432.66	B	B4	truck driver	10+ years
3	66310712	NaN	35000.0	35000.0	35000.0	60 months	14.85	829.90	C	C5	Information Systems Officer	10+ years
4	68476807	NaN	10400.0	10400.0	10400.0	60 months	22.45	289.91	F	F1	Contract Specialist	3 years

In [4]:

```python
df.shape
```

Out[4]:

```
(2260701, 151)
```

In [5]:

```python
len(df.columns)
```

Out[5]:

```
151
```

Understand the data and clean for any NAN values

In [6]:

```python
isna_df = df.isna().sum()
```

In [7]:

```python
isna_df
```

Out[7]:

```
id                                                0
member_id                                   2260701
loan_amnt                                        33
funded_amnt                                      33
funded_amnt_inv                                  33
term                                             33
int_rate                                         33
installment                                      33
grade                                            33
sub_grade                                        33
emp_title                                    167002
emp_length                                   146940
home_ownership                                   33
annual_inc                                       37
verification_status                              33
issue_d                                          33
loan_status                                      33
pymnt_plan                                       33
```

url	33
desc	2134634
purpose	33
title	23358
zip_code	34
addr_state	33
dti	1744
delinq_2yrs	62
earliest_cr_line	62
fico_range_low	33
fico_range_high	33
inq_last_6mths	63
mths_since_last_delinq	1158535
mths_since_last_record	1901545
open_acc	62
pub_rec	62
revol_bal	33
revol_util	1835
total_acc	62
initial_list_status	33
out_prncp	33
out_prncp_inv	33
total_pymnt	33
total_pymnt_inv	33
total_rec_prncp	33
total_rec_int	33
total_rec_late_fee	33
recoveries	33
collection_recovery_fee	33
last_pymnt_d	2460
last_pymnt_amnt	33
next_pymnt_d	1345343
last_credit_pull_d	105
last_fico_range_high	33
last_fico_range_low	33
collections_12_mths_ex_med	178
mths_since_last_major_derog	1679926
policy_code	33
application_type	33
annual_inc_joint	2139991
dti_joint	2139995
verification_status_joint	2144971
acc_now_delinq	62
tot_coll_amt	70309
tot_cur_bal	70309
open_acc_6m	866163
open_act_il	866162
open_il_12m	866162
open_il_24m	866162
mths_since_rcnt_il	909957
total_bal_il	866162
il_util	1068883
open_rv_12m	866162
open_rv_24m	866162
max_bal_bc	866162
all_util	866381
total_rev_hi_lim	70309
inq_fi	866162
total_cu_tl	866163
inq_last_12m	866163
acc_open_past_24mths	50063
avg_cur_bal	70379
bc_open_to_buy	74968
bc_util	76104
chargeoff_within_12_mths	178
delinq_amnt	62
mo_sin_old_il_acct	139104
mo_sin_old_rev_tl_op	70310
mo_sin_rcnt_rev_tl_op	70310
mo_sin_rcnt_tl	70309
mort_acc	50063
mths_since_recent_bc	73445
mths_since_recent_bc_dlq	1741000
mths_since_recent_inq	295468
mths_since_recent_revol_delinq	1520342
num_accts_ever_120_pd	70309
num_actv_bc_tl	70309
num_actv_rev_tl	70309
num_bc_sats	58623
num_bc_tl	70309
num_il_tl	70309
num_op_rev_tl	70309
num_rev_accts	70310
num_rev_tl_bal_gt_0	70309
num_sats	58623
num_tl_120dpd_2m	153690
num_tl_30dpd	70309
num_tl_90g_dpd_24m	70309
num_tl_op_past_12m	70309

```
pct_tl_nvr_dlq                                  70464
percent_bc_gt_75                                75412
pub_rec_bankruptcies                             1398
tax_liens                                         138
tot_hi_cred_lim                                 70309
total_bal_ex_mort                               50063
total_bc_limit                                  50063
total_il_high_credit_limit                      70309
revol_bal_joint                               2152681
sec_app_fico_range_low                        2152680
sec_app_fico_range_high                       2152680
sec_app_earliest_cr_line                      2152680
sec_app_inq_last_6mths                        2152680
sec_app_mort_acc                              2152680
sec_app_open_acc                              2152680
sec_app_revol_util                            2154517
sec_app_open_act_il                           2152680
sec_app_num_rev_accts                         2152680
sec_app_chargeoff_within_12_mths              2152680
sec_app_collections_12_mths_ex_med            2152680
sec_app_mths_since_last_major_derog           2224759
hardship_flag                                      33
hardship_type                                 2249784
hardship_reason                               2249784
hardship_status                               2249784
deferral_term                                 2249784
hardship_amount                               2249784
hardship_start_date                           2249784
hardship_end_date                             2249784
payment_plan_start_date                       2249784
hardship_length                               2249784
hardship_dpd                                  2249784
hardship_loan_status                          2249784
orig_projected_additional_accrued_interest    2252050
hardship_payoff_balance_amount                2249784
hardship_last_payment_amount                  2249784
disbursement_method                                33
debt_settlement_flag                               33
debt_settlement_flag_date                     2226455
settlement_status                             2226455
settlement_date                               2226455
settlement_amount                             2226455
settlement_percentage                         2226455
settlement_term                               2226455
dtype: int64
```

Columnwise, find percentage of NULL Values

In [8]:

```
percent_df = df.isna().mean()
```

In [9]:

```
percent_df*100
```

Out[9]:

```
id                                      0.000000
member_id                             100.000000
loan_amnt                               0.001460
funded_amnt                             0.001460
funded_amnt_inv                         0.001460
term                                    0.001460
int_rate                                0.001460
installment                             0.001460
grade                                   0.001460
sub_grade                               0.001460
emp_title                               7.387178
emp_length                              6.499754
home_ownership                          0.001460
annual_inc                              0.001637
verification_status                     0.001460
issue_d                                 0.001460
loan_status                             0.001460
pymnt_plan                              0.001460
url                                     0.001460
desc                                   94.423544
purpose                                 0.001460
title                                   1.033219
zip_code                                0.001504
addr_state                              0.001460
dti                                     0.077144
delinq_2yrs                             0.002743
earliest_cr_line                        0.002743
fico_range_low                          0.001460
fico_range_high                         0.001460
```

```
inq_last_6mths                              0.002787
mths_since_last_delinq                     51.246715
mths_since_last_record                     84.113069
open_acc                                    0.002743
pub_rec                                     0.002743
revol_bal                                   0.001460
revol_util                                  0.081170
total_acc                                   0.002743
initial_list_status                         0.001460
out_prncp                                   0.001460
out_prncp_inv                               0.001460
total_pymnt                                 0.001460
total_pymnt_inv                             0.001460
total_rec_prncp                             0.001460
total_rec_int                               0.001460
total_rec_late_fee                          0.001460
recoveries                                  0.001460
collection_recovery_fee                     0.001460
last_pymnt_d                                0.108816
last_pymnt_amnt                             0.001460
next_pymnt_d                               59.509993
last_credit_pull_d                          0.004645
last_fico_range_high                        0.001460
last_fico_range_low                         0.001460
collections_12_mths_ex_med                  0.007874
mths_since_last_major_derog                74.309960
policy_code                                 0.001460
application_type                            0.001460
annual_inc_joint                           94.660506
dti_joint                                  94.660683
verification_status_joint                  94.880791
acc_now_delinq                              0.002743
tot_coll_amt                                3.110053
tot_cur_bal                                 3.110053
open_acc_6m                                38.313912
open_act_il                                38.313868
open_il_12m                                38.313868
open_il_24m                                38.313868
mths_since_rcnt_il                         40.251099
total_bal_il                               38.313868
il_util                                    47.281042
open_rv_12m                                38.313868
open_rv_24m                                38.313868
max_bal_bc                                 38.313868
all_util                                   38.323555
total_rev_hi_lim                            3.110053
inq_fi                                     38.313868
total_cu_tl                                38.313912
inq_last_12m                               38.313912
acc_open_past_24mths                        2.214490
avg_cur_bal                                 3.113149
bc_open_to_buy                              3.316140
bc_util                                     3.366389
chargeoff_within_12_mths                    0.007874
delinq_amnt                                 0.002743
mo_sin_old_il_acct                          6.153136
mo_sin_old_rev_tl_op                        3.110097
mo_sin_rcnt_rev_tl_op                       3.110097
mo_sin_rcnt_tl                              3.110053
mort_acc                                    2.214490
mths_since_recent_bc                        3.248771
mths_since_recent_bc_dlq                   77.011511
mths_since_recent_inq                      13.069751
mths_since_recent_revol_delinq             67.250910
num_accts_ever_120_pd                       3.110053
num_actv_bc_tl                              3.110053
num_actv_rev_tl                             3.110053
num_bc_sats                                 2.593134
num_bc_tl                                   3.110053
num_il_tl                                   3.110053
num_op_rev_tl                               3.110053
num_rev_accts                               3.110097
num_rev_tl_bal_gt_0                         3.110053
num_sats                                    2.593134
num_tl_120dpd_2m                            6.798334
num_tl_30dpd                                3.110053
num_tl_90g_dpd_24m                          3.110053
num_tl_op_past_12m                          3.110053
pct_tl_nvr_dlq                              3.116909
percent_bc_gt_75                            3.335779
pub_rec_bankruptcies                        0.061839
tax_liens                                   0.006104
tot_hi_cred_lim                             3.110053
total_bal_ex_mort                           2.214490
total_bc_limit                              2.214490
total_il_high_credit_limit                  3.110053
revol_bal_joint                            95.221836
sec_app_fico_range_low                     95.221792
sec_app_fico_range_high                    95.221792
```

```
sec_app_earliest_cr_line                       95.221792
sec_app_inq_last_6mths                         95.221792
sec_app_mort_acc                               95.221792
sec_app_open_acc                               95.221792
sec_app_revol_util                             95.303050
sec_app_open_act_il                            95.221792
sec_app_num_rev_accts                          95.221792
sec_app_chargeoff_within_12_mths               95.221792
sec_app_collections_12_mths_ex_med             95.221792
sec_app_mths_since_last_major_derog            98.410139
hardship_flag                                   0.001460
hardship_type                                  99.517097
hardship_reason                                99.517097
hardship_status                                99.517097
deferral_term                                  99.517097
hardship_amount                                99.517097
hardship_start_date                            99.517097
hardship_end_date                              99.517097
payment_plan_start_date                        99.517097
hardship_length                                99.517097
hardship_dpd                                   99.517097
hardship_loan_status                           99.517097
orig_projected_additional_accrued_interest     99.617331
hardship_payoff_balance_amount                 99.517097
hardship_last_payment_amount                   99.517097
disbursement_method                             0.001460
debt_settlement_flag                            0.001460
debt_settlement_flag_date                      98.485160
settlement_status                              98.485160
settlement_date                                98.485160
settlement_amount                              98.485160
settlement_percentage                          98.485160
settlement_term                                98.485160
dtype: float64
```

filter percentages less than 20%

In [10]:
```
x = percent_df[percent_df*100>20]   # Means they have more than 20% NANs
```

In [11]:
```
x.index  # When you do .index we get  columns as  a series
```

Out[11]:
```
Index(['member_id', 'desc', 'mths_since_last_delinq', 'mths_since_last_record',
       'next_pymnt_d', 'mths_since_last_major_derog', 'annual_inc_joint',
       'dti_joint', 'verification_status_joint', 'open_acc_6m', 'open_act_il',
       'open_il_12m', 'open_il_24m', 'mths_since_rcnt_il', 'total_bal_il',
       'il_util', 'open_rv_12m', 'open_rv_24m', 'max_bal_bc', 'all_util',
       'inq_fi', 'total_cu_tl', 'inq_last_12m', 'mths_since_recent_bc_dlq',
       'mths_since_recent_revol_delinq', 'revol_bal_joint',
       'sec_app_fico_range_low', 'sec_app_fico_range_high',
       'sec_app_earliest_cr_line', 'sec_app_inq_last_6mths',
       'sec_app_mort_acc', 'sec_app_open_acc', 'sec_app_revol_util',
       'sec_app_open_act_il', 'sec_app_num_rev_accts',
       'sec_app_chargeoff_within_12_mths',
       'sec_app_collections_12_mths_ex_med',
       'sec_app_mths_since_last_major_derog', 'hardship_type',
       'hardship_reason', 'hardship_status', 'deferral_term',
       'hardship_amount', 'hardship_start_date', 'hardship_end_date',
       'payment_plan_start_date', 'hardship_length', 'hardship_dpd',
       'hardship_loan_status', 'orig_projected_additional_accrued_interest',
       'hardship_payoff_balance_amount', 'hardship_last_payment_amount',
       'debt_settlement_flag_date', 'settlement_status', 'settlement_date',
       'settlement_amount', 'settlement_percentage', 'settlement_term'],
      dtype='object')
```

In [12]:
```
len(x.index)          # always use len to get the number of columns on an index
```

Out[12]:
```
58
```

In [13]:
```
type(x)
```

Out[13]:
```
pandas.core.series.Series
```

In [14]:

```
df.drop(x.index,axis=1,inplace=True)    # drop all the columns for NAN >20%
```

In [15]:
```
df.shape    #  NAN - After dropping count has changed
```

Out[15]: (2260701, 93)

In [16]:
```
151-58          #dropped all 58 columns
```

Out[16]: 93

How to deal with NA values columnwise

In [17]:
```
df.head()
```

Out[17]:

	id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_owners
0	68407277	3600.0	3600.0	3600.0	36 months	13.99	123.03	C	C4	leadman	10+ years	MORTGA
1	68355089	24700.0	24700.0	24700.0	36 months	11.99	820.28	C	C1	Engineer	10+ years	MORTGA
2	68341763	20000.0	20000.0	20000.0	60 months	10.78	432.66	B	B4	truck driver	10+ years	MORTGA
3	66310712	35000.0	35000.0	35000.0	60 months	14.85	829.90	C	C5	Information Systems Officer	10+ years	MORTGA
4	68476807	10400.0	10400.0	10400.0	60 months	22.45	289.91	F	F1	Contract Specialist	3 years	MORTGA

In [18]:
```
df.head(2).T  # Transpose taken just for 2 records
```

Out[18]:

	0	1
id	68407277	68355089
loan_amnt	3600.0	24700.0
funded_amnt	3600.0	24700.0
funded_amnt_inv	3600.0	24700.0
term	36 months	36 months
int_rate	13.99	11.99
installment	123.03	820.28
grade	C	C
sub_grade	C4	C1
emp_title	leadman	Engineer
emp_length	10+ years	10+ years
home_ownership	MORTGAGE	MORTGAGE
annual_inc	55000.0	65000.0
verification_status	Not Verified	Not Verified
issue_d	Dec-2015	Dec-2015
loan_status	Fully Paid	Fully Paid
pymnt_plan	n	n
url	https://lendingclub.com/browse/loanDetail.acti...	https://lendingclub.com/browse/loanDetail.acti...
purpose	debt_consolidation	small_business
title	Debt consolidation	Business
zip_code	190xx	577xx
addr_state	PA	SD
dti	5.91	16.06
delinq_2yrs	0.0	1.0
earliest_cr_line	Aug-2003	Dec-1999

fico_range_low	675.0	715.0
fico_range_high	679.0	719.0
inq_last_6mths	1.0	4.0
open_acc	7.0	22.0
pub_rec	0.0	0.0
revol_bal	2765.0	21470.0
revol_util	29.7	19.2
total_acc	13.0	38.0
initial_list_status	w	w
out_prncp	0.0	0.0
out_prncp_inv	0.0	0.0
total_pymnt	4421.723917	25679.66
total_pymnt_inv	4421.72	25679.66
total_rec_prncp	3600.0	24700.0
total_rec_int	821.72	979.66
total_rec_late_fee	0.0	0.0
recoveries	0.0	0.0
collection_recovery_fee	0.0	0.0
last_pymnt_d	Jan-2019	Jun-2016
last_pymnt_amnt	122.67	926.35
last_credit_pull_d	Mar-2019	Mar-2019
last_fico_range_high	564.0	699.0
last_fico_range_low	560.0	695.0
collections_12_mths_ex_med	0.0	0.0
policy_code	1.0	1.0
application_type	Individual	Individual
acc_now_delinq	0.0	0.0
tot_coll_amt	722.0	0.0
tot_cur_bal	144904.0	204396.0
total_rev_hi_lim	9300.0	111800.0
acc_open_past_24mths	4.0	4.0
avg_cur_bal	20701.0	9733.0
bc_open_to_buy	1506.0	57830.0
bc_util	37.2	27.1
chargeoff_within_12_mths	0.0	0.0
delinq_amnt	0.0	0.0
mo_sin_old_il_acct	148.0	113.0
mo_sin_old_rev_tl_op	128.0	192.0
mo_sin_rcnt_rev_tl_op	3.0	2.0
mo_sin_rcnt_tl	3.0	2.0
mort_acc	1.0	4.0
mths_since_recent_bc	4.0	2.0
mths_since_recent_inq	4.0	0.0
num_accts_ever_120_pd	2.0	0.0
num_actv_bc_tl	2.0	5.0
num_actv_rev_tl	4.0	5.0
num_bc_sats	2.0	13.0
num_bc_tl	5.0	17.0
num_il_tl	3.0	6.0
num_op_rev_tl	4.0	20.0
num_rev_accts	9.0	27.0
num_rev_tl_bal_gt_0	4.0	5.0
num_sats	7.0	22.0
num_tl_120dpd_2m	0.0	0.0
num_tl_30dpd	0.0	0.0

num_tl_90g_dpd_24m	0.0	0.0
num_tl_op_past_12m	3.0	2.0
pct_tl_nvr_dlq	76.9	97.4
percent_bc_gt_75	0.0	7.7
pub_rec_bankruptcies	0.0	0.0
tax_liens	0.0	0.0
tot_hi_cred_lim	178050.0	314017.0
total_bal_ex_mort	7746.0	39475.0
total_bc_limit	2400.0	79300.0
total_il_high_credit_limit	13734.0	24667.0
hardship_flag	N	N
disbursement_method	Cash	Cash
debt_settlement_flag	N	N

In [19]:

```
df['emp_length'].value_counts()      #Check the value count for the particular columnn to modify tat column
```

Out[19]:

```
10+ years    748005
2 years      203677
< 1 year     189988
3 years      180753
1 year       148403
5 years      139698
4 years      136605
6 years      102628
7 years       92695
8 years       91914
9 years       79395
Name: emp_length, dtype: int64
```

In [20]:

```
df.isna().sum()
```

Out[20]:

```
id                             0
loan_amnt                     33
funded_amnt                   33
funded_amnt_inv               33
term                          33
int_rate                      33
installment                   33
grade                         33
sub_grade                     33
emp_title                 167002
emp_length                146940
home_ownership                33
annual_inc                    37
verification_status           33
issue_d                       33
loan_status                   33
pymnt_plan                    33
url                           33
purpose                       33
title                      23358
zip_code                      34
addr_state                    33
dti                         1744
delinq_2yrs                   62
earliest_cr_line              62
fico_range_low                33
fico_range_high               33
inq_last_6mths                63
open_acc                      62
pub_rec                       62
revol_bal                     33
revol_util                  1835
total_acc                     62
initial_list_status           33
out_prncp                     33
out_prncp_inv                 33
total_pymnt                   33
total_pymnt_inv               33
total_rec_prncp               33
total_rec_int                 33
total_rec_late_fee            33
recoveries                    33
collection_recovery_fee       33
```

```
last_pymnt_d                   2460
last_pymnt_amnt                  33
last_credit_pull_d              105
last_fico_range_high             33
last_fico_range_low              33
collections_12_mths_ex_med      178
policy_code                      33
application_type                 33
acc_now_delinq                   62
tot_coll_amt                  70309
tot_cur_bal                   70309
total_rev_hi_lim              70309
acc_open_past_24mths          50063
avg_cur_bal                   70379
bc_open_to_buy                74968
bc_util                       76104
chargeoff_within_12_mths        178
delinq_amnt                      62
mo_sin_old_il_acct           139104
mo_sin_old_rev_tl_op          70310
mo_sin_rcnt_rev_tl_op         70310
mo_sin_rcnt_tl                70309
mort_acc                      50063
mths_since_recent_bc          73445
mths_since_recent_inq        295468
num_accts_ever_120_pd         70309
num_actv_bc_tl                70309
num_actv_rev_tl               70309
num_bc_sats                   58623
num_bc_tl                     70309
num_il_tl                     70309
num_op_rev_tl                 70309
num_rev_accts                 70310
num_rev_tl_bal_gt_0           70309
num_sats                      58623
num_tl_120dpd_2m             153690
num_tl_30dpd                  70309
num_tl_90g_dpd_24m            70309
num_tl_op_past_12m            70309
pct_tl_nvr_dlq                70464
percent_bc_gt_75              75412
pub_rec_bankruptcies           1398
tax_liens                       138
tot_hi_cred_lim               70309
total_bal_ex_mort             50063
total_bc_limit                50063
total_il_high_credit_limit    70309
hardship_flag                    33
disbursement_method              33
debt_settlement_flag             33
dtype: int64
```

In [21]:

```
# Create a list of columns that can be deleted which doesnt make sense
```

In [22]:

```
delete_cols = ['id','home_ownership','verification_status','issue_d','url','title','zip_code',"addr_state",'earli
'last_pymnt_d','last_pymnt_amnt','last_credit_pull_d']
```

In [23]:

```
set(x.index).intersection(set(delete_cols))  #Did an intersection with an existing x.index for any overlaps
```

Out[23]:

```
set()
```

In [24]:

```
df.drop(delete_cols,axis=1,inplace=True)
```

Categorize loan_status column for fully paid and charged off

In [25]:

```
df.columns
```

Out[25]:

```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term', 'int_rate',
       'installment', 'grade', 'sub_grade', 'emp_title', 'emp_length',
       'annual_inc', 'loan_status', 'pymnt_plan', 'purpose', 'dti',
       'delinq_2yrs', 'fico_range_low', 'fico_range_high', 'inq_last_6mths',
       'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'initial_list_status', 'out_prncp', 'out_prncp_inv', 'total_pymnt',
       'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
       'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
```

```
        'last_fico_range_high', 'last_fico_range_low',
        'collections_12_mths_ex_med', 'policy_code', 'application_type',
        'acc_now_delinq', 'tot_coll_amt', 'tot_cur_bal', 'total_rev_hi_lim',
        'acc_open_past_24mths', 'avg_cur_bal', 'bc_open_to_buy', 'bc_util',
        'chargeoff_within_12_mths', 'delinq_amnt', 'mo_sin_old_il_acct',
        'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op', 'mo_sin_rcnt_tl',
        'mort_acc', 'mths_since_recent_bc', 'mths_since_recent_inq',
        'num_accts_ever_120_pd', 'num_actv_bc_tl', 'num_actv_rev_tl',
        'num_bc_sats', 'num_bc_tl', 'num_il_tl', 'num_op_rev_tl',
        'num_rev_accts', 'num_rev_tl_bal_gt_0', 'num_sats', 'num_tl_120dpd_2m',
        'num_tl_30dpd', 'num_tl_90g_dpd_24m', 'num_tl_op_past_12m',
        'pct_tl_nvr_dlq', 'percent_bc_gt_75', 'pub_rec_bankruptcies',
        'tax_liens', 'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit',
        'total_il_high_credit_limit', 'hardship_flag', 'disbursement_method',
        'debt_settlement_flag'],
       dtype='object')
```

In [26]:

```
df['loan_status'].value_counts()
```

Out[26]:

```
Fully Paid                                             1076751
Current                                                 878317
Charged Off                                             268559
Late (31-120 days)                                       21467
In Grace Period                                           8436
Late (16-30 days)                                         4349
Does not meet the credit policy. Status:Fully Paid        1988
Does not meet the credit policy. Status:Charged Off        761
Default                                                     40
Name: loan_status, dtype: int64
```

In [27]:

```
df['loan_status'] = df['loan_status'].replace(['Does not meet the credit policy. Status:Fully Paid'],'Fully Paid'
```

In [28]:

```
df['loan_status'] = df['loan_status'].replace(['Does not meet the credit policy. Status:Charged Off'],'Charged Of
```

In [29]:

```
df['loan_status'].value_counts()
```

Out[29]:

```
Fully Paid            1078739
Current                878317
Charged Off            269320
Late (31-120 days)      21467
In Grace Period          8436
Late (16-30 days)        4349
Default                    40
Name: loan_status, dtype: int64
```

In [30]:

```
filt_df = df[(df['loan_status'] == 'Fully Paid') | (df['loan_status'] == 'Charged Off')]
```

In [31]:

```
filt_df.shape                                    #(2260701, 93) - older count
```

Out[31]:

```
(1348059, 81)
```

In [32]:

```
filt_df['loan_status'].value_counts()
```

Out[32]:

```
Fully Paid     1078739
Charged Off     269320
Name: loan_status, dtype: int64
```

In [33]:

```
filt_df.head()
```

Out[33]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	annual_inc	loan_status
0	3600.0	3600.0	3600.0	36 months	13.99	123.03	C	C4	leadman	10+ years	55000.0	Fully Paid
1	24700.0	24700.0	24700.0	36 months	11.99	820.28	C	C1	Engineer	10+ years	65000.0	Fully Paid
2	20000.0	20000.0	20000.0	60 months	10.78	432.66	B	B4	truck driver	10+ years	63000.0	Fully Paid

4	10400.0	10400.0	10400.0	60 months	22.45	289.91	F	F1	Contract Specialist	3 years	104433.0	Fully Paid
5	11950.0	11950.0	11950.0	36 months	13.44	405.18	C	C3	Veterinary Tecnician	4 years	34000.0	Fully Paid

NAN Treatment

In [34]:

```
filt_df.isna().mean()
```

Out[34]:

```
loan_amnt                     0.000000
funded_amnt                   0.000000
funded_amnt_inv               0.000000
term                          0.000000
int_rate                      0.000000
installment                   0.000000
grade                         0.000000
sub_grade                     0.000000
emp_title                     0.063754
emp_length                    0.058265
annual_inc                    0.000003
loan_status                   0.000000
pymnt_plan                    0.000000
purpose                       0.000000
dti                           0.000277
delinq_2yrs                   0.000022
fico_range_low                0.000000
fico_range_high               0.000000
inq_last_6mths                0.000022
open_acc                      0.000022
pub_rec                       0.000022
revol_bal                     0.000000
revol_util                    0.000665
total_acc                     0.000022
initial_list_status           0.000000
out_prncp                     0.000000
out_prncp_inv                 0.000000
total_pymnt                   0.000000
total_pymnt_inv               0.000000
total_rec_prncp               0.000000
total_rec_int                 0.000000
total_rec_late_fee            0.000000
recoveries                    0.000000
collection_recovery_fee       0.000000
last_fico_range_high          0.000000
last_fico_range_low           0.000000
collections_12_mths_ex_med    0.000108
policy_code                   0.000000
application_type              0.000000
acc_now_delinq                0.000022
tot_coll_amt                  0.052131
tot_cur_bal                   0.052131
total_rev_hi_lim              0.052131
acc_open_past_24mths          0.037113
avg_cur_bal                   0.052148
bc_open_to_buy                0.047396
bc_util                       0.047966
chargeoff_within_12_mths      0.000108
delinq_amnt                   0.000022
mo_sin_old_il_acct            0.080356
mo_sin_old_rev_tl_op          0.052132
mo_sin_rcnt_rev_tl_op         0.052132
mo_sin_rcnt_tl                0.052131
mort_acc                      0.037113
mths_since_recent_bc          0.046712
mths_since_recent_inq         0.131166
num_accts_ever_120_pd         0.052131
num_actv_bc_tl                0.052131
num_actv_rev_tl               0.052131
num_bc_sats                   0.043462
num_bc_tl                     0.052131
num_il_tl                     0.052131
num_op_rev_tl                 0.052131
num_rev_accts                 0.052132
num_rev_tl_bal_gt_0           0.052131
num_sats                      0.043462
num_tl_120dpd_2m              0.089128
num_tl_30dpd                  0.052131
num_tl_90g_dpd_24m            0.052131
num_tl_op_past_12m            0.052131
pct_tl_nvr_dlq                0.052245
percent_bc_gt_75              0.047701
pub_rec_bankruptcies          0.001013
```

```
tax_liens                        0.000078
tot_hi_cred_lim                  0.052131
total_bal_ex_mort                0.037113
total_bc_limit                   0.037113
total_il_high_credit_limit       0.052131
hardship_flag                    0.000000
disbursement_method              0.000000
debt_settlement_flag             0.000000
dtype: float64
```

In [35]:

```
#The below columns doesnt make sense and hence drop it

filt_df.drop('emp_title',axis=1,inplace=True)
filt_df.drop('emp_length',axis=1,inplace=True)
```

Check categorical column that is with object -all strings

In [36]:

```
cat_col = list(filt_df.select_dtypes('object'))            #all strings
```

Check numerical column that is with float64

In [37]:

```
num_col = list(filt_df.select_dtypes('float64'))                #all floats
```

In [38]:

```
x = filt_df[num_col].isna().mean()
```

In [39]:

```
x
```

Out[39]:

```
loan_amnt                        0.000000
funded_amnt                      0.000000
funded_amnt_inv                  0.000000
int_rate                         0.000000
installment                      0.000000
annual_inc                       0.000003
dti                              0.000277
delinq_2yrs                      0.000022
fico_range_low                   0.000000
fico_range_high                  0.000000
inq_last_6mths                   0.000022
open_acc                         0.000022
pub_rec                          0.000022
revol_bal                        0.000000
revol_util                       0.000665
total_acc                        0.000022
out_prncp                        0.000000
out_prncp_inv                    0.000000
total_pymnt                      0.000000
total_pymnt_inv                  0.000000
total_rec_prncp                  0.000000
total_rec_int                    0.000000
total_rec_late_fee               0.000000
recoveries                       0.000000
collection_recovery_fee          0.000000
last_fico_range_high             0.000000
last_fico_range_low              0.000000
collections_12_mths_ex_med       0.000108
policy_code                      0.000000
acc_now_delinq                   0.000022
tot_coll_amt                     0.052131
tot_cur_bal                      0.052131
total_rev_hi_lim                 0.052131
acc_open_past_24mths             0.037113
avg_cur_bal                      0.052148
bc_open_to_buy                   0.047396
bc_util                          0.047966
chargeoff_within_12_mths         0.000108
delinq_amnt                      0.000022
mo_sin_old_il_acct               0.080356
mo_sin_old_rev_tl_op             0.052132
mo_sin_rcnt_rev_tl_op            0.052132
mo_sin_rcnt_tl                   0.052131
mort_acc                         0.037113
mths_since_recent_bc             0.046712
mths_since_recent_inq            0.131166
num_accts_ever_120_pd            0.052131
num_actv_bc_tl                   0.052131
```

```
num_actv_rev_tl              0.052131
num_bc_sats                  0.043462
num_bc_tl                    0.052131
num_il_tl                    0.052131
num_op_rev_tl                0.052131
num_rev_accts                0.052132
num_rev_tl_bal_gt_0          0.052131
num_sats                     0.043462
num_tl_120dpd_2m             0.089128
num_tl_30dpd                 0.052131
num_tl_90g_dpd_24m           0.052131
num_tl_op_past_12m           0.052131
pct_tl_nvr_dlq               0.052245
percent_bc_gt_75             0.047701
pub_rec_bankruptcies         0.001013
tax_liens                    0.000078
tot_hi_cred_lim              0.052131
total_bal_ex_mort            0.037113
total_bc_limit               0.037113
total_il_high_credit_limit   0.052131
dtype: float64
```

In [40]:
```
x[x>0].index
```

Out[40]:
```
Index(['annual_inc', 'dti', 'delinq_2yrs', 'inq_last_6mths', 'open_acc',
       'pub_rec', 'revol_util', 'total_acc', 'collections_12_mths_ex_med',
       'acc_now_delinq', 'tot_coll_amt', 'tot_cur_bal', 'total_rev_hi_lim',
       'acc_open_past_24mths', 'avg_cur_bal', 'bc_open_to_buy', 'bc_util',
       'chargeoff_within_12_mths', 'delinq_amnt', 'mo_sin_old_il_acct',
       'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op', 'mo_sin_rcnt_tl',
       'mort_acc', 'mths_since_recent_bc', 'mths_since_recent_inq',
       'num_accts_ever_120_pd', 'num_actv_bc_tl', 'num_actv_rev_tl',
       'num_bc_sats', 'num_bc_tl', 'num_il_tl', 'num_op_rev_tl',
       'num_rev_accts', 'num_rev_tl_bal_gt_0', 'num_sats', 'num_tl_120dpd_2m',
       'num_tl_30dpd', 'num_tl_90g_dpd_24m', 'num_tl_op_past_12m',
       'pct_tl_nvr_dlq', 'percent_bc_gt_75', 'pub_rec_bankruptcies',
       'tax_liens', 'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit',
       'total_il_high_credit_limit'],
      dtype='object')
```

In [41]:
```
nan_columns= x[x>0].index

# write the columns to variable   #type(nan_columns)
```

In [42]:
```
filt_df['num_sats'].isna().sum()
```

Out[42]:
```
58590
```

In [43]:
```
filt_df['num_sats'].fillna(filt_df['num_sats'].mean()).head() #replace num_stats NAN values with mean values
```

Out[43]:
```
0     7.0
1    22.0
2     6.0
4    12.0
5     5.0
Name: num_sats, dtype: float64
```

In [44]:
```
for col in nan_columns:
    filt_df[col] = filt_df[col].fillna(filt_df[col].mean())   # replace nan columns with mean values
```

In [45]:
```
filt_df[nan_columns].isna().sum()          # All nan columns have become 0
```

Out[45]:
```
annual_inc                    0
dti                           0
delinq_2yrs                   0
inq_last_6mths                0
open_acc                      0
pub_rec                       0
revol_util                    0
total_acc                     0
collections_12_mths_ex_med    0
acc_now_delinq                0
tot_coll_amt                  0
```

```
tot_cur_bal                   0
total_rev_hi_lim              0
acc_open_past_24mths          0
avg_cur_bal                   0
bc_open_to_buy                0
bc_util                       0
chargeoff_within_12_mths      0
delinq_amnt                   0
mo_sin_old_il_acct            0
mo_sin_old_rev_tl_op          0
mo_sin_rcnt_rev_tl_op         0
mo_sin_rcnt_tl                0
mort_acc                      0
mths_since_recent_bc          0
mths_since_recent_inq         0
num_accts_ever_120_pd         0
num_actv_bc_tl                0
num_actv_rev_tl               0
num_bc_sats                   0
num_bc_tl                     0
num_il_tl                     0
num_op_rev_tl                 0
num_rev_accts                 0
num_rev_tl_bal_gt_0           0
num_sats                      0
num_tl_120dpd_2m              0
num_tl_30dpd                  0
num_tl_90g_dpd_24m            0
num_tl_op_past_12m            0
pct_tl_nvr_dlq                0
percent_bc_gt_75              0
pub_rec_bankruptcies          0
tax_liens                     0
tot_hi_cred_lim               0
total_bal_ex_mort             0
total_bc_limit                0
total_il_high_credit_limit    0
dtype: int64
```

In [46]:
```
for col in cat_col:
    filt_df[col] = filt_df[col].fillna(filt_df[col].mode())  # Replace cat_col using mode
```

In [47]:
```
filt_df[cat_col].isna().sum()
```

Out[47]:
```
term                    0
grade                   0
sub_grade               0
loan_status             0
pymnt_plan              0
purpose                 0
initial_list_status     0
application_type        0
hardship_flag           0
disbursement_method     0
debt_settlement_flag    0
dtype: int64
```

In [48]:
```
#filt_df['emp_title'].nunique()
```

In [49]:
```
filt_df.select_dtypes('object').columns
```

Out[49]:
```
Index(['term', 'grade', 'sub_grade', 'loan_status', 'pymnt_plan', 'purpose',
       'initial_list_status', 'application_type', 'hardship_flag',
       'disbursement_method', 'debt_settlement_flag'],
      dtype='object')
```

In [50]:
```
filt_df.isna().mean()*100
```

Out[50]:
```
loan_amnt                     0.0
funded_amnt                   0.0
funded_amnt_inv               0.0
term                          0.0
int_rate                      0.0
installment                   0.0
grade                         0.0
sub_grade                     0.0
```

```
annual_inc                    0.0
loan_status                   0.0
pymnt_plan                    0.0
purpose                       0.0
dti                           0.0
delinq_2yrs                   0.0
fico_range_low                0.0
fico_range_high               0.0
inq_last_6mths                0.0
open_acc                      0.0
pub_rec                       0.0
revol_bal                     0.0
revol_util                    0.0
total_acc                     0.0
initial_list_status           0.0
out_prncp                     0.0
out_prncp_inv                 0.0
total_pymnt                   0.0
total_pymnt_inv               0.0
total_rec_prncp               0.0
total_rec_int                 0.0
total_rec_late_fee            0.0
recoveries                    0.0
collection_recovery_fee       0.0
last_fico_range_high          0.0
last_fico_range_low           0.0
collections_12_mths_ex_med    0.0
policy_code                   0.0
application_type              0.0
acc_now_delinq                0.0
tot_coll_amt                  0.0
tot_cur_bal                   0.0
total_rev_hi_lim              0.0
acc_open_past_24mths          0.0
avg_cur_bal                   0.0
bc_open_to_buy                0.0
bc_util                       0.0
chargeoff_within_12_mths      0.0
delinq_amnt                   0.0
mo_sin_old_il_acct            0.0
mo_sin_old_rev_tl_op          0.0
mo_sin_rcnt_rev_tl_op         0.0
mo_sin_rcnt_tl                0.0
mort_acc                      0.0
mths_since_recent_bc          0.0
mths_since_recent_inq         0.0
num_accts_ever_120_pd         0.0
num_actv_bc_tl                0.0
num_actv_rev_tl               0.0
num_bc_sats                   0.0
num_bc_tl                     0.0
num_il_tl                     0.0
num_op_rev_tl                 0.0
num_rev_accts                 0.0
num_rev_tl_bal_gt_0           0.0
num_sats                      0.0
num_tl_120dpd_2m              0.0
num_tl_30dpd                  0.0
num_tl_90g_dpd_24m            0.0
num_tl_op_past_12m            0.0
pct_tl_nvr_dlq                0.0
percent_bc_gt_75              0.0
pub_rec_bankruptcies          0.0
tax_liens                     0.0
tot_hi_cred_lim               0.0
total_bal_ex_mort             0.0
total_bc_limit                0.0
total_il_high_credit_limit    0.0
hardship_flag                 0.0
disbursement_method           0.0
debt_settlement_flag          0.0
dtype: float64
```

In [51]:

```
filt_df.shape   #All objects sorted
```

Out[51]:

```
(1348059, 79)
```

Analysis of Categorical data

Deal with categorical Values

In [52]:
```
filt_df.select_dtypes('float64').nunique()   # Checking for float64 object type
```

Out[52]:
```
loan_amnt                       1560
funded_amnt                     1560
funded_amnt_inv                10041
int_rate                         672
installment                    83530
annual_inc                     64463
dti                             7068
delinq_2yrs                       32
fico_range_low                    48
fico_range_high                   48
inq_last_6mths                    29
open_acc                          85
pub_rec                           38
revol_bal                      84084
revol_util                      1380
total_acc                        144
out_prncp                          1
out_prncp_inv                      1
total_pymnt                  1264345
total_pymnt_inv              1014374
total_rec_prncp               205465
total_rec_int                 517547
total_rec_late_fee             16230
recoveries                    132777
collection_recovery_fee       146222
last_fico_range_high              72
last_fico_range_low               71
collections_12_mths_ex_med        16
policy_code                        1
acc_now_delinq                     9
tot_coll_amt                   12872
tot_cur_bal                   400460
total_rev_hi_lim               26800
acc_open_past_24mths              56
avg_cur_bal                    76865
bc_open_to_buy                 74925
bc_util                         1445
chargeoff_within_12_mths          12
delinq_amnt                     2007
mo_sin_old_il_acct               523
mo_sin_old_rev_tl_op             757
mo_sin_rcnt_rev_tl_op            287
mo_sin_rcnt_tl                   197
mort_acc                          40
mths_since_recent_bc             492
mths_since_recent_inq             27
num_accts_ever_120_pd             40
num_actv_bc_tl                    35
num_actv_rev_tl                   53
num_bc_sats                       53
num_bc_tl                         69
num_il_tl                        115
num_op_rev_tl                     75
num_rev_accts                    108
num_rev_tl_bal_gt_0               47
num_sats                          84
num_tl_120dpd_2m                   7
num_tl_30dpd                       6
num_tl_90g_dpd_24m                30
num_tl_op_past_12m                33
pct_tl_nvr_dlq                   632
percent_bc_gt_75                 246
pub_rec_bankruptcies              13
tax_liens                         37
tot_hi_cred_lim               427441
total_bal_ex_mort             177996
total_bc_limit                 17087
total_il_high_credit_limit    162549
dtype: int64
```

category to numerical

In [53]:
```
filt_df[cat_col].nunique()
```

Out[53]:
```
term                   2
grade                  7
sub_grade             35
loan_status            2
```

```
pymnt_plan                1
purpose                  14
initial_list_status       2
application_type          2
hardship_flag             1
disbursement_method       2
debt_settlement_flag      2
dtype: int64
```

In [54]:

```
from tqdm import tqdm
for col in tqdm(cat_col):
    filt_df[col] = filt_df[col].replace(filt_df[col].unique(),list(range(len(filt_df[col].unique()))))
```

```
100%|██████████████████████████████████████████████████████████████████████████████████| 11/11 [00:18<00:00,  1.6
4s/it]
```

In [55]:

```
filt_df.shape
```

Out[55]:

```
(1348059, 79)
```

In [56]:

```
filt_df.loan_status.head(50)
```

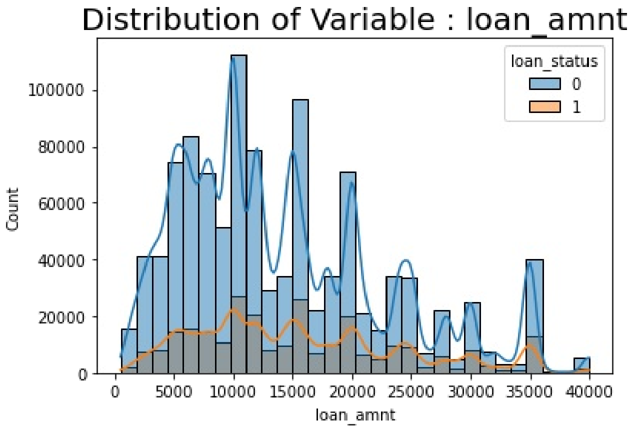
Out[56]:

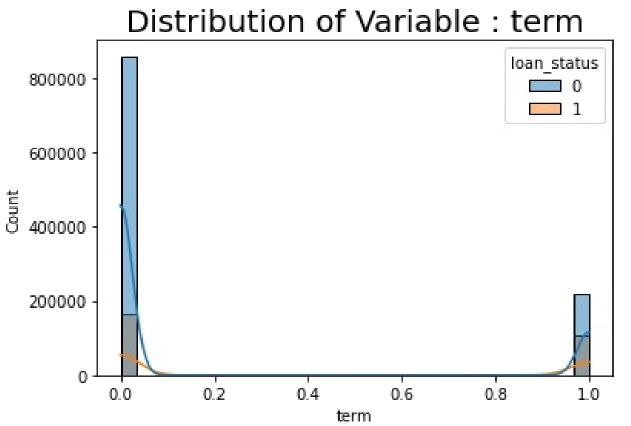
```
0     0
1     0
2     0
4     0
5     0
6     0
7     0
8     0
9     0
12    0
13    1
14    0
15    0
16    0
17    0
19    0
20    0
21    0
22    0
23    0
24    0
25    1
26    0
27    0
28    0
29    0
30    1
31    1
32    0
33    1
35    0
36    0
37    0
38    0
39    0
40    0
41    1
43    0
44    0
45    0
46    0
47    0
49    0
50    0
54    0
56    0
57    0
58    0
59    0
60    1
Name: loan_status, dtype: int64
```

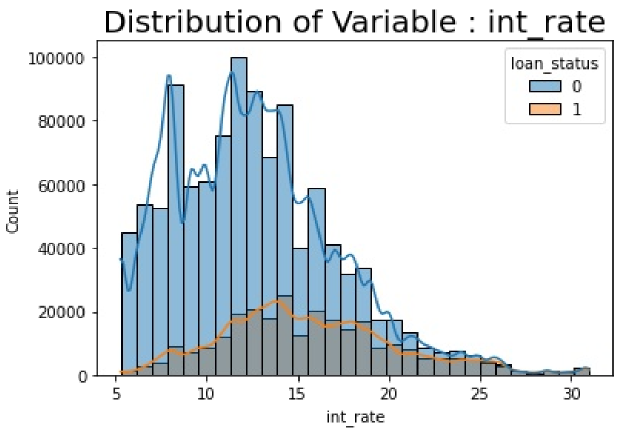
Plotting the columns to understand the distribution of the variables

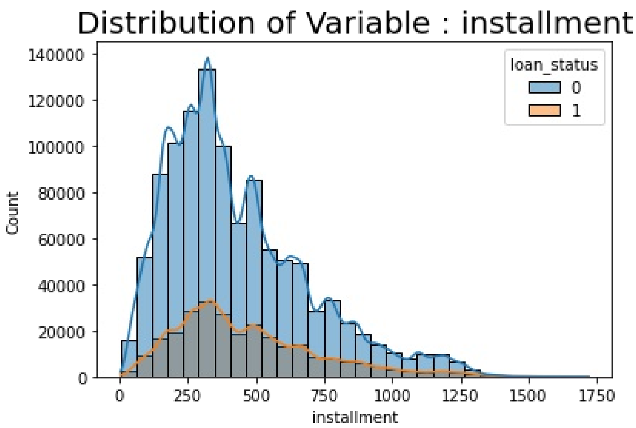
In [57]:

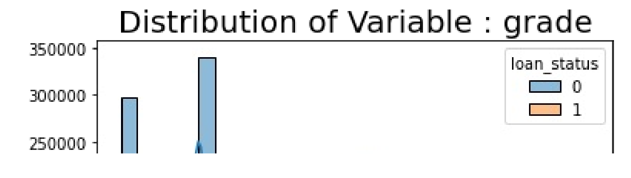
```
for col in ['loan_amnt',  'term', 'int_rate','installment', 'grade', 'sub_grade','purpose']:
    #fig = plt.figure(figsize=(17, 6))

    sns.histplot(data=filt_df, x=col, bins=30,kde=True, hue="loan_status")
    plt.title('Distribution of Variable : '+ col, fontsize = 20)
    plt.show();
# plt.show();
```

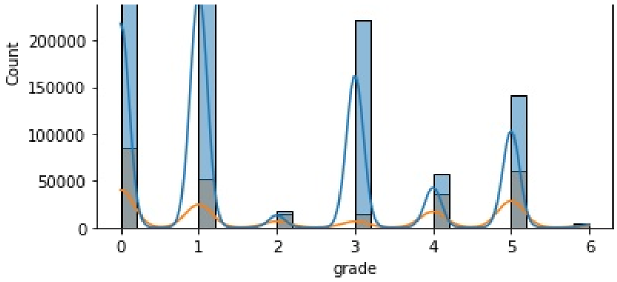




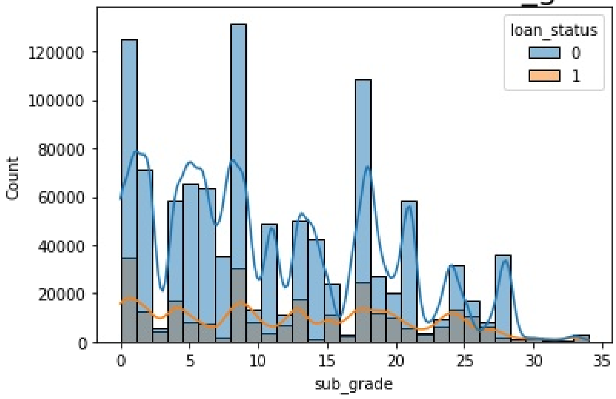




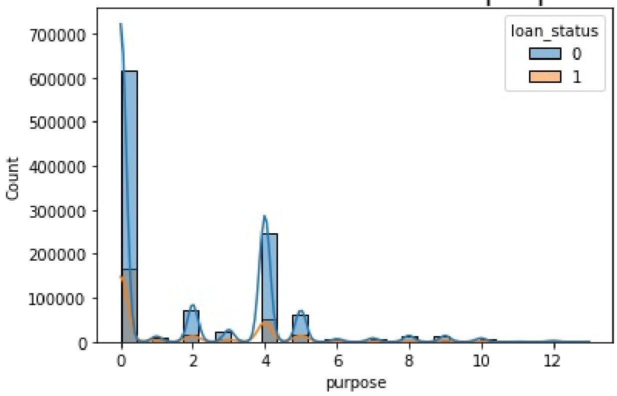




Distribution of Variable : sub_grade



Distribution of Variable : purpose



Shapiro-Wilk test to check the normality

In [58]:

```
from scipy.stats import shapiro
from scipy import stats
import pandas as pd
from datetime import datetime
```

In [59]:

```
filt_df.shape
```

Out[59]:

```
(1348059, 79)
```

In [60]:

```
for col in num_col:
    shapiro_test=stats.shapiro(filt_df[col])
    print(shapiro_test)
```

```
ShapiroResult(statistic=0.9379510283470154, pvalue=0.0)
ShapiroResult(statistic=0.9381945729255676, pvalue=0.0)
ShapiroResult(statistic=0.9399996399879456, pvalue=0.0)
ShapiroResult(statistic=0.9670817255973816, pvalue=0.0)
ShapiroResult(statistic=0.9328425526618958, pvalue=0.0)
ShapiroResult(statistic=0.40756088495254517, pvalue=0.0)
ShapiroResult(statistic=0.641416311264038l, pvalue=0.0)
ShapiroResult(statistic=0.4034233093261719, pvalue=0.0)
ShapiroResult(statistic=0.8876957297325134, pvalue=0.0)
```

```
ShapiroResult(statistic=0.8895211219787598, pvalue=0.0)
ShapiroResult(statistic=0.7085381150245667, pvalue=0.0)
ShapiroResult(statistic=0.9247516989707947, pvalue=0.0)
ShapiroResult(statistic=0.3787853717803955, pvalue=0.0)
ShapiroResult(statistic=0.47629356384277344, pvalue=0.0)
ShapiroResult(statistic=0.9837788343429565, pvalue=0.0)
ShapiroResult(statistic=0.9526931643486023, pvalue=0.0)
ShapiroResult(statistic=1.0, pvalue=1.0)
ShapiroResult(statistic=1.0, pvalue=1.0)
ShapiroResult(statistic=0.9197896718978882, pvalue=0.0)
ShapiroResult(statistic=0.9198870062828064, pvalue=0.0)
ShapiroResult(statistic=0.9239315986633301, pvalue=0.0)
ShapiroResult(statistic=0.7427215576171875, pvalue=0.0)
ShapiroResult(statistic=0.13025116920471191, pvalue=0.0)
ShapiroResult(statistic=0.2718522548675537, pvalue=0.0)
ShapiroResult(statistic=0.25459951162338257, pvalue=0.0)
ShapiroResult(statistic=0.9579343199729919, pvalue=0.0)
ShapiroResult(statistic=0.6874939203262329, pvalue=0.0)
ShapiroResult(statistic=0.09160852432250977, pvalue=0.0)
ShapiroResult(statistic=1.0, pvalue=1.0)
ShapiroResult(statistic=0.03776836395263672, pvalue=0.0)
ShapiroResult(statistic=0.0035368800163269043, pvalue=0.0)
ShapiroResult(statistic=0.7712674736976624, pvalue=0.0)
ShapiroResult(statistic=0.5361955165863037, pvalue=0.0)
ShapiroResult(statistic=0.9149291515350342, pvalue=0.0)
ShapiroResult(statistic=0.7015171051025391, pvalue=0.0)
ShapiroResult(statistic=0.6297277212142944, pvalue=0.0)
ShapiroResult(statistic=0.958914041519165, pvalue=0.0)
ShapiroResult(statistic=0.05478096008300781, pvalue=0.0)
ShapiroResult(statistic=0.004641056060791016, pvalue=0.0)
ShapiroResult(statistic=0.9578121304512024, pvalue=0.0)
ShapiroResult(statistic=0.9424670934677124, pvalue=0.0)
ShapiroResult(statistic=0.6628600358963013, pvalue=0.0)
ShapiroResult(statistic=0.6579204797744751, pvalue=0.0)
ShapiroResult(statistic=0.8093883395195007, pvalue=0.0)
ShapiroResult(statistic=0.665735125541687, pvalue=0.0)
ShapiroResult(statistic=0.9096857905387878, pvalue=0.0)
ShapiroResult(statistic=0.44327741861343384, pvalue=0.0)
ShapiroResult(statistic=0.8975507616996765, pvalue=0.0)
ShapiroResult(statistic=0.8960646390914917, pvalue=0.0)
ShapiroResult(statistic=0.8818497657775879, pvalue=0.0)
ShapiroResult(statistic=0.9158438444137573, pvalue=0.0)
ShapiroResult(statistic=0.8354514837265015, pvalue=0.0)
ShapiroResult(statistic=0.9089000821113586, pvalue=0.0)
ShapiroResult(statistic=0.9174714684486389, pvalue=0.0)
ShapiroResult(statistic=0.9093953371047974, pvalue=0.0)
ShapiroResult(statistic=0.9279497861862183, pvalue=0.0)
ShapiroResult(statistic=0.0089341402053833, pvalue=0.0)
ShapiroResult(statistic=0.029294073581695557, pvalue=0.0)
ShapiroResult(statistic=0.17028260231018066, pvalue=0.0)
ShapiroResult(statistic=0.879609227180481, pvalue=0.0)
ShapiroResult(statistic=0.7059807777404785, pvalue=0.0)
ShapiroResult(statistic=0.9087631702423096, pvalue=0.0)
ShapiroResult(statistic=0.3844577670097351, pvalue=0.0)
ShapiroResult(statistic=0.10792392492294312, pvalue=0.0)
ShapiroResult(statistic=0.7680385112762451, pvalue=0.0)
ShapiroResult(statistic=0.7205054759979248, pvalue=0.0)
ShapiroResult(statistic=0.7633920907974243, pvalue=0.0)
ShapiroResult(statistic=0.7770864367485046, pvalue=0.0)
```

Inference on Shapiro-Wilk test to check the normality:

As the p-value is less than 0.05, then the null hypothesis that the data are normally distributed is rejected. If the p-value is greater than 0.05, then the null hypothesis is not rejected

Define X and y (independent and dependent features for model creation)

In [61]:
```
X = filt_df.drop('loan_status',axis=1) #other than loan status
```

In [62]:
```
X.columns
```

Out[62]:
```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'term', 'int_rate',
       'installment', 'grade', 'sub_grade', 'annual_inc', 'pymnt_plan',
       'purpose', 'dti', 'delinq_2yrs', 'fico_range_low', 'fico_range_high',
       'inq_last_6mths', 'open_acc', 'pub_rec', 'revol_bal', 'revol_util',
       'total_acc', 'initial_list_status', 'out_prncp', 'out_prncp_inv',
       'total_pymnt', 'total_pymnt_inv', 'total_rec_prncp', 'total_rec_int',
```

```
       'total_rec_late_fee', 'recoveries', 'collection_recovery_fee',
       'last_fico_range_high', 'last_fico_range_low',
       'collections_12_mths_ex_med', 'policy_code', 'application_type',
       'acc_now_delinq', 'tot_coll_amt', 'tot_cur_bal', 'total_rev_hi_lim',
       'acc_open_past_24mths', 'avg_cur_bal', 'bc_open_to_buy', 'bc_util',
       'chargeoff_within_12_mths', 'delinq_amnt', 'mo_sin_old_il_acct',
       'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op', 'mo_sin_rcnt_tl',
       'mort_acc', 'mths_since_recent_bc', 'mths_since_recent_inq',
       'num_accts_ever_120_pd', 'num_actv_bc_tl', 'num_actv_rev_tl',
       'num_bc_sats', 'num_bc_tl', 'num_il_tl', 'num_op_rev_tl',
       'num_rev_accts', 'num_rev_tl_bal_gt_0', 'num_sats', 'num_tl_120dpd_2m',
       'num_tl_30dpd', 'num_tl_90g_dpd_24m', 'num_tl_op_past_12m',
       'pct_tl_nvr_dlq', 'percent_bc_gt_75', 'pub_rec_bankruptcies',
       'tax_liens', 'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit',
       'total_il_high_credit_limit', 'hardship_flag', 'disbursement_method',
       'debt_settlement_flag'],
      dtype='object')
```

In [63]:

```
y=filt_df['loan_status']
```

In [64]:

```
finalDF = filt_df.copy()
```

Model Selection

Training and testing the model

In [65]:

```
# training and test set splitting

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# get x and y
# x = df_copy.drop(columns='loan_status',axis=1)
# y = df_copy['loan_status']

# feature scaling to bring the features into same range
scaler = StandardScaler()
X = scaler.fit_transform(X)

# split the data. 70% for training and 30% for testing
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.30, shuffle = True)
```

Logistic Regression

In [66]:

```
# build the model
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(random_state= 42)

# fit the model on training data
lr_model.fit(X_train,y_train)

# make prediction on test data
y_pred = lr_model.predict(X_test)
```

In [67]:

```
out = pd.DataFrame({'Y_True':y_test,'y_Pred':y_pred})
```

In [68]:

```
# Create classification report
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
```

In [69]:

```
confusion_matrix(y_test,y_pred)
```

Out[69]:

```
array([[323592,     43],
       [   457,  80326]], dtype=int64)
```

In [70]:

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    323635
           1       1.00      0.99      1.00     80783

    accuracy                           1.00    404418
   macro avg       1.00      1.00      1.00    404418
weighted avg       1.00      1.00      1.00    404418
```

In [71]:
```
# Results
f1_results = []
```

In [72]:
```
import sklearn
```

In [73]:
```
sklearn.__version__
```

Out[73]:
```
'1.1.1'
```

In [74]:
```
# Store off results
f1_results.append({
    'Name':'Logistic Reg',
    'f1-pos':sklearn.metrics.f1_score(y_test, y_pred),
    'f1-neg':sklearn.metrics.f1_score(1-y_test, 1-y_pred)})
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    323635
           1       1.00      0.99      1.00     80783

    accuracy                           1.00    404418
   macro avg       1.00      1.00      1.00    404418
weighted avg       1.00      1.00      1.00    404418
```

PCA + Logistic Regression

In [75]:
```
# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(n_components = 10)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
```

In [76]:
```
X_train_pca.shape
```

Out[76]:
```
(943641, 10)
```

In [77]:
```
X_test_pca.shape
```

Out[77]:
```
(404418, 10)
```

In [78]:
```
y_train.shape
```

Out[78]:
```
(943641,)
```

In [79]:
```
# build the model
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(random_state= 42)

# fit the model on training data
lr_model.fit(X_train_pca,y_train)
```

```
# make prediction on test data
y_lr_pred = lr_model.predict(X_test_pca)
```

In [80]:

```
# Store off results
f1_results.append({
    'Name':'PCA+Log Reg',
    'f1-pos':sklearn.metrics.f1_score(y_test, y_lr_pred),
    'f1-neg':sklearn.metrics.f1_score(1-y_test, 1-y_lr_pred)})
print(classification_report(y_test,y_lr_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.98      0.97    323635
           1       0.93      0.86      0.89     80783

    accuracy                           0.96    404418
   macro avg       0.95      0.92      0.93    404418
weighted avg       0.96      0.96      0.96    404418
```

LDA

In [81]:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda_model = LinearDiscriminantAnalysis()
lda_model.fit(X_train, y_train)
y_pred = lda_model.predict(X_test)
confusion_matrix(y_test,y_pred)
```

Out[81]:

```
array([[322639,    996],
       [ 12443,  68340]], dtype=int64)
```

In [82]:

```
# Store off results
f1_results.append({
    'Name':'LDA',
    'f1-pos':sklearn.metrics.f1_score(y_test, y_pred),
    'f1-neg':sklearn.metrics.f1_score(1-y_test, 1-y_pred)})
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98    323635
           1       0.99      0.85      0.91     80783

    accuracy                           0.97    404418
   macro avg       0.97      0.92      0.95    404418
weighted avg       0.97      0.97      0.97    404418
```

RESAMPLING Methods

Over/Undersampling

Apply oversampling and undersampling and train a logistic regression model to observe impact of balancing the samples.

Synthetic Minority Oversampling Technique (SMOTE) is a statistical technique for increasing the number of cases in your dataset in a balanced way

Oversampling

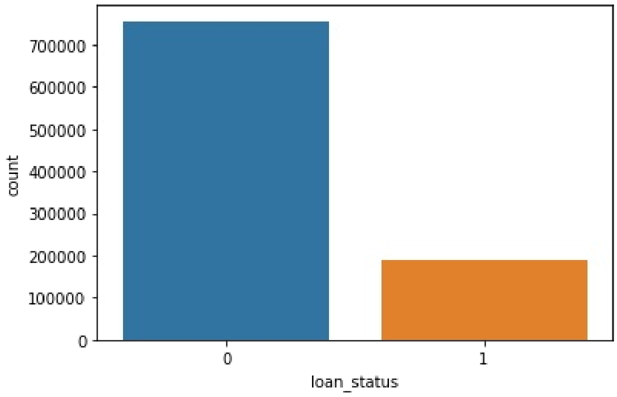
In [83]:

```
from imblearn.over_sampling import SMOTE
smt = SMOTE()
# X_train, y_train = smt.fit_resample(X_train, y_train)

# smt = SMOTE(sampling_strategy=0.2)
X_smote, y_smote = smt.fit_resample(X_train, y_train)
```

In [84]:

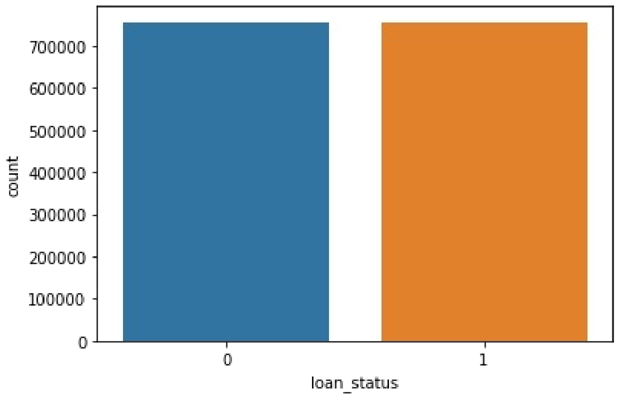
```
sns.countplot(y_train)
```

Out[84]: <AxesSubplot:xlabel='loan_status', ylabel='count'>



In [85]:
```
sns.countplot(y_smote)
```

Out[85]: <AxesSubplot:xlabel='loan_status', ylabel='count'>



In [86]:
```
# build the model
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(random_state= 42)

# fit the model on training data
lr_model.fit(X_smote,y_smote)

# make prediction on test data
y_pred_smote = lr_model.predict(X_test)
```

In [87]:
```
X_smote.shape
```

Out[87]:
```
(1510208, 78)
```

In [88]:
```
y_smote.shape
```

Out[88]:
```
(1510208,)
```

In [89]:
```
confusion_matrix(y_test,y_pred_smote)
```

Out[89]:
```
array([[323581,     54],
       [   381,  80402]], dtype=int64)
```

In [90]:
```
f1_results.append({
    'Name':'Oversample-LR',
    'f1-pos':sklearn.metrics.f1_score(y_test, y_pred_smote),
    'f1-neg':sklearn.metrics.f1_score(1-y_test, 1-y_pred_smote)})
print(classification_report(y_test,y_pred_smote))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    323635
           1       1.00      1.00      1.00     80783

    accuracy                           1.00    404418
   macro avg       1.00      1.00      1.00    404418
weighted avg       1.00      1.00      1.00    404418
```

Undersampling

In [91]:
```
from imblearn.under_sampling import RandomUnderSampler
undersampler = RandomUnderSampler()
X_under, y_under = undersampler.fit_resample(X_train, y_train)
```

In [92]:
```
X_train.shape
```

Out[92]:
```
(943641, 78)
```

In [93]:
```
X_under.shape
```

Out[93]:
```
(377074, 78)
```

In [94]:
```
# build the model
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(random_state= 42)

# fit the model on training data
lr_model.fit(X_under, y_under)

# make prediction on test data
y_pred_under = lr_model.predict(X_test)
```

In [95]:
```
confusion_matrix(y_test,y_pred_under)
```

Out[95]:
```
array([[323535,    100],
       [   371,  80412]], dtype=int64)
```

In [96]:
```
f1_results.append({
    'Name':'Undersample-LR',
    'f1-pos':sklearn.metrics.f1_score(y_test, y_pred_under),
    'f1-neg':sklearn.metrics.f1_score(1-y_test, 1-y_pred_under)})
print(classification_report(y_test,y_pred_under))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    323635
           1       1.00      1.00      1.00     80783

    accuracy                           1.00    404418
   macro avg       1.00      1.00      1.00    404418
weighted avg       1.00      1.00      1.00    404418
```

Results

In [97]:
```
# Look at cumulative results
# Pos -> Loan Paid completely
# Neg -> Load defaulted
pd.DataFrame(f1_results)
```

Out[97]:

	Name	f1-pos	f1-neg

0	Logistic Reg	0.996897	0.999228
1	PCA+Log Reg	0.892676	0.974444
2	LDA	0.910478	0.979598
3	Oversample-LR	0.997302	0.999328
4	Undersample-LR	0.997080	0.999273

Statistical Analysis

Hypothesis Testing

Here I can group the loans to "fully paid" & "charged-off", and then use hypothesis tests to compare the two distributions of each feature.

If the test statistic is small or the p-value is high (>0.05, 95% confidence level), we cannot reject the null hypothesis that the distributions of the two samples are the same and if p<0.05, different distributions.

** chi-squared Tests:**

- Numerical features: We can use K-S tests
- Features with only 0 or 1 values, we can use proportion Z tests to check whether the difference in mean values is statistically significant.
- For categorical features, we can use chi-squared Tests

In [98]:

```
filt_df.head()
```

Out[98]:

	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	annual_inc	loan_status	pymnt_plan	purpose	
0	3600.0	3600.0	3600.0	0	13.99	123.03	0	0	55000.0	0	0	0	5.
1	24700.0	24700.0	24700.0	0	11.99	820.28	0	1	65000.0	0	0	1	16.
2	20000.0	20000.0	20000.0	1	10.78	432.66	1	2	63000.0	0	0	2	10.
4	10400.0	10400.0	10400.0	1	22.45	289.91	2	3	104433.0	0	0	3	25.
5	11950.0	11950.0	11950.0	0	13.44	405.18	0	4	34000.0	0	0	0	10.

Define helper function

In [99]:

```
from scipy.stats import ks_2samp
from treeinterpreter import treeinterpreter as ti
from statsmodels.stats.proportion import proportions_ztest
from scipy.stats import chi2_contingency

def run_proportion_Z_test(feature):
    dist1 = df.loc[df.loan_status == 0, feature]
    dist2 = df.loc[df.loan_status == 1, feature]
    n1 = len(dist1)
    p1 = dist1.sum()
    n2 = len(dist2)
    p2 = dist2.sum()
    z_score, p_value = proportions_ztest([p1, p2], [n1, n2])
    print(feature+':')
    print('z-score = {}; p-value = {}'.format(z_score, p_value),'\n')


def run_chi2_test(df, feature):

    dist1 = df.loc[df.loan_status == 0,feature].value_counts().sort_index().tolist()
    dist2 = df.loc[df.loan_status == 1,feature].value_counts().sort_index().tolist()
    chi2, p, dof, expctd = chi2_contingency([dist1,dist2])
    print(feature+':')
    print("chi-square test statistic:", chi2)
    print("p-value", p, '\n')
```

p-value > 0.05 - same distribution **We accept null hypothesis**

p-value <0.05, different distributions **We reject null hypothesis and accept alternate hypothesis**

Null Hypothesis: There is no significant difference between the mean loan amount of paid and charged off loans.

Alternate Hypothesis: There is a significant difference between the mean loan amount of paid and charged off loans.

In [100…

```
df = filt_df.copy()
list_float = filt_df.select_dtypes(exclude=['object']).columns
list_object = ['term', 'grade', 'sub_grade', 'pymnt_plan', 'purpose']
```

In [101…

```
for col in list_object:
    finalDF[col] = finalDF[col].astype('object')
```

Chi-square test

In [102…

```
for i in list_object:
    run_chi2_test(finalDF, i)
```

```
term:
chi-square test statistic: 41635.37988090927
p-value 0.0

grade:
chi-square test statistic: 92310.08444669266
p-value 0.0

sub_grade:
chi-square test statistic: 96540.69702173014
p-value 0.0

pymnt_plan:
chi-square test statistic: 0.0
p-value 1.0

purpose:
chi-square test statistic: 4182.803111844084
p-value 0.0
```

Insights:

If we look at the categorical variables, p-value is lesser than 0.05 for **Term**, **Grade**, **Sub-Grade** and **Purpose**.

We reject null hypothesis and accept alternate hypothesis. So, there is a significant difference between the groups of paid and charged off loans.

For Payment plans, there is no differnce between the two groups, since the p-value is greater than 0.05. We accept the Null Hypothesis.

2 Sample Z test (Column wise)

- we will implement 2 sample z-test where one variable will be categorical with two categories and the other variable will be continuous to apply the z-test.

Null Hypothesis: There is no significant difference between the mean loan amount of paid and charged off loans.

Alternate Hypothesis: There is a significant difference between the mean loan amount of paid and charged off loans.

NOTE - we can also use t-test but it is used to compare the mean of two given samples like the Z-test. However, It is implemented when the sample size is less than 30. It assumes a normal distribution of the sample. It can also be one-sample or two-sample.

- **Loan Status** - '0' - Fully Paid
- **Loan Status** - '1' - Charged Off

Loan Amount

In [103…

```
from numpy import sqrt
from scipy.stats import norm
full_paid_mean=df.loc[df['loan_status']==0,'loan_amnt'].mean()
charged_off_mean=df.loc[df['loan_status']==1,'loan_amnt'].mean()
full_paid_std=df.loc[df['loan_status']==0,'loan_amnt'].std()
charged_off_std=df.loc[df['loan_status']==1,'loan_amnt'].std()
no_of_full_paid=df.loc[df['loan_status']==0,'loan_amnt'].count()
```

```
no_of_charged_off=df.loc[df['loan_status']==1,'loan_amnt'].count()

def twoSampZ(X1, X2, mudiff, sd1, sd2, n1, n2):
    pooledSE = sqrt(sd1**2/n1 + sd2**2/n2)
    z = ((X1 - X2) - mudiff)/pooledSE
    pval = 2*(1 - norm.cdf(abs(z)))
    return round(z,3), pval
z,p= twoSampZ(full_paid_mean,charged_off_mean,0,full_paid_std,charged_off_std,no_of_full_paid,no_of_charged_off)
print('Z', z)
print('p', p)

if p<0.05:
    print("We reject null hypothesis and accept Alternate Hypothesis")
    print('Alternate Hypothesis: There is a significant difference between the mean loan amount of paid and charg
else:
    print("We accept null hypothesis")
```

```
Z -75.212
p 0.0
We reject null hypothesis and accept Alternate Hypothesis
Alternate Hypothesis: There is a significant difference between the mean loan amount of paid and charged off loan
s.
```

Interest Rate

In [104…

```
from numpy import sqrt
from scipy.stats import norm

full_paid_mean=df.loc[df['loan_status']==0,'int_rate'].mean()
charged_off_mean=df.loc[df['loan_status']==1,'int_rate'].mean()
full_paid_std=df.loc[df['loan_status']==0,'int_rate'].std()
charged_off_std=df.loc[df['loan_status']==1,'int_rate'].std()
no_of_full_paid=df.loc[df['loan_status']==0,'int_rate'].count()
no_of_charged_off=df.loc[df['loan_status']==1,'int_rate'].count()

def twoSampZ(X1, X2, mudiff, sd1, sd2, n1, n2):
    pooledSE = sqrt(sd1**2/n1 + sd2**2/n2)
    z = ((X1 - X2) - mudiff)/pooledSE
    pval = 2*(1 - norm.cdf(abs(z)))
    return round(z,3), pval
z,p= twoSampZ(full_paid_mean,charged_off_mean,0,full_paid_std,charged_off_std,no_of_full_paid,no_of_charged_off)
print('Z', z)
print('p', p)

if p<0.05:
    print("We reject null hypothesis and accept Alternate Hypothesis")
    print('Alternate Hypothesis: There is a significant difference between the mean Interest Rate of paid and cha
else:
    print("We accept null hypothesis")
```

```
Z -296.074
p 0.0
We reject null hypothesis and accept Alternate Hypothesis
Alternate Hypothesis: There is a significant difference between the mean Interest Rate of paid and charged off lo
ans.
```

Installment

In [105…

```
from numpy import sqrt
from scipy.stats import norm

full_paid_mean=df.loc[df['loan_status']==0,'installment'].mean()
charged_off_mean=df.loc[df['loan_status']==1,'installment'].mean()
full_paid_std=df.loc[df['loan_status']==0,'installment'].std()
charged_off_std=df.loc[df['loan_status']==1,'installment'].std()
no_of_full_paid=df.loc[df['loan_status']==0,'installment'].count()
no_of_charged_off=df.loc[df['loan_status']==1,'installment'].count()

def twoSampZ(X1, X2, mudiff, sd1, sd2, n1, n2):
    pooledSE = sqrt(sd1**2/n1 + sd2**2/n2)
    z = ((X1 - X2) - mudiff)/pooledSE
    pval = 2*(1 - norm.cdf(abs(z)))
    return round(z,3), pval
z,p= twoSampZ(full_paid_mean,charged_off_mean,0,full_paid_std,charged_off_std,no_of_full_paid,no_of_charged_off)
print('Z', z)
print('p', p)
```

```
if p<0.05:
    print("We reject null hypothesis and accept Alternate Hypothesis")
    print('Alternate Hypothesis: There is a significant difference between the mean +66+66 of paid and charged of
else:
    print("We accept null hypothesis")
```

```
Z -59.518
p 0.0
We reject null hypothesis and accept Alternate Hypothesis
Alternate Hypothesis: There is a significant difference between the mean +66+66 of paid and charged off loans.
```

In the same way, we can do the same for any column.

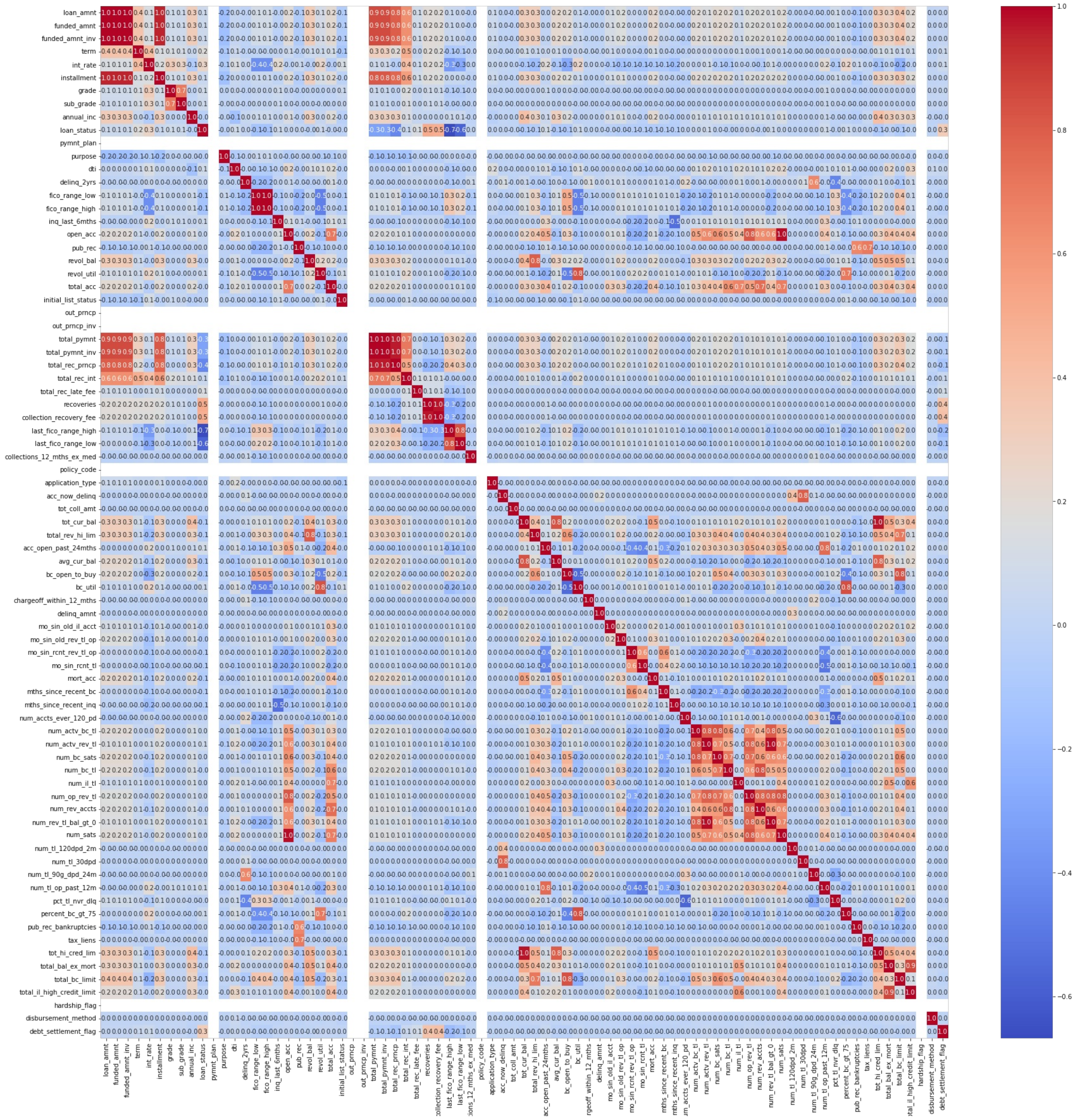
Correlation Test

- A correlation test is a metric to evaluate the extent to which variables are associated with one another.

Correlation Analysis

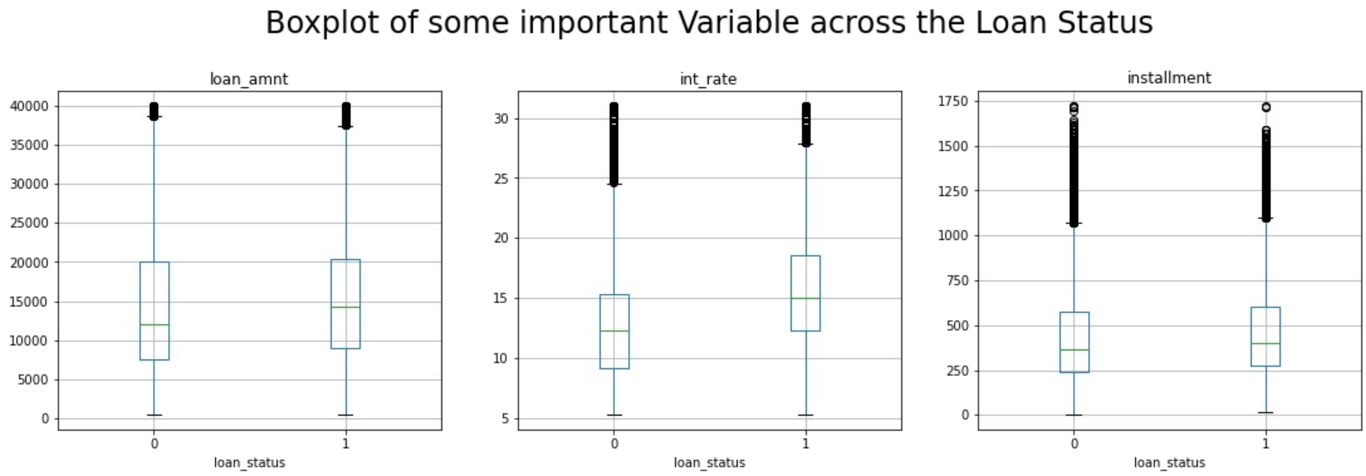
In [106…

```
fig, ax = plt.subplots(figsize=(30,30))          # Sample figsize in inches
cm_df = sns.heatmap(df[list_float].corr(),annot=True, fmt = ".1f", cmap = "coolwarm", ax=ax)
```



In [107…

```python
# Box plots for Categorical Target Variable "GoodCredit" and continuous predictors
ContinuousColsList=['loan_amnt', 'int_rate', 'installment']

import matplotlib.pyplot as plt
fig, PlotCanvas=plt.subplots(nrows=1, ncols=len(ContinuousColsList), figsize=(18,5))

# Creating box plots for each continuous predictor against the Target Variable "GoodCredit"
for PredictorCol , i in zip(ContinuousColsList, range(len(ContinuousColsList))):
    filt_df.boxplot(column=PredictorCol, by='loan_status', figsize=(5,5), vert=True, ax=PlotCanvas[i])
plt.suptitle('Boxplot of some important Variable across the Loan Status', fontsize =24, y=1.08)
plt.show();
```



Analysis of Variance

ANOVA

Statistical Feature Selection (Categorical Vs Continuous) using ANOVA test

ANOVA stands for Analysis of variance. As the name, suggests it uses variance as its parameter to compare multiple independent groups. ANOVA can be one-way ANOVA or two-way ANOVA. One-way ANOVA is applied when there are three or more independent groups of a variable.

Analysis of variance(ANOVA) is performed to check if there is any relationship between the given continuous and categorical variable

In [108…

```python
# Defining a function to find the statistical relationship with all the categorical variables
def FunctionAnova(inpData, TargetVariable, ContinuousPredictorList):
    from scipy.stats import f_oneway

    # Creating an empty list of final selected predictors
    SelectedPredictors=[]

    print('##### ANOVA Results ##### \n')
    for predictor in ContinuousPredictorList:
        CategoryGroupLists=inpData.groupby(TargetVariable)[predictor].apply(list)
        AnovaResults = f_oneway(*CategoryGroupLists)
        # If the ANOVA P-Value is <0.05, that means we reject H0
        if (AnovaResults[1] < 0.05):
            print(predictor, 'is correlated with', TargetVariable, '| P-Value:', AnovaResults[1])
            SelectedPredictors.append(predictor)
        else:
            print(predictor, 'is NOT correlated with', TargetVariable, '| P-Value:', AnovaResults[1])

    return(SelectedPredictors)
# Calling the function to check which categorical variables are correlated with target
ContinuousVariables=list_float

FunctionAnova(inpData=filt_df, TargetVariable='loan_status', ContinuousPredictorList=ContinuousVariables)
```

```
##### ANOVA Results #####

loan_amnt is correlated with loan_status | P-Value: 0.0
funded_amnt is correlated with loan_status | P-Value: 0.0
funded_amnt_inv is correlated with loan_status | P-Value: 0.0
term is correlated with loan_status | P-Value: 0.0
int_rate is correlated with loan_status | P-Value: 0.0
```

```
installment is correlated with loan_status | P-Value: 0.0
grade is correlated with loan_status | P-Value: 0.0
sub_grade is correlated with loan_status | P-Value: 0.0
annual_inc is correlated with loan_status | P-Value: 0.0
loan_status is correlated with loan_status | P-Value: 0.0
pymnt_plan is NOT correlated with loan_status | P-Value: nan
purpose is correlated with loan_status | P-Value: 2.3677742696141588e-247
dti is correlated with loan_status | P-Value: 0.0
delinq_2yrs is correlated with loan_status | P-Value: 7.333434290443957e-111
fico_range_low is correlated with loan_status | P-Value: 0.0
fico_range_high is correlated with loan_status | P-Value: 0.0
inq_last_6mths is correlated with loan_status | P-Value: 0.0
open_acc is correlated with loan_status | P-Value: 5.954339248117238e-230
pub_rec is correlated with loan_status | P-Value: 3.4751252101445443e-202
revol_bal is correlated with loan_status | P-Value: 4.641607497040615e-115
revol_util is correlated with loan_status | P-Value: 0.0
total_acc is correlated with loan_status | P-Value: 5.6562557275083236e-40
initial_list_status is correlated with loan_status | P-Value: 2.9320675769544683e-15
out_prncp is NOT correlated with loan_status | P-Value: nan
out_prncp_inv is NOT correlated with loan_status | P-Value: nan
total_pymnt is correlated with loan_status | P-Value: 0.0
total_pymnt_inv is correlated with loan_status | P-Value: 0.0
total_rec_prncp is correlated with loan_status | P-Value: 0.0
total_rec_int is correlated with loan_status | P-Value: 0.0
total_rec_late_fee is correlated with loan_status | P-Value: 0.0
recoveries is correlated with loan_status | P-Value: 0.0
collection_recovery_fee is correlated with loan_status | P-Value: 0.0
last_fico_range_high is correlated with loan_status | P-Value: 0.0
last_fico_range_low is correlated with loan_status | P-Value: 0.0
collections_12_mths_ex_med is correlated with loan_status | P-Value: 9.653828422612811e-77
policy_code is NOT correlated with loan_status | P-Value: nan
application_type is correlated with loan_status | P-Value: 4.624806603712036e-78
acc_now_delinq is correlated with loan_status | P-Value: 5.966809447469724e-06
tot_coll_amt is NOT correlated with loan_status | P-Value: 0.6125087427451646
tot_cur_bal is correlated with loan_status | P-Value: 0.0
total_rev_hi_lim is correlated with loan_status | P-Value: 0.0
acc_open_past_24mths is correlated with loan_status | P-Value: 0.0
avg_cur_bal is correlated with loan_status | P-Value: 0.0
bc_open_to_buy is correlated with loan_status | P-Value: 0.0
bc_util is correlated with loan_status | P-Value: 0.0
chargeoff_within_12_mths is correlated with loan_status | P-Value: 0.0002681539466620759
delinq_amnt is correlated with loan_status | P-Value: 0.001217049218469726
mo_sin_old_il_acct is correlated with loan_status | P-Value: 1.2172146448012276e-188
mo_sin_old_rev_tl_op is correlated with loan_status | P-Value: 0.0
mo_sin_rcnt_rev_tl_op is correlated with loan_status | P-Value: 0.0
mo_sin_rcnt_tl is correlated with loan_status | P-Value: 0.0
mort_acc is correlated with loan_status | P-Value: 0.0
mths_since_recent_bc is correlated with loan_status | P-Value: 0.0
mths_since_recent_inq is correlated with loan_status | P-Value: 0.0
num_accts_ever_120_pd is correlated with loan_status | P-Value: 4.4360135851500946e-32
num_actv_bc_tl is correlated with loan_status | P-Value: 0.0
num_actv_rev_tl is correlated with loan_status | P-Value: 0.0
num_bc_sats is correlated with loan_status | P-Value: 3.857080379888324e-59
num_bc_tl is correlated with loan_status | P-Value: 2.9371360102457876e-86
num_il_tl is correlated with loan_status | P-Value: 4.11863501249806e-13
num_op_rev_tl is correlated with loan_status | P-Value: 3.0824047396457046e-305
num_rev_accts is correlated with loan_status | P-Value: 1.2981982310251494e-08
num_rev_tl_bal_gt_0 is correlated with loan_status | P-Value: 0.0
num_sats is correlated with loan_status | P-Value: 3.301926817401082e-204
num_tl_120dpd_2m is NOT correlated with loan_status | P-Value: 0.15163010108095454
num_tl_30dpd is correlated with loan_status | P-Value: 0.003143579863255129
num_tl_90g_dpd_24m is correlated with loan_status | P-Value: 2.750743256966899e-29
num_tl_op_past_12m is correlated with loan_status | P-Value: 0.0
pct_tl_nvr_dlq is correlated with loan_status | P-Value: 4.042620980716689e-33
percent_bc_gt_75 is correlated with loan_status | P-Value: 0.0
pub_rec_bankruptcies is correlated with loan_status | P-Value: 3.6307762350173283e-190
tax_liens is correlated with loan_status | P-Value: 2.5131626510889307e-29
tot_hi_cred_lim is correlated with loan_status | P-Value: 0.0
total_bal_ex_mort is correlated with loan_status | P-Value: 1.7000181361733167e-05
total_bc_limit is correlated with loan_status | P-Value: 0.0
total_il_high_credit_limit is NOT correlated with loan_status | P-Value: 0.14919565549806574
hardship_flag is NOT correlated with loan_status | P-Value: nan
disbursement_method is NOT correlated with loan_status | P-Value: 0.8445984342202176
debt_settlement_flag is correlated with loan_status | P-Value: 0.0
```

Out[108…

```
['loan_amnt',
 'funded_amnt',
 'funded_amnt_inv',
 'term',
 'int_rate',
 'installment',
 'grade',
 'sub_grade',
 'annual_inc',
 'loan_status',
 'purpose',
 'dti',
 'delinq_2yrs',
 'fico_range_low',
```

```
 'fico_range_high',
 'inq_last_6mths',
 'open_acc',
 'pub_rec',
 'revol_bal',
 'revol_util',
 'total_acc',
 'initial_list_status',
 'total_pymnt',
 'total_pymnt_inv',
 'total_rec_prncp',
 'total_rec_int',
 'total_rec_late_fee',
 'recoveries',
 'collection_recovery_fee',
 'last_fico_range_high',
 'last_fico_range_low',
 'collections_12_mths_ex_med',
 'application_type',
 'acc_now_delinq',
 'tot_cur_bal',
 'total_rev_hi_lim',
 'acc_open_past_24mths',
 'avg_cur_bal',
 'bc_open_to_buy',
 'bc_util',
 'chargeoff_within_12_mths',
 'delinq_amnt',
 'mo_sin_old_il_acct',
 'mo_sin_old_rev_tl_op',
 'mo_sin_rcnt_rev_tl_op',
 'mo_sin_rcnt_tl',
 'mort_acc',
 'mths_since_recent_bc',
 'mths_since_recent_inq',
 'num_accts_ever_120_pd',
 'num_actv_bc_tl',
 'num_actv_rev_tl',
 'num_bc_sats',
 'num_bc_tl',
 'num_il_tl',
 'num_op_rev_tl',
 'num_rev_accts',
 'num_rev_tl_bal_gt_0',
 'num_sats',
 'num_tl_30dpd',
 'num_tl_90g_dpd_24m',
 'num_tl_op_past_12m',
 'pct_tl_nvr_dlq',
 'percent_bc_gt_75',
 'pub_rec_bankruptcies',
 'tax_liens',
 'tot_hi_cred_lim',
 'total_bal_ex_mort',
 'total_bc_limit',
 'debt_settlement_flag']
```

In [109…

```python
import statsmodels.api as sm
from statsmodels.formula.api import ols

for x in list_float:
    model = ols('loan_status' + '~' + x, data = finalDF).fit() #Oridnary least square method
    result_anova = sm.stats.anova_lm(model) # ANOVA Test
    print(result_anova)
```

```
                  df         sum_sq     mean_sq            F  PR(>F)
loan_amnt        1.0     918.815156  918.815156  5771.860589     0.0
Residual   1348057.0  214595.481681    0.159189          NaN     NaN
                    df         sum_sq     mean_sq            F  PR(>F)
funded_amnt        1.0     919.716984  919.716984  5777.550017     0.0
Residual     1348057.0  214594.579853    0.159188          NaN     NaN
                        df         sum_sq     mean_sq            F  PR(>F)
funded_amnt_inv        1.0     905.893613  905.893613  5690.346736     0.0
Residual         1348057.0  214608.403224    0.159198          NaN     NaN
                 df         sum_sq      mean_sq             F  PR(>F)
term            1.0    6656.415709  6656.415709  42963.319087     0.0
Residual  1348057.0  208857.881129     0.154933           NaN     NaN
                 df         sum_sq       mean_sq             F  PR(>F)
int_rate        1.0   14408.157529  14408.157529  96580.928266     0.0
Residual  1348057.0  201106.139308      0.149182           NaN     NaN
                    df         sum_sq     mean_sq            F  PR(>F)
installment        1.0     570.151337  570.151337  3575.796399     0.0
Residual     1348057.0  214944.145500    0.159447          NaN     NaN
                 df         sum_sq      mean_sq             F  PR(>F)
```

```
grade           6.0   14757.620357  2459.603393  16515.880473     0.0
Residual  1348052.0  200756.676480     0.148924           NaN     NaN
                  df         sum_sq     mean_sq           F  PR(>F)
sub_grade       34.0   15433.968717  453.940256  3058.38343     0.0
Residual   1348024.0  200080.328120    0.148425         NaN     NaN
                   df         sum_sq     mean_sq            F  PR(>F)
annual_inc        1.0     375.272065  375.272065  2351.447556     0.0
Residual    1348057.0  215139.024772    0.159592          NaN     NaN
                    df        sum_sq       mean_sq            F  PR(>F)
loan_status        1.0  2.155143e+05  2.155143e+05  1.096280e+32     0.0
Residual     1348057.0  2.650105e-21  1.965870e-27           NaN     NaN
                   df         sum_sq  mean_sq   F  PR(>F)
pymnt_plan        0.0       0.000000      NaN NaN     NaN
Residual    1348058.0  215514.296837  0.15987 NaN     NaN
                 df         sum_sq    mean_sq          F  PR(>F)
purpose        13.0     668.705058  51.438851  322.75219     0.0
Residual  1348045.0  214845.591779   0.159376        NaN     NaN
                 df         sum_sq      mean_sq            F  PR(>F)
dti             1.0    1528.481033  1528.481033  9629.047365     0.0
Residual  1348057.0  213985.815804     0.158736          NaN     NaN
                    df         sum_sq    mean_sq           F         PR(>F)
delinq_2yrs        1.0      80.003115  80.003115  500.610916  7.333434e-111
Residual     1348057.0  215434.293722   0.159811         NaN            NaN
                       df         sum_sq      mean_sq           F  PR(>F)
fico_range_low        1.0    3682.947433  3682.947433  23437.62187     0.0
Residual        1348057.0  211831.349404     0.157138          NaN     NaN
                        df         sum_sq      mean_sq             F  PR(>F)
fico_range_high        1.0    3682.873837  3682.873837  23437.145379     0.0
Residual         1348057.0  211831.423000     0.157138           NaN     NaN
                       df         sum_sq     mean_sq           F  PR(>F)
inq_last_6mths        1.0     937.636594  937.636594  5890.610714     0.0
Residual        1348057.0  214576.660243    0.159175          NaN     NaN
                 df         sum_sq     mean_sq          F         PR(>F)
open_acc        1.0     167.513001  167.513001  1048.6206  5.954339e-230
Residual  1348057.0  215346.783836    0.159746        NaN            NaN
                 df         sum_sq     mean_sq           F         PR(>F)
pub_rec         1.0     147.106370  147.106370  920.789148  3.475125e-202
Residual  1348057.0  215367.190467    0.159761         NaN            NaN
                  df         sum_sq    mean_sq           F         PR(>F)
revol_bal        1.0      83.087088  83.087088  519.915991  4.641607e-115
Residual   1348057.0  215431.209749   0.159809         NaN            NaN
                   df         sum_sq     mean_sq            F  PR(>F)
revol_util        1.0     775.759093  775.759093  4869.957143     0.0
Residual    1348057.0  214738.537744    0.159295          NaN     NaN
                  df         sum_sq    mean_sq           F        PR(>F)
total_acc        1.0      27.993564  27.993564  175.124445  5.656256e-40
Residual   1348057.0  215486.303274   0.159850         NaN           NaN
                           df         sum_sq   mean_sq          F  \
initial_list_status        1.0       9.961534  9.961534  62.312971   
Residual             1348057.0  215504.335303  0.159863        NaN   

                           PR(>F)  
initial_list_status  2.932068e-15  
Residual                      NaN  
                  df         sum_sq   mean_sq         F    PR(>F)
out_prncp        1.0       0.039845  0.039845  0.249232  0.617616
Residual   1348058.0  215514.296837  0.159870       NaN       NaN
                      df         sum_sq   mean_sq         F    PR(>F)
out_prncp_inv        1.0       0.039845  0.039845  0.249232  0.617616
Residual       1348058.0  215514.296837  0.159870       NaN       NaN
                    df         sum_sq       mean_sq              F  PR(>F)
total_pymnt        1.0   21423.791451  21423.791451  148799.097484     0.0
Residual     1348057.0  194090.505386      0.143978            NaN     NaN
                        df         sum_sq       mean_sq              F  PR(>F)
total_pymnt_inv        1.0   21352.747324  21352.747324  148251.394636     0.0
Residual         1348057.0  194161.549513      0.144031            NaN     NaN
                        df         sum_sq       mean_sq              F  PR(>F)
total_rec_prncp        1.0   41517.214273  41517.214273  321658.101947     0.0
Residual         1348057.0  173997.082564      0.129072            NaN     NaN
                      df         sum_sq     mean_sq            F  PR(>F)
total_rec_int        1.0     659.310716  659.310716  4136.689785     0.0
Residual       1348057.0  214854.986121    0.159381          NaN     NaN
                         df         sum_sq      mean_sq            F  \
total_rec_late_fee        1.0    4282.371577  4282.371577  27329.585594   
Residual            1348057.0  211231.925260     0.156694           NaN   

                    PR(>F)  
total_rec_late_fee     0.0  
Residual               NaN  
                   df         sum_sq       mean_sq              F  PR(>F)
recoveries        1.0   54911.804218  54911.804218  460915.897698     0.0
Residual    1348057.0  160602.492620      0.119136            NaN     NaN
                              df         sum_sq       mean_sq  \
collection_recovery_fee        1.0   49469.666056  49469.666056   
Residual                 1348057.0  166044.630781      0.123173   

                                     F  PR(>F)  
collection_recovery_fee  401626.534394     0.0  
Residual                           NaN     NaN  
```

```
                            df         sum_sq       mean_sq             F  \
last_fico_range_high        1.0   95919.739542  95919.739542  1.081197e+06
Residual              1348057.0  119594.557295      0.088716           NaN

                      PR(>F)
last_fico_range_high     0.0
Residual                 NaN
                           df         sum_sq       mean_sq              F  \
last_fico_range_low        1.0   71267.753033  71267.753033  666033.242928
Residual             1348057.0  144246.543804      0.107003            NaN

                     PR(>F)
last_fico_range_low     0.0
Residual                NaN
                                  df         sum_sq    mean_sq           F  \
collections_12_mths_ex_med        1.0      54.951004  54.951004  343.810038
Residual                    1348057.0  215459.345833   0.159830         NaN

                                  PR(>F)
collections_12_mths_ex_med  9.653828e-77
Residual                             NaN
                    df         sum_sq   mean_sq         F    PR(>F)
policy_code        1.0       0.039913  0.039913  0.249662  0.617313
Residual     1348058.0  215514.296837  0.159870       NaN       NaN
                         df         sum_sq    mean_sq           F  \
application_type        1.0      55.919511  55.919511  349.871235
Residual          1348057.0  215458.377326   0.159829         NaN

                        PR(>F)
application_type  4.624807e-78
Residual                   NaN
                       df         sum_sq   mean_sq          F    PR(>F)
acc_now_delinq        1.0       3.277129  3.277129  20.498981  0.000006
Residual        1348057.0  215511.019708  0.159868        NaN       NaN
                     df         sum_sq   mean_sq         F    PR(>F)
tot_coll_amt        1.0       0.041013  0.041013  0.256538  0.612509
Residual      1348057.0  215514.255824  0.159870       NaN       NaN
                    df         sum_sq      mean_sq            F  PR(>F)
tot_cur_bal        1.0    1044.741735  1044.741735  6566.766124     0.0
Residual     1348057.0  214469.555102     0.159095          NaN     NaN
                         df         sum_sq     mean_sq            F  PR(>F)
total_rev_hi_lim        1.0     574.074271  574.074271  3600.465421     0.0
Residual          1348057.0  214940.222567    0.159444          NaN     NaN
                             df         sum_sq      mean_sq             F  \
acc_open_past_24mths        1.0    2083.727722  2083.727722  13161.112548
Residual              1348057.0  213430.569115     0.158325           NaN

                      PR(>F)
acc_open_past_24mths     0.0
Residual                 NaN
                    df         sum_sq      mean_sq           F  PR(>F)
avg_cur_bal        1.0    1283.447305  1283.447305  8076.14836     0.0
Residual     1348057.0  214230.849532     0.158918         NaN     NaN
                       df         sum_sq      mean_sq            F  PR(>F)
bc_open_to_buy        1.0    1390.673674  1390.673674  8755.257141     0.0
Residual        1348057.0  214123.623163     0.158839          NaN     NaN
                 df         sum_sq     mean_sq           F  PR(>F)
bc_util         1.0     903.529495  903.529495  5675.43407     0.0
Residual  1348057.0  214610.767342    0.159200         NaN     NaN
                                df         sum_sq   mean_sq          F  \
chargeoff_within_12_mths        1.0       2.123178  2.123178  13.280759
Residual                  1348057.0  215512.173659  0.159869        NaN

                            PR(>F)
chargeoff_within_12_mths  0.000268
Residual                       NaN
                    df         sum_sq   mean_sq          F    PR(>F)
delinq_amnt        1.0       1.672922  1.672922  10.464325  0.001217
Residual     1348057.0  215512.623915  0.159869        NaN       NaN
                           df         sum_sq     mean_sq           F  \
mo_sin_old_il_acct        1.0     137.152324  137.152324  858.443687
Residual            1348057.0  215377.144513    0.159769         NaN

                           PR(>F)
mo_sin_old_il_acct  1.217215e-188
Residual                      NaN
                            df         sum_sq     mean_sq            F  \
mo_sin_old_rev_tl_op        1.0     525.526879  525.526879  3295.242761
Residual              1348057.0  214988.769958    0.159480          NaN

                      PR(>F)
mo_sin_old_rev_tl_op     0.0
Residual                 NaN
                             df         sum_sq     mean_sq            F  \
mo_sin_rcnt_rev_tl_op        1.0     599.431543  599.431543  3759.944134
Residual               1348057.0  214914.865295    0.159426          NaN

                       PR(>F)
mo_sin_rcnt_rev_tl_op     0.0
```

```
Residual                   NaN
                       df        sum_sq     mean_sq           F  PR(>F)
mo_sin_rcnt_tl        1.0     626.349490  626.349490  3929.279538     0.0
Residual        1348057.0  214887.947347    0.159406          NaN     NaN
                 df        sum_sq      mean_sq           F  PR(>F)
mort_acc        1.0    1184.413459  1184.413459  7449.529805     0.0
Residual  1348057.0  214329.883378     0.158992          NaN     NaN
                             df        sum_sq     mean_sq           F  \
mths_since_recent_bc        1.0     546.605318  546.605318  3427.748234
Residual              1348057.0  214967.691519    0.159465          NaN

                      PR(>F)
mths_since_recent_bc     0.0
Residual                 NaN
                              df        sum_sq     mean_sq           F  \
mths_since_recent_inq        1.0     607.618996  607.618996  3811.445272
Residual               1348057.0  214906.677841    0.159420          NaN

                       PR(>F)
mths_since_recent_inq     0.0
Residual                  NaN
                              df        sum_sq    mean_sq          F  \
num_accts_ever_120_pd        1.0      22.218573  22.218573  138.993056
Residual               1348057.0  215492.078264   0.159854         NaN

                             PR(>F)
num_accts_ever_120_pd  4.436014e-32
Residual                        NaN
                       df        sum_sq     mean_sq           F  PR(>F)
num_actv_bc_tl        1.0     356.541400  356.541400  2233.887077     0.0
Residual        1348057.0  215157.755438    0.159606          NaN     NaN
                        df        sum_sq      mean_sq           F  PR(>F)
num_actv_rev_tl        1.0    1033.275151  1033.275151  6494.34523     0.0
Residual         1348057.0  214481.021686     0.159104         NaN     NaN
                    df        sum_sq    mean_sq          F        PR(>F)
num_bc_sats        1.0      42.037659  42.037659  262.999797  3.857080e-59
Residual     1348057.0  215472.259178   0.159839         NaN           NaN
                  df        sum_sq    mean_sq          F        PR(>F)
num_bc_tl        1.0      61.936617  61.936617  387.529242  2.937136e-86
Residual   1348057.0  215452.360220   0.159824         NaN           NaN
                  df        sum_sq   mean_sq          F        PR(>F)
num_il_tl        1.0       8.406742  8.406742  52.586811  4.118635e-13
Residual   1348057.0  215505.890095  0.159864        NaN           NaN
                      df        sum_sq     mean_sq           F  \
num_op_rev_tl        1.0     222.845061  222.845061  1395.354261
Residual       1348057.0  215291.451776    0.159705          NaN

                      PR(>F)
num_op_rev_tl  3.082405e-305
Residual                 NaN
                      df        sum_sq   mean_sq          F        PR(>F)
num_rev_accts        1.0       5.169191  5.169191  32.334429  1.298198e-08
Residual       1348057.0  215509.127646  0.159866        NaN           NaN
                            df        sum_sq     mean_sq           F  PR(>F)
num_rev_tl_bal_gt_0        1.0     993.114404  993.114404  6240.758182     0.0
Residual             1348057.0  214521.182433    0.159134          NaN     NaN
                 df        sum_sq     mean_sq          F         PR(>F)
num_sats        1.0     148.592553  148.592553  930.098093  3.301927e-204
Residual  1348057.0  215365.704284    0.159760         NaN            NaN
                         df        sum_sq   mean_sq         F   PR(>F)
num_tl_120dpd_2m        1.0       0.328657  0.328657  2.055777  0.15163
Residual          1348057.0  215513.968180  0.159870       NaN      NaN
                     df        sum_sq   mean_sq         F    PR(>F)
num_tl_30dpd        1.0       1.394417  1.394417  8.722234  0.003144
Residual      1348057.0  215512.902420  0.159869       NaN       NaN
                           df        sum_sq    mean_sq          F  \
num_tl_90g_dpd_24m        1.0      20.178059  20.178059  126.226989
Residual            1348057.0  215494.118778   0.159855         NaN

                          PR(>F)
num_tl_90g_dpd_24m  2.750743e-29
Residual                     NaN
                           df        sum_sq      mean_sq          F  PR(>F)
num_tl_op_past_12m        1.0    1510.622877  1510.622877  9515.75132     0.0
Residual            1348057.0  214003.673961     0.158750         NaN     NaN
                       df        sum_sq    mean_sq          F        PR(>F)
pct_tl_nvr_dlq        1.0      22.979108  22.979108  143.751254  4.042621e-33
Residual        1348057.0  215491.317729   0.159853         NaN           NaN
                         df        sum_sq     mean_sq           F  PR(>F)
percent_bc_gt_75        1.0     937.664622  937.664622  5890.787567     0.0
Residual          1348057.0  214576.632215    0.159175          NaN     NaN
                             df        sum_sq     mean_sq          F  \
pub_rec_bankruptcies        1.0     138.273330  138.273330  865.46463
Residual              1348057.0  215376.023507    0.159768        NaN

                             PR(>F)
pub_rec_bankruptcies  3.630776e-190
Residual                        NaN
                  df        sum_sq    mean_sq           F        PR(>F)
```

```
tax_liens        1.0      20.206715  20.206715  126.406267  2.513163e-29
Residual   1348057.0  215494.090122   0.159855         NaN           NaN
                         df         sum_sq      mean_sq            F  PR(>F)
tot_hi_cred_lim         1.0    1268.866650  1268.866650  7983.855565     0.0
Residual          1348057.0  214245.430187     0.158929          NaN     NaN
                           df         sum_sq   mean_sq          F    PR(>F)
total_bal_ex_mort         1.0       2.957407  2.957407  18.49904  0.000017
Residual            1348057.0  215511.339430  0.159868       NaN       NaN
                        df         sum_sq      mean_sq            F  PR(>F)
total_bc_limit         1.0    1088.204378  1088.204378  6841.338719     0.0
Residual         1348057.0  214426.092460     0.159063          NaN     NaN
                                    df         sum_sq   mean_sq         F  \
total_il_high_credit_limit         1.0       0.332603  0.332603  2.080458
Residual                     1348057.0  215513.964234  0.159870       NaN

                              PR(>F)
total_il_high_credit_limit  0.149196
Residual                         NaN
                      df         sum_sq   mean_sq         F    PR(>F)
hardship_flag        1.0       0.039845  0.039845  0.249232  0.617616
Residual       1348058.0  215514.296837  0.159870       NaN       NaN
                            df         sum_sq   mean_sq         F    PR(>F)
disbursement_method        1.0       0.006143  0.006143  0.038422  0.844598
Residual             1348057.0  215514.290695  0.159870       NaN       NaN
                             df         sum_sq       mean_sq              F  \
debt_settlement_flag        1.0   21845.984890  21845.984890  152062.216874
Residual              1348057.0  193668.311947      0.143665            NaN

                      PR(>F)
debt_settlement_flag     0.0
Residual                 NaN
```

Yes - Charged Off

No - Paid Off

In [110…

```python
from scipy import stats
yes = filt_df['loan_amnt'][filt_df['loan_status']==1]
no = filt_df['loan_amnt'][filt_df['loan_status']==0]
stats.f_oneway(yes, no)
```

Out[110…

```
F_onewayResult(statistic=5771.860588732891, pvalue=0.0)
```

In [111…

```python
yes = filt_df['int_rate'][filt_df['loan_status']==1]
no = filt_df['int_rate'][filt_df['loan_status']==0]
stats.f_oneway(yes, no)
```

Out[111…

```
F_onewayResult(statistic=96580.9282659643, pvalue=0.0)
```

In [112…

```python
yes = filt_df['installment'][filt_df['loan_status']==1]
no = filt_df['installment'][filt_df['loan_status']==0]
stats.f_oneway(yes, no)
```

Out[112…

```
F_onewayResult(statistic=3575.796398894918, pvalue=0.0)
```

All the tests shows very small p-values which means there is difference in means between Charged OFF=YES and Paid OFF=No for the numerical variables. This means that the change in the value of the numerical variables has effect on whether it loan was paid or charged off.

COMPARING TWO SAMPLES

Mann and Whitney test

Mann and Whitney's test or Wilcoxon rank-sum test is the non-parametric statistic hypothesis test that is used to analyze the difference between two independent samples of ordinal data. In this test, we have provided two randomly drawn samples and we have to verify whether these two samples is from the same population.

for Loan Amount

In [113…

```python
# code for Mann-Whitney  test
from scipy.stats import mannwhitneyu
# Take batch 1 and batch 2 data as per above example
yes = filt_df['loan_amnt'][filt_df['loan_status']==1]
no = filt_df['loan_amnt'][filt_df['loan_status']==0]

# perform mann whitney test
stat, p_value = mannwhitneyu(yes, no)
print('Statistics=%.2f, p=%.2f' % (stat, p_value))
# Level of significance
alpha = 0.05
# conclusion
if p_value < alpha:
    print('Reject Null Hypothesis (Significant difference between two samples)')
else:
    print('Do not Reject Null Hypothesis (No significant difference between two samples)')
```

```
Statistics=160036383091.00, p=0.00
Reject Null Hypothesis (Significant difference between two samples)
```

for Interest Rate

In [114…

```python
# code for Mann-Whitney U test
from scipy.stats import mannwhitneyu
# Take batch 1 and batch 2 data as per above example
yes = filt_df['int_rate'][filt_df['loan_status']==1]
no = filt_df['int_rate'][filt_df['loan_status']==0]

# perform mann whitney test
stat, p_value = mannwhitneyu(yes, no)
print('Statistics=%.2f, p=%.2f' % (stat, p_value))
# Level of significance
alpha = 0.05
# conclusion
if p_value < alpha:
    print('Reject Null Hypothesis (Significant difference between two samples)')
else:
    print('Do not Reject Null Hypothesis (No significant difference between two samples)')
```

```
Statistics=198516050583.50, p=0.00
Reject Null Hypothesis (Significant difference between two samples)
```

for dti

In [115…

```python
# code for Mann-Whitney U test
from scipy.stats import mannwhitneyu
# Take batch 1 and batch 2 data as per above example
yes = filt_df['dti'][filt_df['loan_status']==1]
no = filt_df['dti'][filt_df['loan_status']==0]

# perform mann whitney test
stat, p_value = mannwhitneyu(yes, no)
print('Statistics=%.2f, p=%.2f' % (stat, p_value))
# Level of significance
alpha = 0.05
# conclusion
if p_value < alpha:
    print('Reject Null Hypothesis (Significant difference between two samples)')
else:
    print('Do not Reject Null Hypothesis (No significant difference between two samples)')
```

```
Statistics=167569938521.00, p=0.00
Reject Null Hypothesis (Significant difference between two samples)
```

and so on, we can check for both samples/groups across various numerical variables

In [ ]:

In [ ]:

In [ ]:

```

```

In [ ]:

```

```