
LendingClub

PROJECT INTERIM REPORT

Batch details	PGP-DSE OCT 23
Team members	Latika Anmol Gupta Pavan Jacob Divya Naorem Galaxy
Domain of Project	Big Data Analytics in Finance
Proposed project title	Predicting Loan Default Risk for Lending Club Borrowers
Group Number	5
Team Leader	Latika
Mentor Name	Charakani Siva Prasad

Dataset name: **Lending Club Loan Data**

Table of Contents

1.PROJECT DESCRIPTION

The Lending Club Loan Data project focuses on predicting the likelihood of borrowers defaulting on their loans. Lending Club, a prominent peer-to-peer lending platform in the United States, connects borrowers seeking funds with investors looking to earn returns. The dataset, covering loans from 2007 to 2015, includes comprehensive details such as loan status, payment history, credit scores, and other borrower attributes.

The primary objective of this project is to develop a machine learning model that accurately identifies high-risk borrowers who are likely to default on their loans. By analyzing various features in the dataset, such as borrower demographics, financial metrics, and loan characteristics, the model aims to assist Lending Club in minimizing financial losses and improving decision-making in loan approvals. Accurate prediction of loan defaults is crucial for managing risk, optimizing the loan portfolio, and ensuring sustainable returns for investors.

In addition to risk assessment, the project aims to enhance operational efficiency and customer segmentation. By identifying patterns and correlations within the data, the analysis can streamline loan origination and underwriting processes, reducing costs and improving customer experience. Moreover, understanding different borrower segments enables tailored loan products and marketing strategies, addressing specific needs of each segment and fostering better customer relationships.

The methodology involves thorough data preprocessing, including handling missing values, scaling features, and detecting outliers. Exploratory Data Analysis (EDA) will be conducted to uncover insights and guide feature engineering. Various machine learning algorithms will be trained and evaluated using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC to ensure the model's effectiveness. This structured approach ensures a robust predictive model that leverages data-driven insights to enhance Lending Club's operations and decision-making processes.

This project employs advanced data analytics and machine learning techniques to enhance the accuracy and efficiency of risk predictions. By systematically processing and analyzing the data, the project seeks to provide actionable insights for improving loan approval processes and minimizing financial losses.

Key Steps and Methodologies:

1. **Undersample Data:** To address the class imbalance in the dataset, where 80% of the loans are fully paid and only 20% are defaulted, undersampling techniques are employed. This involves reducing the number of fully paid loans to ensure a balanced representation of both classes, facilitating more effective model training.
    - **Data Sampling:**
        - 10% data sample: 80k records
        - 25% data sample: 40k records
        - Full dataset: 120k records
2. **Data Understanding:** Initial steps include exploring the dataset to understand its structure and characteristics. This involves examining the shape of the data and the data types (dtypes) of various features.
    - **Shape and Dtypes:**

        - Determine the number of records and features.
        - Identify data types of each feature to ensure proper handling during preprocessing.
3. **Dtypes Casting:** Ensure all features are cast to their original data types for consistency. This step is crucial for accurate data manipulation and model training.
4. **Target Definition:** Clearly define the target variable for the prediction task. In this case, the loan status is the target variable, with 'Fully paid' labeled as 0 and 'Default' labeled as 1.
5. **Exploratory Data Analysis (EDA):** Conduct thorough EDA to uncover underlying patterns and insights within the data. This step includes multiple sub-tasks:
    - **Null Values:**
        - Identify and handle missing values in the dataset.
    - **Outlier Detection:**
        - Detect and manage outliers to ensure they do not skew the analysis.
    - **Missing Value Imputation:**
        - Apply appropriate techniques to impute missing values.
    - **Univariate and Bivariate Analysis:**
        - Perform univariate analysis to understand the distribution of individual features.
        - Conduct bivariate analysis to explore relationships between pairs of features.
    - **Statistical Tests:**
        - Conduct statistical tests to validate the significance of observed patterns and relationships.
6. **Train & Test Split:** Split the dataset into training and testing subsets to evaluate model performance. A common split ratio of 70:30 is used, where 70% of the data is allocated for training and 30% for testing.
7. **Baseline Model:** Develop a baseline model to serve as a reference point for evaluating more complex models. Logistic Regression is selected as the initial baseline model due to its simplicity and effectiveness in binary classification tasks.

Background Research:

According to a study published in Expert Systems with Applications in 2009, Yeh and Lien explored using data mining techniques to predict credit card default. They focused on how selecting the right features impacts model accuracy and compared the performance of different predictive models. They used decision trees, logistic regression, and neural networks to handle complex financial data. Their findings indicated that neural networks, while more accurate, have higher computational costs and are less interpretable than other methods. The study highlights the need to balance model complexity with practical usability in financial applications.

In a 2015 study published in the European Journal of Operational Research, Lessmann et al. compared various machine learning models for predicting credit default, focusing on both performance and interpretability. They evaluated support vector machines, random forests, and gradient boosting. While random forests and gradient boosting offered high predictive accuracy, they lacked transparency. Support vector machines provided a good balance between accuracy and interpretability. This study emphasizes the challenge of finding models that are both accurate and easy to understand, which is crucial for regulatory compliance and risk management.

A 2015 study in Computational Economics by Malekipirbazari and Aksakalli bridged traditional statistical methods and modern machine learning algorithms for credit risk assessment. They discussed feature engineering and model validation techniques, essential for developing robust credit scoring models. Their research showed that machine learning models like random forests and gradient boosting could outperform traditional methods, but feature engineering remains critical. The study illustrates that combining domain knowledge with advanced algorithms can significantly improve model performance, providing a comprehensive approach to credit risk modeling.

In a paper published in the Journal of the Operational Research Society in 2003, Baesens et al. conducted a comparative analysis of various machine learning models for credit scoring, including logistic regression, neural networks, and kernel-based methods. They found that neural networks and kernel-based methods often provided higher accuracy, but logistic regression was preferred for its simplicity and interpretability. The study suggests that the choice of model depends on the financial institution's needs, such as the need for transparency versus the demand for higher predictive power.

Peer-to-peer (P2P) lending is a financial innovation that connects individuals seeking to invest their money with those in need of loans. By cutting out traditional financial intermediaries, P2P lending platforms offer several advantages: borrowers can often secure loans at lower

interest rates, while investors can achieve higher returns on their investments. Lending Club, founded in 2006, is one of the pioneers in this space, operating as a marketplace that matches investors with borrowers. It facilitates personal loans, business loans, and other forms of credit, leveraging data to streamline the lending process and manage risks. This research aims to leverage data-driven insights to enhance Lending Club's decision-making processes, improve loan portfolio performance, and ensure better risk management.

The background research highlights the evolution of credit risk prediction methodologies, emphasizing the balance between model accuracy and interpretability. By building on these insights, this study seeks to develop robust machine learning models that can effectively predict loan default risk for Lending Club borrowers, ultimately contributing to more informed lending practices and improved financial outcomes for both borrowers and investors.

2. DATA DICTIONARY

Column No.	Column Name	Description
1.	acceptD	The date which the borrower accepted the offer
2.	accNowDelinq	The number of accounts on which the borrower is now delinquent.
3.	accOpenPast24Mths	Number of trades opened in past 24 months.
4.	addrState	The state provided by the borrower in the loan application
5.	all_util	Balance to credit limit on all trades
6.	annual_inc_joint	The combined self-reported annual income provided by the co-borrowers during registration
7.	annualInc	The self-reported annual income provided by the borrower during registration.
8.	application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers
9.	avg_cur_bal	Average current balance of all accounts
10.	bcOpenToBuy	Total open to buy on revolving bankcards.
11.	bcUtil	Ratio of total current balance to high credit/credit limit for all bankcard accounts.
12.	chargeoff_within_12_mths	Number of charge-offs within 12 months
13.	collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections
14.	creditPullD	The date LC pulled credit for this loan
15.	delinq2Yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
16.	delinqAmnt	The past-due amount owed for the accounts on which the borrower is now delinquent.
17.	desc	Loan description provided by the borrower
18.	dti	A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income.
19.	dti_joint	A ratio calculated using the co-borrowers' total monthly payments on the total debt obligations, excluding mortgages and the requested LC loan, divided by the co-borrowers' combined self-reported monthly income
20.	earliestCrLine	The date the borrower's earliest reported credit line was opened
21.	effective_int_rate	The effective interest rate is equal to the interest rate on a Note reduced by Lending Club's estimate of the impact of uncollected interest prior to charge off.
22.	emp_title	The job title supplied by the Borrower when applying for the loan.*

23.	empLength	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
24.	expD	The date the listing will expire
25.	expDefaultRate	The expected default rate of the loan.
26.	ficoRangeHigh	The upper boundary range the borrower's FICO at loan origination belongs to.
27.	ficoRangeLow	The lower boundary range the borrower's FICO at loan origination belongs to.
28.	fundedAmnt	The total amount committed to that loan at that point in time.
29.	grade	LC assigned loan grade
30.	homeOwnership	The home ownership status provided by the borrower during registration. Our values are: RENT, OWN, MORTGAGE, OTHER.
31.	id	A unique LC assigned ID for the loan listing.
32.	il_util	Ratio of total current balance to high credit/credit limit on all install acct
33.	ils_exp_d	wholeloan platform expiration date
34.	initialListStatus	The initial listing status of the loan. Possible values are – W, F
35.	inq_fi	Number of personal finance inquiries
36.	inq_last_12m	Number of credit inquiries in past 12 months
37.	inqLast6Mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
38.	installment	The monthly payment owed by the borrower if the loan originates.
39.	intRate	Interest Rate on the loan
40.	isIncV	Indicates if income was verified by LC, not verified, or if the income source was verified
41.	listD	The date which the borrower's application was listed on the platform.
42.	loanAmnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
43.	max_bal_bc	Maximum current balance owed on all revolving accounts
44.	memberId	A unique LC assigned Id for the borrower member.
45.	mo_sin_old_rev_tl_op	Months since oldest revolving account opened
46.	mo_sin_rcnt_rev_tl_op	Months since most recent revolving account opened
47.	mo_sin_rcnt_tl	Months since most recent account opened
48.	mortAcc	Number of mortgage accounts.
49.	msa	Metropolitan Statistical Area of the borrower.
50.	mths_since_last_major_derog	Months since most recent 90-day or worse rating
51.	mths_since_oldest_il_open	Months since oldest bank installment account opened
52.	mths_since_rcnt_il	Months since most recent installment accounts opened
53.	mthsSinceLastDelinq	The number of months since the borrower's last delinquency.
54.	mthsSinceLastRecord	The number of months since the last public record.
55.	mthsSinceMostRecentInq	Months since most recent inquiry.
56.	mthsSinceRecentBc	Months since most recent bankcard account opened.
57.	mthsSinceRecentLoanDelinq	Months since most recent personal finance delinquency.
58.	mthsSinceRecentRevolDelinq	Months since most recent revolving delinquency.

59.	num_accts_ever_120_pd	Number of accounts ever 120 or more days past due
60.	num_actv_bc_tl	Number of currently active bankcard accounts
61.	num_actv_rev_tl	Number of currently active revolving trades
62.	num_bc_sats	Number of satisfactory bankcard accounts
63.	num_bc_tl	Number of bankcard accounts
64.	num_il_tl	Number of installment accounts
65.	num_op_rev_tl	Number of open revolving accounts
66.	num_rev_accts	Number of revolving accounts
67.	num_rev_tl_bal_gt_0	Number of revolving trades with balance >0
68.	num_sats	Number of satisfactory accounts
69.	num_tl_120dpd_2m	Number of accounts currently 120 days past due (updated in past 2 months)
70.	num_tl_30dpd	Number of accounts currently 30 days past due (updated in past 2 months)
71.	num_tl_90g_dpd_24m	Number of accounts 90 or more days past due in last 24 months
72.	num_tl_op_past_12m	Number of accounts opened in past 12 months
73.	open_acc_6m	Number of open trades in last 6 months
74.	open_il_12m	Number of installment accounts opened in past 12 months
75.	open_il_24m	Number of installment accounts opened in past 24 months
76.	open_act_il	Number of currently active installment trades
77.	open_rv_12m	Number of revolving trades opened in past 12 months
78.	open_rv_24m	Number of revolving trades opened in past 24 months
79.	openAcc	The number of open credit lines in the borrower's credit file.
80.	pct_tl_nvr_dlq	Percent of trades never delinquent
81.	percentBcGt75	Percentage of all bankcard accounts > 75% of limit.
82.	pub_rec_bankruptcies	Number of public record bankruptcies
83.	pubRec	Number of derogatory public records
84.	purpose	A category provided by the borrower for the loan request.
85.	reviewStatus	The status of the loan during the listing period. Values: APPROVED, NOT_APPROVED.
86.	reviewStatusD	The date the loan application was reviewed by LC
87.	revolBal	Total credit revolving balance
88.	revolUtil	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
89.	serviceFeeRate	Service fee rate paid by the investor for this loan.
90.	subGrade	LC assigned loan subgrade
91.	tax_liens	Number of tax liens
92.	term	The number of payments on the loan. Values are in months and can be either 36 or 60.
93.	title	The loan title provided by the borrower
94.	tot_coll_amt	Total collection amounts ever owed
95.	tot_cur_bal	Total current balance of all accounts
96.	tot_hi_cred_lim	Total high credit/credit limit
97.	total_bal_il	Total current balance of all installment accounts
98.	total_cu_tl	Number of finance trades
99.	total_il_high_credit_limit	Total installment high credit/credit limit
100.	total_rev_hi_lim	Total revolving high credit/credit limit
101.	totalAcc	The total number of credit lines currently in the borrower's credit file

102.	totalBalExMort	Total credit balance excluding mortgage
103.	totalBcLimit	Total bankcard high credit/credit limit
104.	url	URL for the LC page with listing data.
105.	verified_status_joint	Indicates if the co-borrowers' joint income was verified by LC, not verified, or if the income source was verified
106.	zip_code	The first 3 numbers of the zip code provided by the borrower in the loan application.
107.	revol_bal_joint	Sum of revolving credit balance of the co-borrowers, net of duplicate balances
108.	sec_app_fico_range_low	FICO range (high) for the secondary applicant
109.	sec_app_fico_range_high	FICO range (low) for the secondary applicant
110.	sec_app_earliest_cr_line	Earliest credit line at time of application for the secondary applicant
111.	sec_app_inq_last_6mths	Credit inquiries in the last 6 months at time of application for the secondary applicant
112.	sec_app_mort_acc	Number of mortgage accounts at time of application for the secondary applicant
113.	sec_app_open_acc	Number of open trades at time of application for the secondary applicant
114.	sec_app_revol_util	Ratio of total current balance to high credit/credit limit for all revolving accounts
115.	sec_app_open_act_il	Number of currently active installment trades at time of application for the secondary applicant
116.	sec_app_num_rev_accts	Number of revolving accounts at time of application for the secondary applicant
117.	sec_app_chargeoff_within_12_mths	Number of charge-offs within last 12 months at time of application for the secondary applicant
118.	sec_app_collections_12_mths_ex_med	Number of collections within last 12 months excluding medical collections at time of application for the secondary applicant
119.	sec_app_mths_since_last_major_derog	Months since most recent 90-day or worse rating at time of application for the secondary applicant
120.	disbursement_method	The method by which the borrower receives their loan. Possible values are: CASH, DIRECT_PAY

3.DATA DESCRIPTION

1. Dataset and Datatypes

```
In [4]: df = pd.read_csv('loan.csv')
        df.head()
```

Out[4]:

	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	a
0	NaN	NaN	2500	2500	2500.0	36 months	13.56	84.92	C	C1	Chef	10+ years	RENT	
1	NaN	NaN	30000	30000	30000.0	60 months	18.94	777.23	D	D2	Postmaster	10+ years	MORTGAGE	
2	NaN	NaN	5000	5000	5000.0	36 months	17.97	180.69	D	D1	Administrative	6 years	MORTGAGE	
3	NaN	NaN	4000	4000	4000.0	36 months	18.94	146.51	D	D2	IT Supervisor	10+ years	MORTGAGE	
4	NaN	NaN	30000	30000	30000.0	60 months	16.14	731.78	C	C4	Mechanic	10+ years	MORTGAGE	

The Lending Club dataset is a comprehensive financial dataset containing information about loans issued through the Lending Club platform. It includes details such as id, member id, loan amount, interest rate, term, emp length, borrower's credit score, employment information, and loan status.

```
In [5]: df.dtypes
```

Out[5]:

```
id                      float64
member_id               float64
loan_amnt                 int64
funded_amnt               int64
funded_amnt_inv         float64
term                     object
int_rate                float64
installment             float64
grade                    object
sub_grade                object
emp_title                object
emp_length               object
home_ownership           object
annual_inc              float64
verification_status      object
issue_d                  object
loan_status              object
pymnt_plan               object
url                     float64
```

The dataset contains 22,60,668 rows and 147 columns. Most columns have complete data for all entries like ‘loan_amt’, ‘funded_amnt’, ‘funded_amnt_inv’, ‘term’ and more.But there are some with missing values, such as ‘id’, ‘memeber_id’, ‘emp_title’, ‘emp_length’ and so on. The data types are a mix of integers, floats, and objects.

2. Undersampling the dataset

```
#Undersample the data
# Calculate 20% of the total rows
sample_size = int(0.2 * len(df2))

# Desired proportions
prop_1 = 0.7991
prop_0 = 0.2009

# Calculate the number of samples needed for each loan_status
sample_size_1 = int(sample_size * prop_1)
sample_size_0 = int(sample_size * prop_0)

# Separate the dataset by loan_status
df_1 = df2[df2['current_loan_status'] == 1]
df_0 = df2[df2['current_loan_status'] == 0]

# Perform undersampling
undersampled_df_1 = df_1.sample(n=sample_size_1, random_state=42)
undersampled_df_0 = df_0.sample(n=sample_size_0, random_state=42)

# Combine the undersampled dataframes
undersampled_df = pd.concat([undersampled_df_1, undersampled_df_0])

# Shuffle the undersampled dataset to mix the rows
undersampled_df = undersampled_df.sample(frac=1, random_state=42).reset_index(drop=True)

# Save the undersampled dataset to a new CSV file
undersampled_df.to_csv('final_undersampled_df_change.csv', index=False)

print(f'Original dataset size: {len(df)}')
print(f'Undersampled dataset size: {len(undersampled_df)}')
print(f'Number of loan_status 1: {undersampled_df["current_loan_status"].value_counts()[1]}')
print(f'Number of loan_status 0: {undersampled_df["current_loan_status"].value_counts()[0]}')
```

```
Original dataset size: 2260668
Undersampled dataset size: 261276
Number of loan_status 1: 208786
Number of loan_status 0: 52490
```

We had '**Original dataset size:** 2,260,668' then we did '**Undersampled dataset size:** 261,276'. There were number of non-paid rows was 52490 and paid was 208786. The output shows the undersampled dataset has a closer representation of the desired class proportions which is 79.91% for loan_status 1 which is 'Does not meet the credit policy. Status: Charged Off', 'Charged Off', 'Default' and 20.09% for loan_status 0 which means "Does not meet the credit policy. Status: Fully Paid", "Fully Paid". Overall, the undersampling process seems to have achieved the intended outcome: reducing the dataset size while balancing the class distribution to address the class imbalance.

3. Data Type Conversion

```
undersampled_df2['earliest_cr_line'] = pd.to_datetime(undersampled_df2['earliest_cr_line'])
undersampled_df2['issue_d'] = pd.to_datetime(undersampled_df2['issue_d'])

undersampled_df2['months_to_issue'] =(undersampled_df2['issue_d'] - undersampled_df2['earliest_cr_line']).dt.days / 30
undersampled_df2['months_to_issue'] = undersampled_df2['months_to_issue'].fillna(0).astype(int)
```

```
undersampled_df2['term'] = undersampled_df2['term'].astype(str).str.replace(' months', '').astype(int)
```

```
undersampled_df7['current_loan_status']=undersampled_df7['current_loan_status'].astype(object)
```

Here, for the four columns 'earliest_cr_line', 'months_to_issue', 'term' and 'current_loan_status', the following process takes place:

- **'earliest_cr_line'**: converted from text to datetime format.
- **'months_to_issue'**: find the difference between two columns and to get an approximate number of months, we divide this difference by 30 and convert it into an int.
- **'Term':** We remove the word "months" from each value and convert the cleaned values to integers. It was then converted into an object later on.
- '**Current_loan_status**': We convert the column from int to object.

4. Dropping Unwanted values

```
max_missing = undersampled_df.columns[(round(undersampled_df.isnull().sum()/len(undersampled_df.index), 2)*100)>70]
max_missing
```

```
Index(['id', 'member_id', 'url', 'desc', 'mths_since_last_record',
       'next_pymnt_d', 'mths_since_last_major_derog', 'annual_inc_joint',
       'dti_joint', 'verification_status_joint', 'mths_since_recent_bc_dlq',
       'revol_bal_joint', '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',
       'loan_condition'],
      dtype='object')
```

```
undersampled_df.drop(columns=max_missing,axis=1,inplace=True)
```

```
undersampled_df.drop(columns = ["funded_amnt", "funded_amnt_inv"], axis=1,inplace=True)
```

```
undersampled_df.drop(columns = ["zip_code", "total_pymnt", "total_pymnt_inv", "total_rec_prncp", "last_pymnt_d", "last_credit_pul
                    axis=1,inplace=True)
```

```
undersampled_df2.drop(columns="addr_state", axis=1,inplace=True)
```

```
undersampled_df2.drop(columns=['emp_title','title'],axis=1,inplace=True)
```

The first code outputs an empty list created to indicate the columns have more than 70% of the null values like 'id', 'member_id', 'url', 'desc', 'mths_since_last_record', 'next_pymnt_d' and so on. Then we dropped all the columns from max_missing. Afterwards, we removed funded_amnt, funded_amnt_inv, zip_code, total_payment, total_rec_prcnp and so on to avoid data leaking.

As we mentioned above, we did the difference of issue_d and earliest_cr_line which results in **months_to_issue.** That is why we dropped both columns as well. Here we trying to predict the difference between the earliest credit line at the time of application for the secondary applicant and the date on which the loan was issued. After dropping the column, we were left with columns and rows 76.

4.NULL VALUE AND OUTLIERS TREATMENT

```
undersampled_df2.isnull().sum()
```

```
loan_amnt                      0
term                           0
int_rate                       0
installment                    0
grade                          0
sub_grade                      0
emp_title                  20350
emp_length                 19018
home_ownership                 0
annual_inc                     1
verification_status            0
issue_d                        0
pymnt_plan                     0
purpose                        0
title                       4284
dti                           63
delinq_2yrs                    0
earliest_cr_line               6
inq_last_6mths                 6
mths_since_last_delinq    128065
open_acc                       6
pub_rec                        6
revol_bal                      0
revol_util                   173
total_acc                      6
initial_list_status            0
out_prncp                      0
out_prncp_inv                  0
total_rec_int                  0
total_rec_late_fee             0
recoveries                     0
collection_recovery_fee        0
```

last_pymnt_amnt	0
collections_12_mths_ex_med	27
policy_code	0
application_type	0
acc_now_delinq	6
tot_coll_amt	11725
tot_cur_bal	11725
open_acc_6m	151062
open_act_il	151062
open_il_12m	151062
open_il_24m	151062
mths_since_rcnt_il	154355
total_bal_il	151062
il_util	165063
open_rv_12m	151062
open_rv_24m	151062
max_bal_bc	151062
all_util	151070
total_rev_hi_lim	11725
inq_fi	151062
total_cu_tl	151062
inq_last_12m	151062
acc_open_past_24mths	8196
avg_cur_bal	11730
bc_open_to_buy	11039
bc_util	11192
chargeoff_within_12_mths	27
delinq_amnt	6
mo_sin_old_il_acct	19482
mo_sin_old_rev_tl_op	11726
mo_sin_rcnt_rev_tl_op	11726
mo_sin_rcnt_tl	11725
mort_acc	8196
mths_since_recent_bc	10828

```
mths_since_recent_inq              28090
mths_since_recent_revol_delinq    171732
num_accts_ever_120_pd              11725
num_actv_bc_tl                     11725
num_actv_rev_tl                    11725
num_bc_sats                         9649
num_bc_tl                          11725
num_il_tl                          11725
num_op_rev_tl                      11725
num_rev_accts                      11725
num_rev_tl_bal_gt_0                11725
num_sats                            9649
num_tl_120dpd_2m                   24013
num_tl_30dpd                       11725
num_tl_90g_dpd_24m                 11725
num_tl_op_past_12m                 11725
pct_tl_nvr_dlq                     11751
percent_bc_gt_75                   11112
pub_rec_bankruptcies                 316
tax_liens                             15
tot_hi_cred_lim                    11725
total_bal_ex_mort                   8196
total_bc_limit                      8196
total_il_high_credit_limit         11725
hardship_flag                          0
disbursement_method                    0
debt_settlement_flag                   0
current_loan_status                    0
region                                 0
months_to_issue                        0
dtype: int64
```

In that particular code, we check for the null values. Most features like loan amount, interest rate, etc. have no missing values. However, some features regarding employment, housing, and credit history have a high percentage of missing values. After that, we made the **columns_to_impute** dataset. Therefore, missing data poses a challenge. Cleaning or estimating these missing values might be necessary before using this data to build models that assess loan eligibility or predict borrower behavior.

```
columns_to_impute = ['annual_inc', 'dti', 'mths_since_last_delinq', 'pub_rec', 'revol_util', 'total_acc', 'tot_coll_amt',
                     'tot_cur_bal', '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', 'total_rev_hi_lim',
                     'inq_fi', 'total_cu_tl', 'inq_last_12m', 'avg_cur_bal', 'bc_open_to_buy', 'bc_util','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_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_op_past_12m',
                     'percent_bc_gt_75', 'tax_liens', 'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit',
                     'total_il_high_credit_limit', 'delinq_amnt', 'open_acc', 'inq_last_6mths']

undersampled_df6[columns_to_impute] = undersampled_df6[columns_to_impute].apply(lambda x: x.fillna(x.median()))
```

```
columns_to_impute_mode = ['emp_length', 'mths_since_recent_revol_delinq', 'num_accts_ever_120_pd', 'pct_tl_nvr_dlq',
                          'pub_rec_bankruptcies']

undersampled_df6[columns_to_impute_mode] = undersampled_df6[columns_to_impute_mode].apply(lambda x: x.fillna(x.mode()[0]))
```

```
columns_to_impute_zero = ['collections_12_mths_ex_med', 'acc_now_delinq', 'chargeoff_within_12_mths', 'num_tl_120dpd_2m',
                          'num_tl_30dpd', 'num_tl_90g_dpd_24m']

undersampled_df6[columns_to_impute_zero] = undersampled_df6[columns_to_impute_zero].apply(lambda x: x.fillna(0))
```

```
columns_to_impute_mean = ['acc_open_past_24mths' ]
undersampled_df6[columns_to_impute_mean] = undersampled_df6[columns_to_impute_mean].apply(lambda x: x.fillna(x.mean()))
```

The code provided checks for missing values in various columns of a dataset. Each name is listed, along with the count of missing values for that column. It focuses on a specific set of columns likely containing missing values (annual_income, credit utilization ratios, account details etc.). These missing values were problematic. To address this, imputation is done. It replaces missing values in these columns with the median value for that specific column in the dataset. The median is chosen because the data is highly skewed, and highly affected by the outliers. Furthermore, we replace null values with mode in 'emp_length', 'mths_since_recent_revol_delinq', 'num_accts_ever_120_pd' and, 'acc_now_delinq', 'chargeoff_within_12_mths', 'num_tl_120dpd_2m', 'num_tl_30dpd' with zero(0) because frequency of the values are high that is why use mode imputation. In addition, acc_open_past_24mths with mean because data is almost normally distributed and not affected by the outliers.

5. Feature Engineering

```
state_to_timezone = {
    'GA': 'Eastern', 'CA': 'Pacific', 'TX': 'Central', 'MS': 'Central', 'IN': 'Eastern',
    'VA': 'Eastern', 'NJ': 'Eastern', 'OR': 'Pacific', 'PA': 'Eastern', 'AZ': 'Mountain',
    'MN': 'Central', 'CO': 'Mountain', 'CT': 'Eastern', 'NY': 'Eastern', 'SC': 'Eastern',
    'KY': 'Eastern', 'OH': 'Eastern', 'NC': 'Eastern', 'OK': 'Central', 'AR': 'Central',
    'AL': 'Central', 'NV': 'Pacific', 'FL': 'Eastern', 'MO': 'Central', 'NM': 'Mountain',
    'IL': 'Central', 'MA': 'Eastern', 'LA': 'Central', 'MD': 'Eastern', 'WI': 'Central',
    'MI': 'Eastern', 'RI': 'Eastern', 'SD': 'Central', 'WA': 'Pacific', 'ID': 'Mountain',
    'TN': 'Central', 'UT': 'Mountain', 'KS': 'Central', 'HI': 'Hawaii-Aleutian', 'NH': 'Eastern',
    'DE': 'Eastern', 'NE': 'Central', 'ND': 'Central', 'MT': 'Mountain', 'WY': 'Mountain',
    'ME': 'Eastern', 'DC': 'Eastern', 'WV': 'Eastern', 'AK': 'Alaska', 'VT': 'Eastern',
    'IA': 'Central'
}
```

```
undersampled_df2['region'] = undersampled_df2['addr_state'].map(state_to_timezone)
```

tot_hi_cred_lim	total_bal_ex_mort	total_bc_limit	total_il_high_credit_limit	hardship_flag	disbursement_method	debt_settlement_flag	current_loan_status	region
100396.0	107553.0	1300.0	98096.0	N	Cash	N	1.0	Eastern
115639.0	93867.0	3500.0	105439.0	N	Cash	Y	1.0	Pacific
98743.0	70914.0	14400.0	73243.0	N	Cash	N	1.0	Central
NaN	NaN	NaN	NaN	N	Cash	N	1.0	Pacific
12200.0	8762.0	7500.0	0.0	N	Cash	N	0.0	Central

This code performs feature engineering by creating a new column called ‘region’ based on the U.S. state abbreviations in the ‘addr_state’ column. The ‘region maps each state abbreviation (e.g., ‘GA’ for Georgia in ‘Eastern’, ‘CA’ maps to ‘Pacific’, ‘TX’ to ‘Central’, and so on). By using ‘.map(state_to_timezone)’ the code applies this mapping to the ‘addr_state’ column. As a result, a new column called ‘region’ is added to the dataset, indicating the 6 time zones associated with each state.

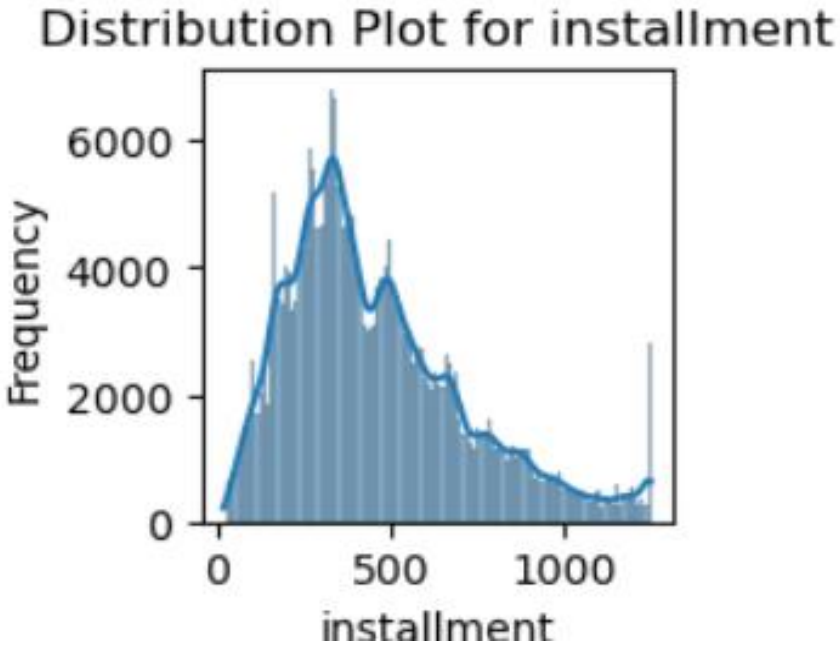
6. Dropping Noise Rows

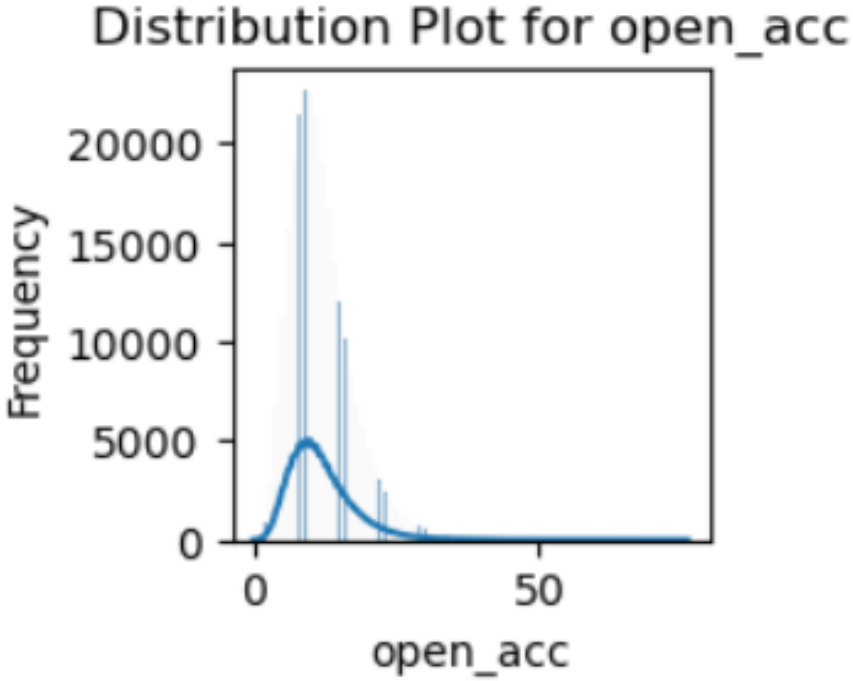
```
scaled_df2.drop(columns=['collections_12_mths_ex_med','acc_now_delinq','policy_code'],axis=1,inplace=True)
```

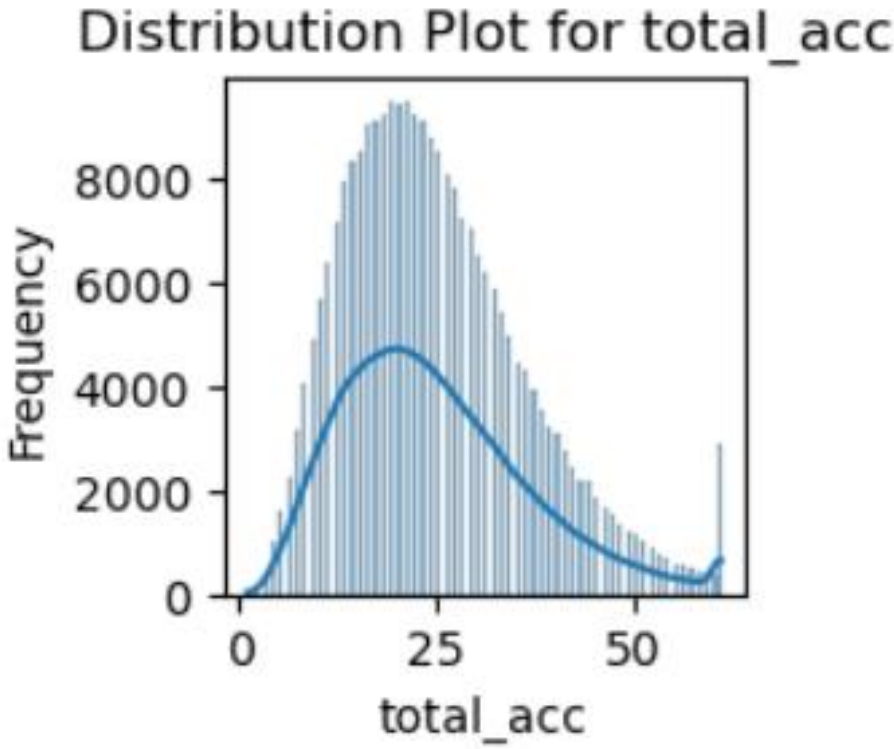
```
scaled_df2.drop(columns=['installment','out_prncp','collection_recovery_fee','avg_cur_bal','bc_util','num_actv_rev_tl',
                         'num_bc_sats','num_op_rev_tl','num_rev_accts','num_rev_tl_bal_gt_0','num_sats','percent_bc_gt_75',
                         'tot_hi_cred_lim','total_bc_limit','total_il_high_credit_limit','months_to_issue'],axis=1,inplace=True)
```

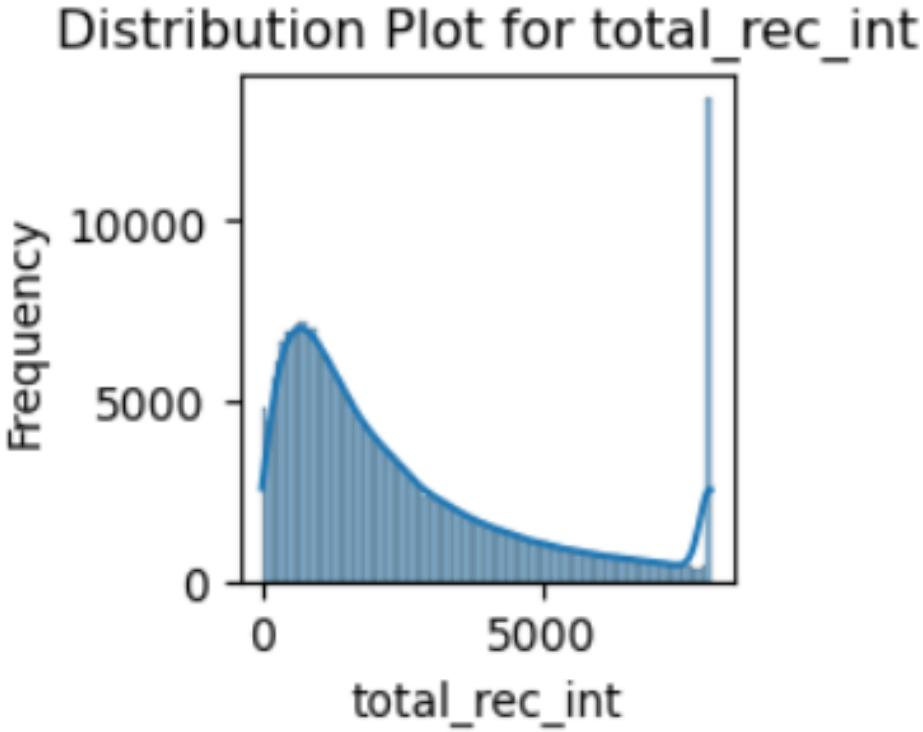
In this code, we drop some columns as they showed no relation for eg. 'collections_12_mths_ex_med','acc_now_delinq', and 'policy_code'. On the other hand,'installment','out_prncp','collection_recovery_fee','avg_cur_bal','bc_util','num_actv_rev_tl','num_bc_sats’ and etc are dropped because they are highly correlated.

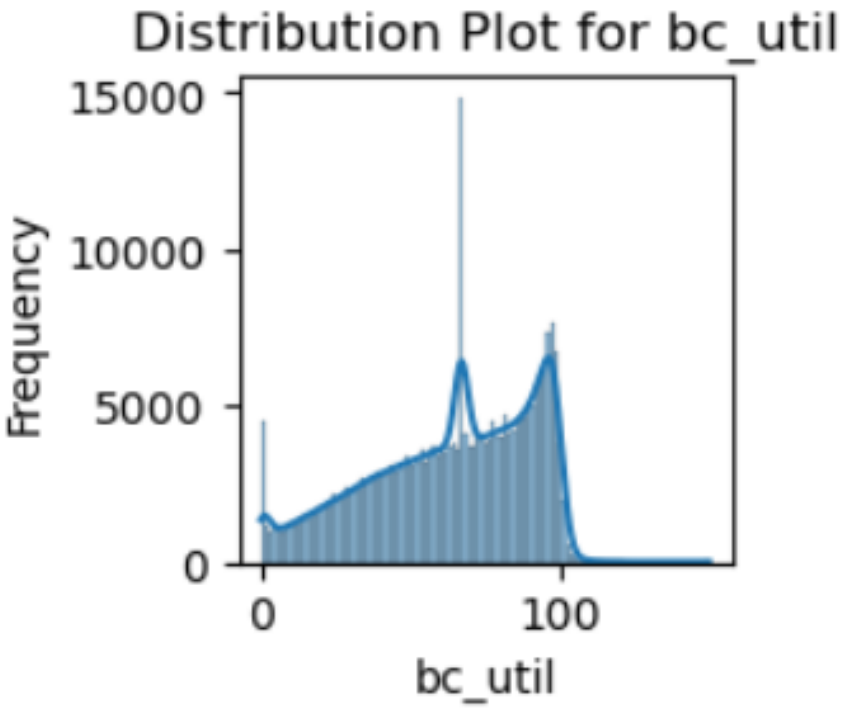
7. Skewness and Kurtosis






Distribution Plot for total_acc
Frequency
8000
6000
4000
2000
0
0
25
50
total_acc


Distribution Plot for total_rec_int
Frequency
10000
5000
0
0
5000
total_rec_int



These observations suggest that there are a few outliers in the data that are driving the skewness and kurtosis. It is also possible that the data is not normally distributed, which could explain the kurtosis.

8. Target Variable

```
paid = ["Does not meet the credit policy. Status:Fully Paid", "Fully Paid"]

# good_loan -1 bad_loan - 0
not_paid = ['Does not meet the credit policy. Status:Charged Off', 'Charged Off','Default']

df['loan_condition'] = np.nan

def loan_type(status):
    if status in not_paid:
        return 0
    elif status in paid:
        return 1

df['current_loan_status']=df['loan_status'].apply(loan_type)
```

```
df.groupby(by=['current_loan_status']).count()
```

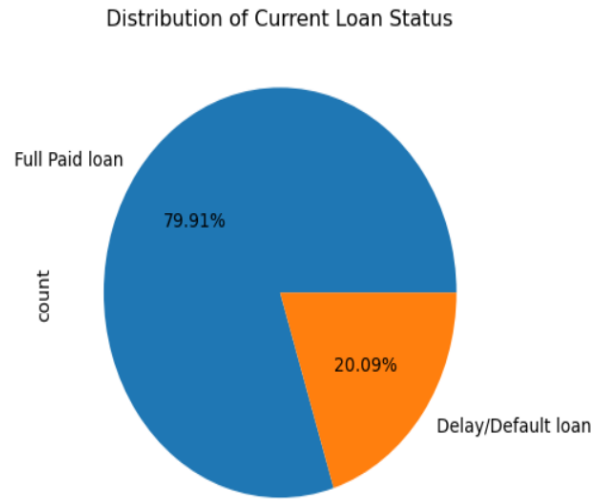
current_loan_status	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	hon
0.0	0	0	1043940	1043940	1043940	1043940	1043940	1043940	1043940	1043940	982945	988861	
1.0	0	0	262447	262447	262447	262447	262447	262447	262447	262447	240701	242035	

```
df2 = df[df['current_loan_status'].isin([0,1])]
df2.count()
```

```
id                          0
member_id                   0
loan_amnt             1306387
funded_amnt           1306387
funded_amnt_inv       1306387
term                  1306387
int_rate              1306387
installment           1306387
grade                 1306387
sub_grade             1306387
emp_title             1223646
emp_length            1230896
home_ownership        1306387
annual_inc            1306383
verification_status    1306387
issue_d               1306387
loan_status           1306387
pymnt_plan            1306387
url                         0
```

```
print(df['current_loan_status'].value_counts())
df['current_loan_status'].value_counts().plot(kind='pie',autopct='%1.2f%%', labels=['Full Paid loan','Delay/Default loan'])
plt.title('Distribution of Current Loan Status')
```

```
current_loan_status
0.0    1043940
1.0     262447
Name: count, dtype: int64

Text(0.5, 1.0, 'Distribution of Current Loan Status')
```



The ‘paid’ list contains two strings representing loan statuses: “Does not meet the credit policy. Status:Fully Paid” and “Fully Paid” while the ‘not_paid’ list includes three strings for loan statuses: “Does not meet the credit policy. Status:Charged Off,” “Charged Off,” and “Default”. A new column called loan_condition is added to the DataFrame (df). Initially, all values in this column are set to NaN (missing). As for the loan_type function takes a loan status as input, If the status is in the not_paid list, it returns 1 (indicating a bad loan) and if the status is in the paid list, it returns 0 (indicating a good loan). Then a new column called current_loan_status is created when the loan_type function is applied to each value in the loan_status column of the DataFrame. The loan_type function is applied to each value in the loan_status column of the DataFrame. For filtering relevant rows a new DataFrame (df2) is created where current_loan_status is either 0 (fully paid) or 1 (delayed/default). Finally the distribution of loan statuses is plotted as a pie chart, showing the percentage of fully paid loans and delayed/default loans.

5. UNIVARIATE AND BIVARIATE ANALYSIS

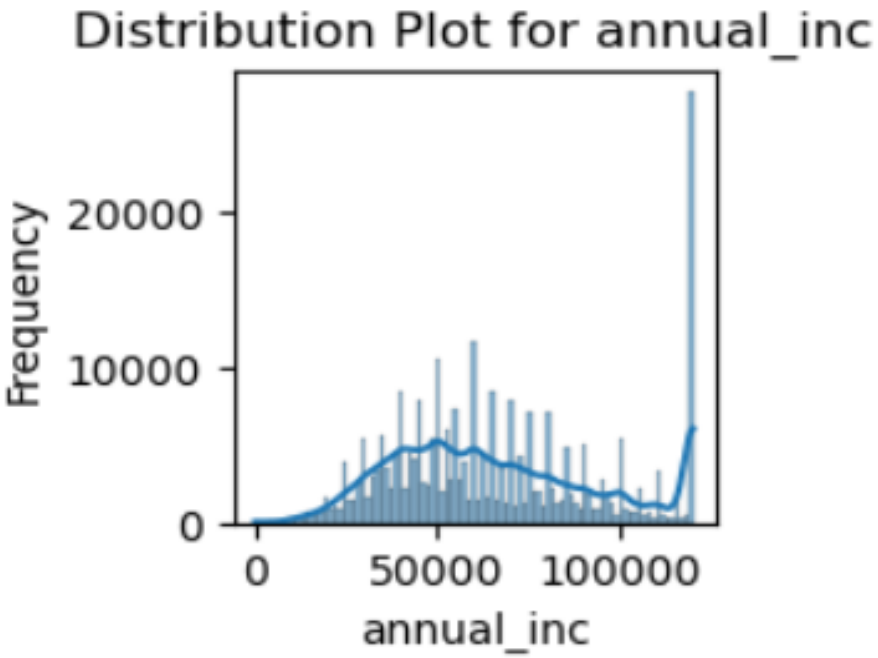
1. Univariate Analysis

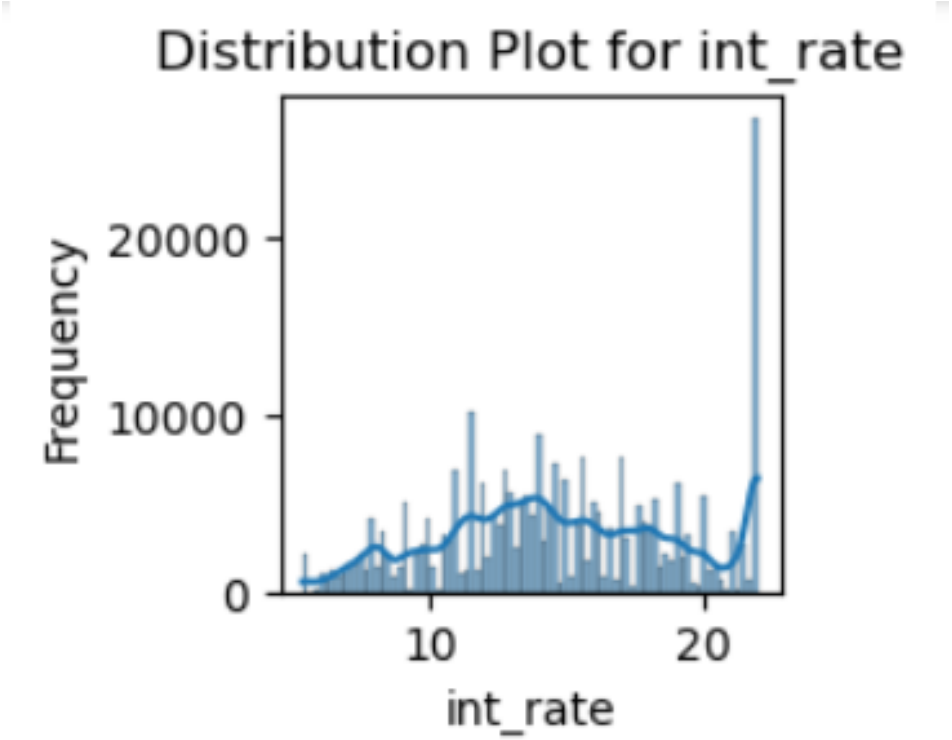
1.1. Numerical

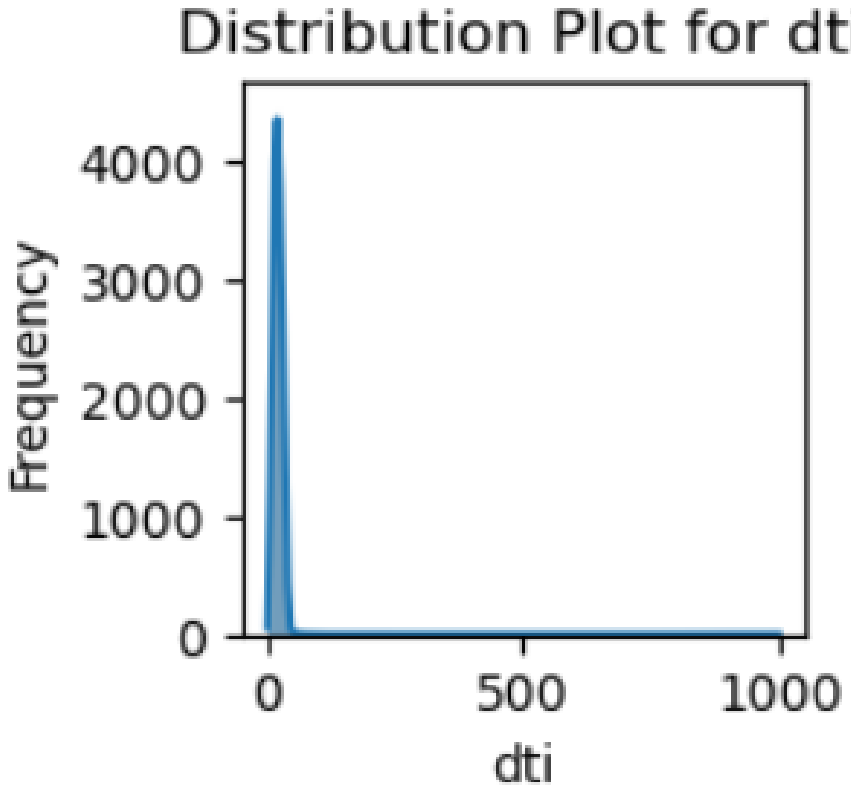
```
def plot_distplots(df, columns):
    """
    Plots distplots for each column in the columns list.

    Parameters:
    df (pandas.DataFrame): The DataFrame containing the data.
    columns (list): List of column names to plot distplots for.
    """
    for column in columns:
        plt.figure(figsize=(2, 2))
        sns.histplot(df[column], kde=True)
        plt.title(f'Distribution Plot for {column}')
        plt.xlabel(column)
        plt.ylabel('Frequency')
        plt.show()
```

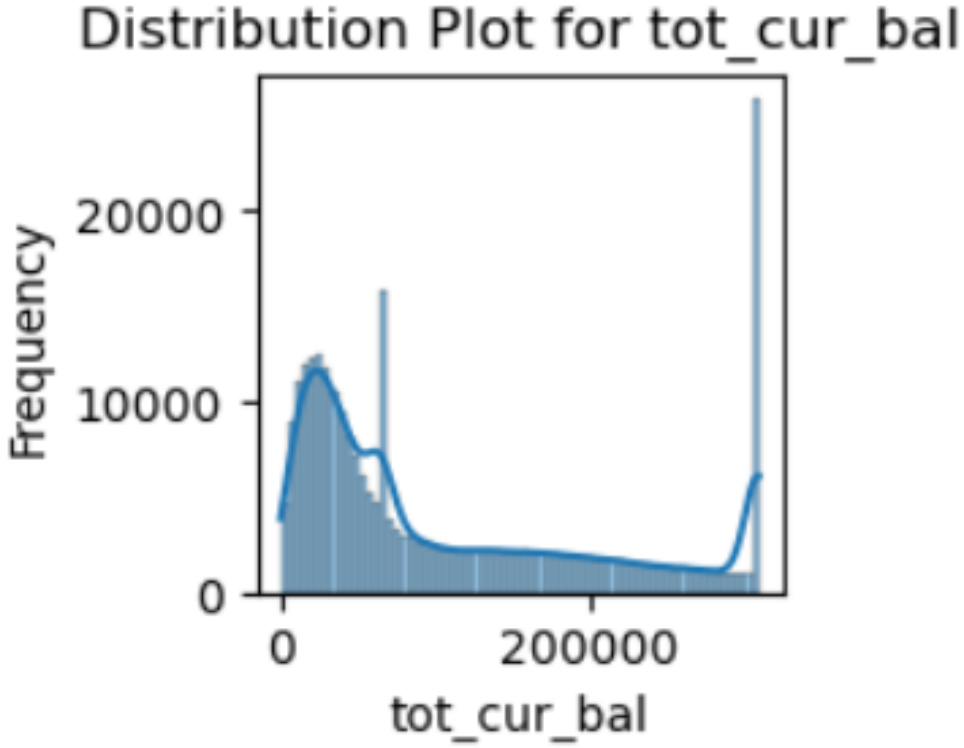
```
numeric_columns= num_df.columns.to_list()
plot_distplots(num_df,numeric_columns)
```

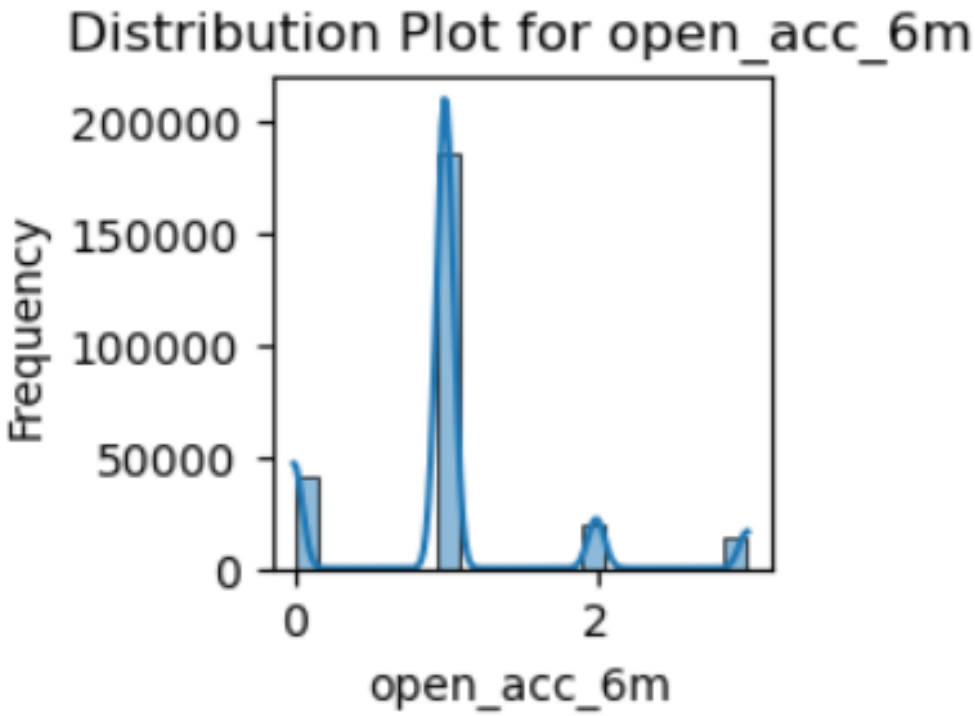
This code offers a function named plot_distplots for visualizing data distributions across multiple columns called 'Int_rate', 'Annual_inc' , 'Dti', 'Inq_last_6mths' : ''Mths_since_last_delinq', 'tot_cur_bal', 'open_acc_6m' and so on.


Distribution Plot for annual_inc
Frequency
20000
10000
0
0
50000
100000
annual_inc


Distribution Plot for int_rate
Frequency
20000
10000
0
10
20
int_rate


Distribution Plot for dti
Frequency
4000
3000
2000
1000
0
0
500
1000
dti


Distribution Plot for tot_cur_bal
Frequency
20000
10000
0
0
200000
tot_cur_bal

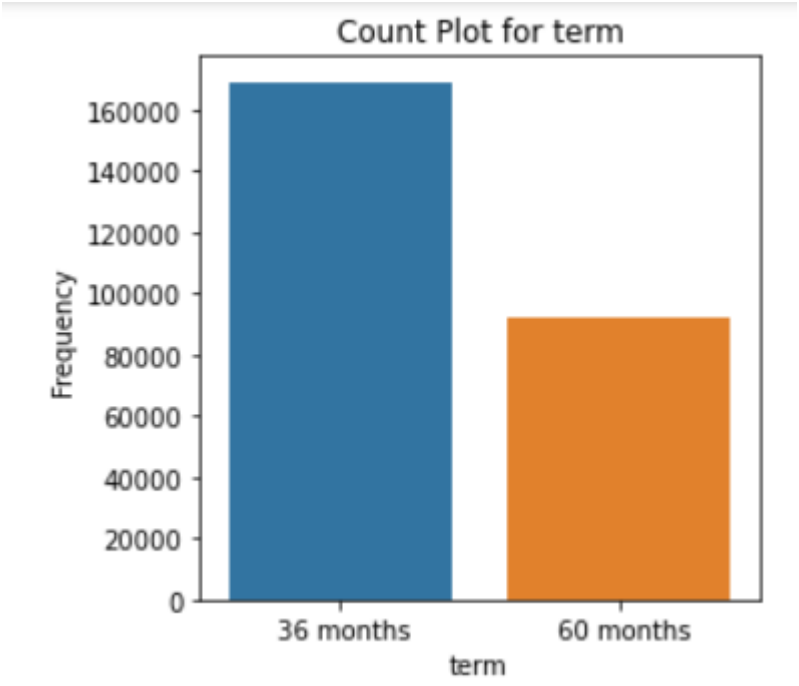


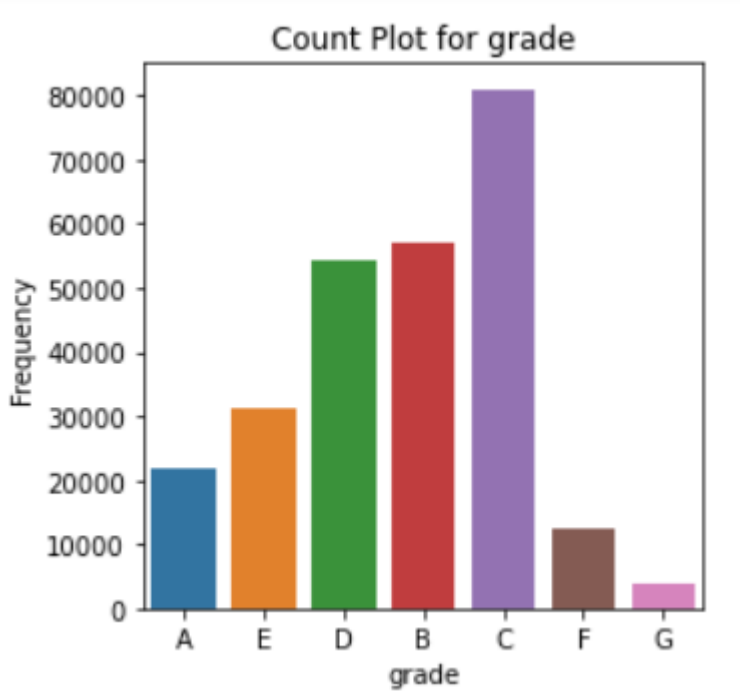
- **'Int_rate':** The distribution plot for the interest rate (int_rate) shows a significant concentration of data points around the 10-20% range, with a noticeable spike at approximately 20%. The plot suggests a relatively consistent distribution within this range, except for the sharp increase at the end.
- **'Annual_inc':** The distribution plot for annual income (annual_inc) indicates that most data points are concentrated between $25,000 and $75,000, with a sharp spike occurring at approximately $100,000. The distribution shows a gradual increase within the primary range before this significant spike at the higher income level.
- **'Dti':** The distribution plot for debt-to-income ratio (dti) shows a high concentration of data points near 0, with frequency rapidly decreasing as the dti value increases. The plot indicates that most values are clustered at the low end of the dti spectrum, with very few instances beyond 50.
- **'Tot_cur_bal' :** The distribution plot for the `tot_cur_bal` variable shows a high concentration of values at lower balances, with a significant drop followed by a long tail and occasional spikes at higher balances. This suggests that most observations have low `tot_cur_bal` values, with fewer observations spread across a wider range of higher balances.
- **'Open_acc_6m':** The distribution plot for the `open_acc_6m` variable shows a sharp peak at 1, indicating that most observations have exactly one account opened in the last 6 months. There are also smaller peaks at 0, 2, and 3, suggesting some variation in the number of accounts opened, but the majority have just one.

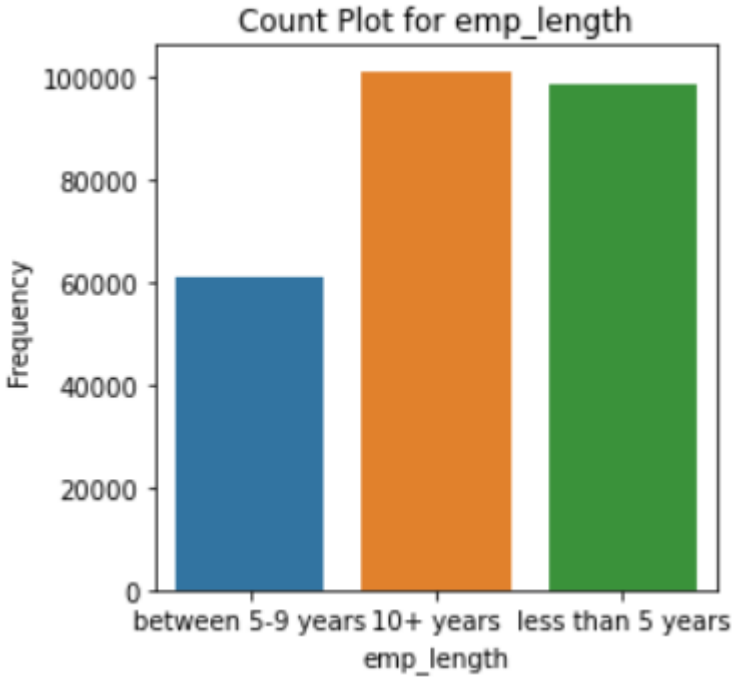
1.2. Categorical

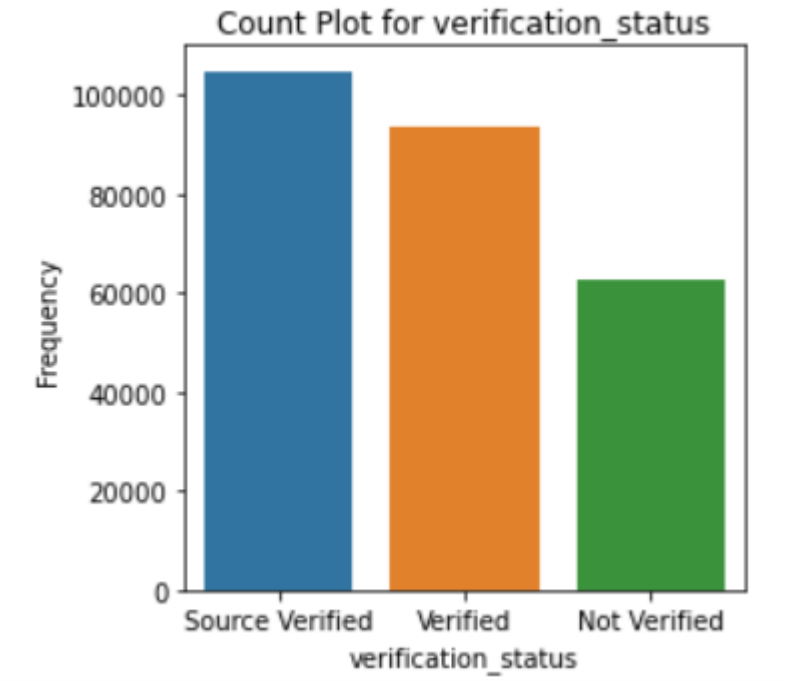
```
def count_plots(df, columns):

    for column in columns:
        plt.figure(figsize=(10, 6))
        sns.countplot(df[column])
        plt.title(f'Count Plot for {column}')
        plt.xlabel(column)
        plt.ylabel('Frequency')
        plt.show()
```

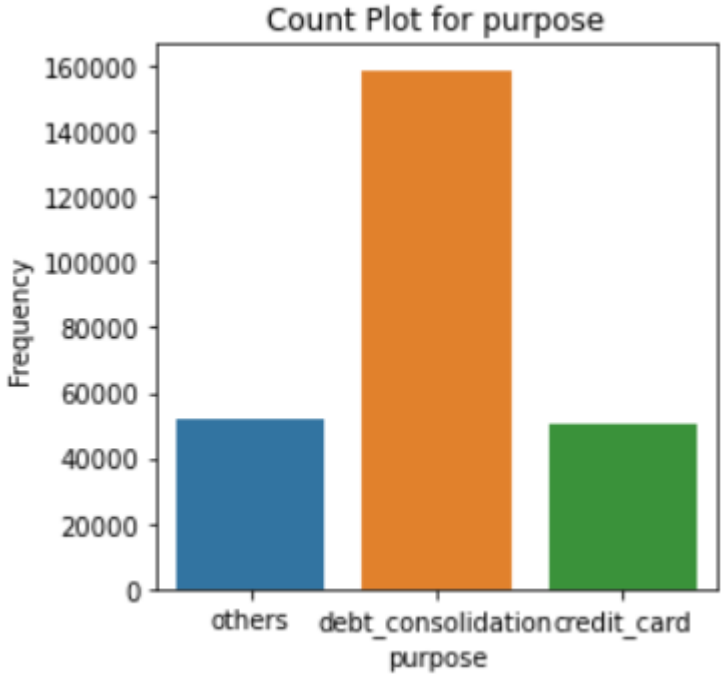
```
count_plots(category_df,class_col)
```




Count Plot for grade
Frequency
80000
70000
60000
50000
40000
30000
20000
10000
0
A
E
D
B
C
F
G
grade


Count Plot for emp_length
Frequency
100000
80000
60000
40000
20000
0
between 5-9 years
10+ years
less than 5 years
emp_length





This code offers a function named count_plots for visualising data distributions across multiple columns called 'term', 'grade', 'emp_length', 'home_ownership': 'verification_status', 'pymnt_plan', 'purpose' and so on.

•**'term'**: The count plot shows the distribution of loan terms (duration) in the dataset. There are two categories: '36 months' and '60 months'. The significant difference in frequencies suggests an imbalance between the two terms. '36 months' loans are

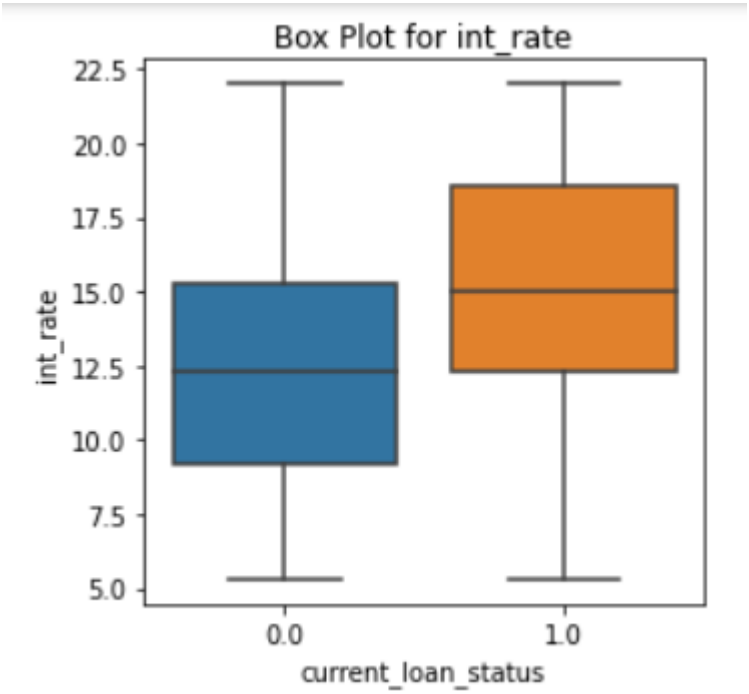
more common than ‘60 months’ loans. Lenders or investors might consider the distribution when making decisions.

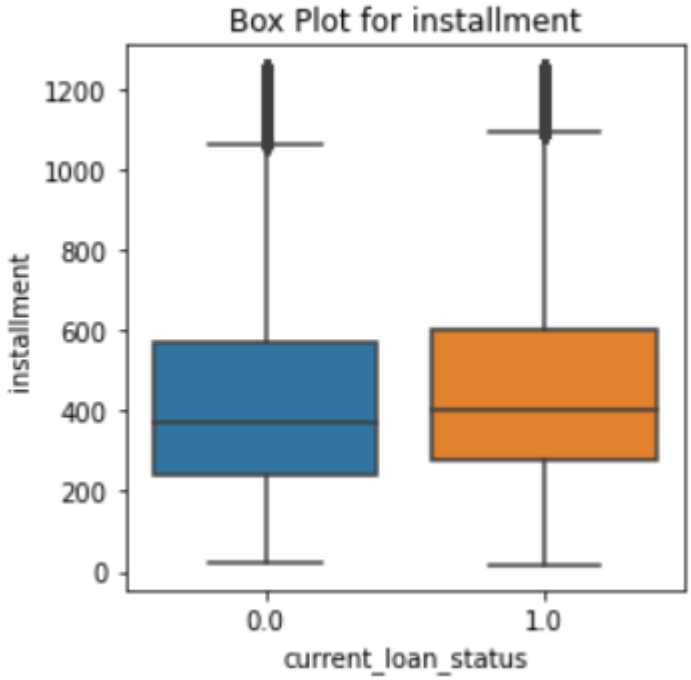
•**'emp_length'**: There are three categories: ‘between 5-9 years’, ‘10+ years’, and ‘less than 5 years’. The plot provides insights into workforce experience. It helps identify whether the majority of employees are relatively new (less than 5 years) or have significant tenure (10+ years). As such organisations can use this information to understand the composition of their workforce in terms of experience levels.

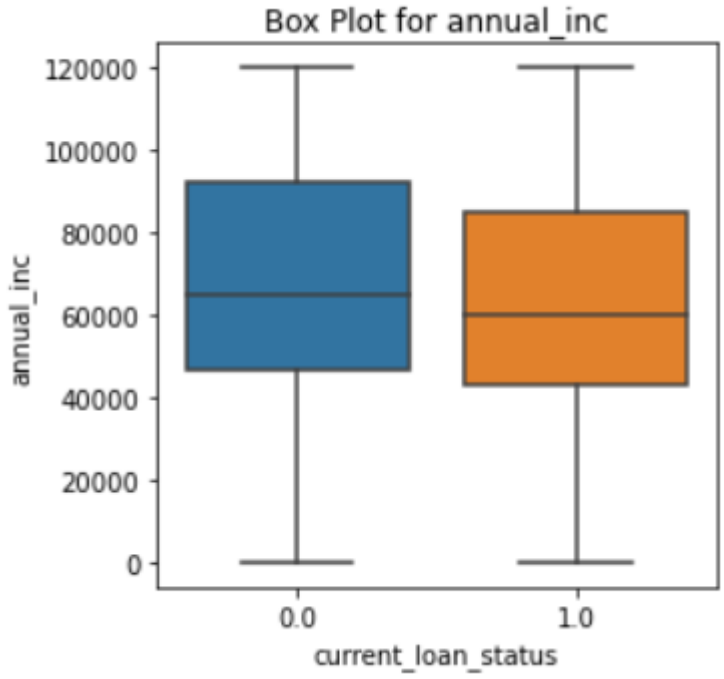
• **'verification_status'**: We have three categories: “Source Verified,” “Verified,” and “Not Verified.” The count plot displays the distribution of different verification statuses within the dataset. The tallest bar represents the “Source Verified” category, indicating that the highest number of records fall into this group. As for the “Verified” category has a moderate frequency, and the “Not Verified” category has the lowest frequency. Organisations or analysts can use this information to understand how many records are verified versus not verified.

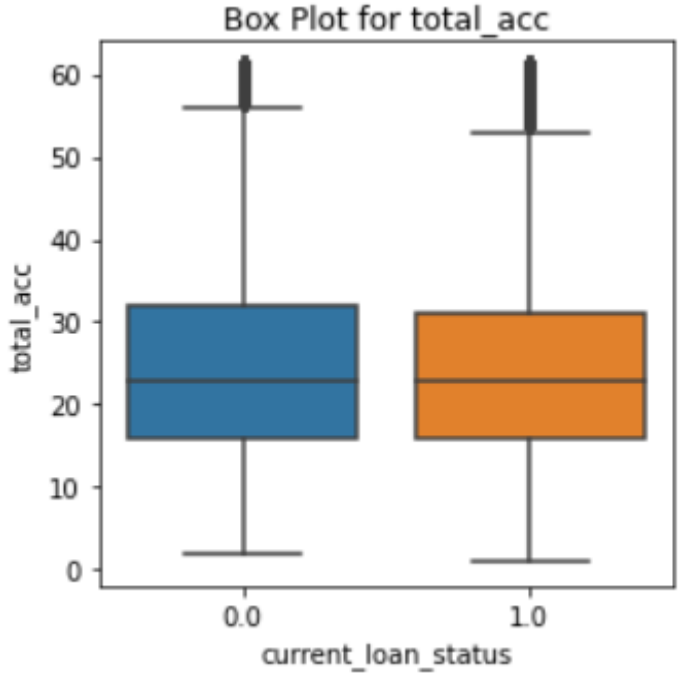
2. Bivariate Analysis

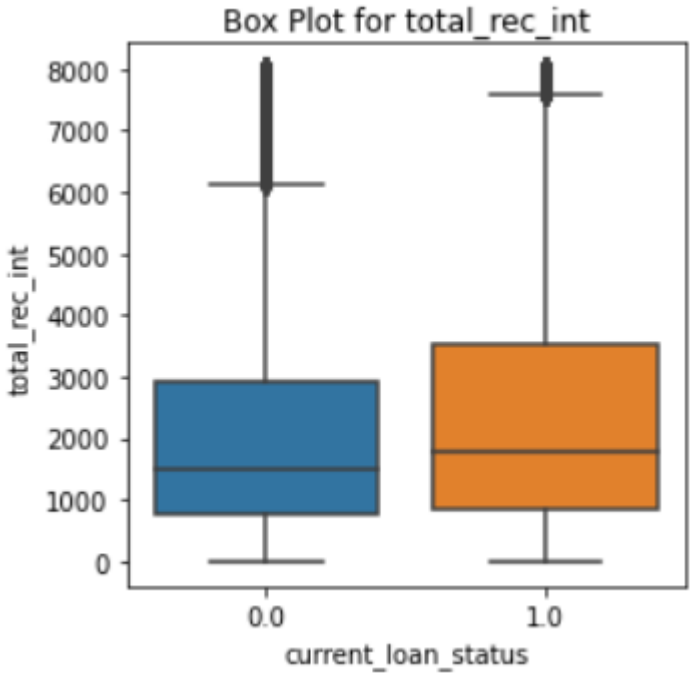
2.1. Categorical vs Numerical




Box Plot for installment
installment
1200
1000
800
600
400
200
0
0.0
1.0
current_loan_status


Box Plot for annual_inc
annual_inc
120000
100000
80000
60000
40000
20000
0
0.0
1.0
current_loan_status





The box plot shows the relationship between the interest rate (`**int_rate**`) and the current loan status (`**current_loan_status**`). It indicates that loans with a status of 1 (presumably current) have a higher median interest rate compared to those with a status of 0 (presumably not current).

The box plot illustrates the relationship between the instalment amount and the current loan status. It shows that both current and non-current loans have similar median instalment amounts, with a slightly higher median for current loans.

The box plot shows the distribution of annual income (`**annual_inc**`) for different loan statuses. It indicates that the median annual income for borrowers with a current loan status (1) is slightly lower than for those with a non-current loan status (0).

The first box plot shows the distribution of **total_acc** (total number of credit accounts) for both loan status categories (0 and 1). The medians and interquartile ranges are similar, suggesting no significant difference in total accounts between the two loan status groups.

The second box plot shows the distribution of **total_rec_int** (total received interest) for both loan status categories. Here, the medians and interquartile ranges indicate that the total received interest is slightly higher for the loan status category 1 compared to category 0.

6. ENCODING

```
category_df_dummies=pd.get_dummies(data=category_df,columns=class_col,drop_first=True)
```

```
category_df_dummies.head()
```

	term_ 60 months	grade_B	grade_C	grade_D	grade_E	grade_F	grade_G	emp_length_between 5-9 years	emp_length_less than 5 years	home_ownership_OTHERS	home_ownership_OWN
0	0	0	0	0	0	0	0	1	0	0	0
1	1	0	0	0	1	0	0	1	0	0	0
2	0	0	0	0	1	0	0	1	0	0	0
3	1	0	0	1	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	1	0	0

One-hot encoding is preferred over other encoding methods due to its ability to maintain the independence of categorical variables, avoiding assumptions of ordinal relationships. It eliminates bias by representing each category equally, preventing unintended model influences. Compatibility with various machine learning algorithms is a key advantage, as the binary matrix integrates seamlessly with numerical input requirements. Particularly suitable for nominal data, one-hot encoding accurately reflects categorical distinctions without implying false numerical relationships. The resulting binary matrix enhances interpretability, providing clear insights into category presence for each observation. Its explicit representation of missing data ensures consideration during model training. One-hot encoding is scalable to numerous categories, accommodating datasets with extensive variable diversity. By avoiding arithmetic operations on encoded values, it prevents unintentional mathematical relationships. Compatibility with distance metrics facilitates accurate representation of dissimilarity in clustering or nearest neighbors algorithms. As a widely accepted

standard, one-hot encoding stands as a reliable and effective approach in machine learning for handling categorical variables.

7. MODEL BUILDING

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Function to calculate VIF
def calculate_vif(df):
    vif = pd.DataFrame()
    vif["Variable"] = df.columns
    vif["VIF"] = [variance_inflation_factor(df.values, i) for i in range(df.shape[1])]
    return vif
```

```
vif = calculate_vif(df)
```

```
high_vif_columns = vif[vif["VIF"] > 10]["Variable"]
df = df.drop(columns=high_vif_columns)
```

```
X=df.iloc[:,:77]
y=df.current_loan_status
```

```
y=y.astype(int)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=69)
print(f"Training data size: {X_train.shape}")
print(f"Testing data size: {X_test.shape}")
```

```
try:
    logreg = sm.Logit(y_train, X_train).fit(maxiter=100, method='newton')
    print(logreg.summary())
except Exception as e:
    print("Error occurred:", e)
```

```
y_pred = logreg.predict(X_test)
```

```
y_pred_i=y_pred.astype(int)
```

```
result_df=pd.DataFrame(columns=['accuracy','precision','recall','f1-score'])
```

```
row={'accuracy': accuracy_score(y_test,y_pred_i), 'precision': precision_score(y_test,y_pred_i),
     'recall': recall_score(y_test,y_pred_i),'f1-score':f1_score(y_test,y_pred_i)}
```

```
result_df=result_df.append(row,ignore_index=True)
```

```
result_df
```

```
print(metrics.confusion_matrix(y_test,y_pred_i))
```

There is high multicollinearity which is why we use The Variance Inflation Factor (VIF). The significance analysis reveals significant multicollinearity issues in the dataset, particularly among loan grade categories ('grade_C' , 'grade_D' , 'grade_E ') and regional variables ('region_Eastern', ‘region_Center’, ‘region_Pacific’, ‘int_rate').

We divided columns into the independent variable which is ‘x’ and the dependent variable into ‘y’.

This code splits the data into training and testing sets using the train_test_split function from scikit-learn. It splits the features (X) and the target variable (y) into training and testing sets, with 70% of the data used for training (X_train , y_train) and 30% for testing (X_test , y_test).

The logistic regression model attempted to fit with a maximum of 100 iterations using the Newton method. Maximum Likelihood optimization failed to converge . Significant predictors include term_60 months , grade_F , grade_G , home_ownership_RENT , and various employment lengths, with coefficients indicating their influence on the current_loan_status .

The model or iteration achieved a high precision of 1.0, indicating that when it predicted positive cases, it was always correct. However, it had a very low recall of 0.0425, suggesting it missed a large proportion of actual positive cases, resulting in an overall poor F1-score of 0.0815.

8. CONCLUSION

By following these structured steps, the Lending Club Loan Data project aims to create a robust predictive model for loan default risk. This model helps Lending Club make more informed decisions, optimize loan approval processes, and minimize financial losses. Through data-driven insights, the project enhances the overall effectiveness of lending operations, benefiting both borrowers and investors.

The logistic regression model we used to predict whether someone will pay back a loan didn't work perfectly. The precision was good though, which means when it said someone would pay back, it was almost always right. But it missed a lot of people who actually would pay back, showing low recall. Sensitivity was high, meaning it caught a lot of people who would pay back, but it also flagged many who wouldn't. Specificity was decent, indicating it could identify those who wouldn't pay back quite well. Youden's Index, which combines sensitivity and specificity, showed the model was moderately effective in its overall prediction.

To improve the model, we could try tweaking it to converge better, maybe by adjusting the way it learns from the data. We might also want to simplify it or check if some of the factors we're considering are too similar, which could confuse the model. Standardizing the data, which means putting everything on the same scale, could also help. Additionally, we could try using a different method to see if it works better. These changes might make the model more accurate and reliable in predicting loan repayment behaviour.