

Module 8: LENDING CLUB CASE STUDY



Group Submission By:
Ankita Jindal (batch ID: 5811)
Ankur Parashar (batch ID: 5811)

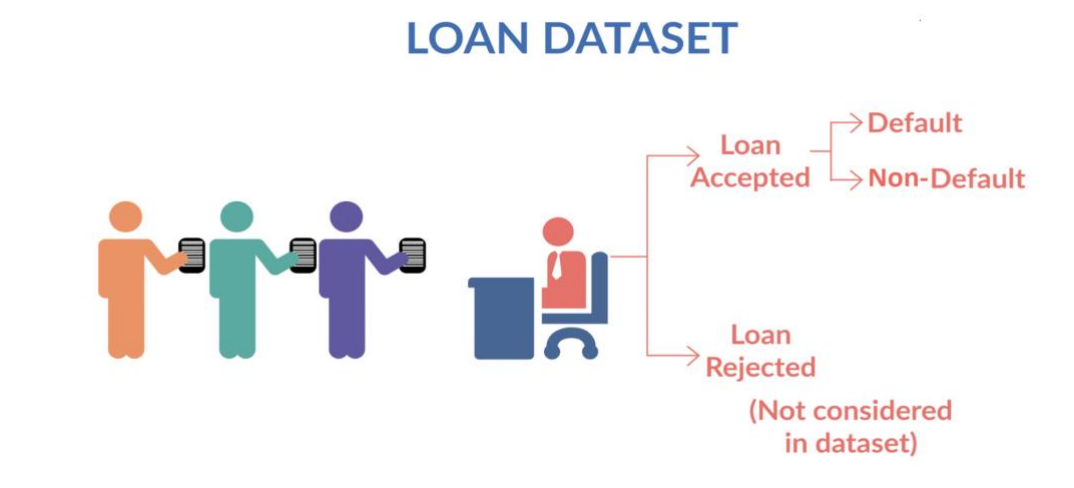
Problem Statement

We work for a consumer finance company which specialises in lending various types of loans to urban customers. When the company receives a loan application, the company has to make a decision for loan approval based on the applicant's profile.

Dataset

The dataset contains information about past loan applicants and whether they 'defaulted' or not. The column giving this information is 'loan_status' and it has three possible values : "Charged Off" (defaulted loans), "Fully Paid" and "Current". Along with this, there are 110 other attributes pertaining to the loan application like : loan_amnt, term, int_rate, installment, grade, annual_inc etc.

We need to analyse the dataset using the EDA (Exploratory Data Analysis) techniques and identify some patterns that may lead to a loan default.

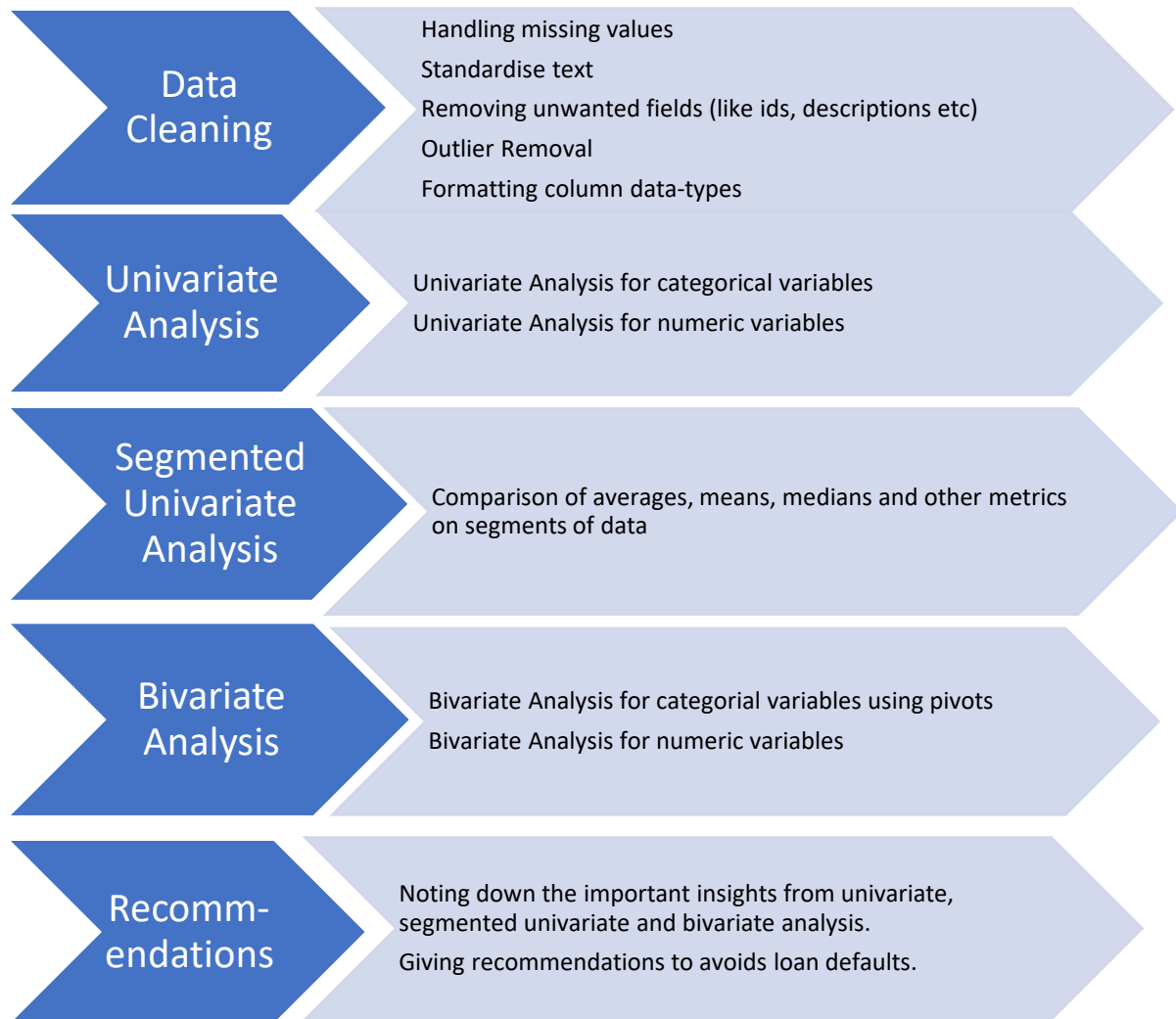


Business Objective

The aim of the case study is to identify the risky loan applicants, thereby cutting down the amount of credit loss for the organisation. The company wants to understand the driving factors (or driver variables) behind loan default, i.e. the variables which are strong indicators of default. The company can utilise this knowledge for its portfolio and risk assessment.

Analysis Approach

The analysis for the lending club case study involves the following steps:



Using the analysis approach given above, we have analyzed the data and tried to come up with some features/combination of features which can be helpful in identifying the borrowers who are most likely to default on their loans.

Data Cleaning

Removal of columns with all nulls: Out of the 111 columns, there were 54 columns which only had null data. We identified and removed those columns using the code below:

```
#Checking the columns with nulls and extracting columns which only have nulls

loanData_nulls=loanData.isnull().sum()
loanData_all_nulls=loanData_nulls[loanData_nulls==len(loanData)]
len_loanData_nulls=len(loanData_nulls)
len_loanData_all_nulls=len(loanData_nulls[loanData_nulls==len(loanData)])

print("Total columns with nulls : ",len_loanData_nulls)
print("Total columns with all nulls : ",len_loanData_all_nulls)

non_null_columns=list(set(loanData.columns) - set(loanData_all_nulls.index))
loanData=loanData[non_null_columns]
```

```
Total columns with nulls : 111
Total columns with all nulls : 54
```

Handling missing values : Checked the columns which had some missing values in them and analysed them using the code snippet below:

```
# Checking columns which have some null values in them
loanData.isnull().sum()[loanData.isnull().sum()>0]
```

```
mths_since_last_record    36931
collections_12_mths_ex_med    56
mths_since_last_delinq    25682
last_pymnt_d              71
emp_length                1075
tax_liens                  39
revol_util                 50
desc                     12940
emp_title                 2459
last_credit_pull_d         2
chargeoff_within_12_mths    56
next_pymnt_d              38577
pub_rec_bankruptcies        697
title                     11
dtype: int64
```

Data Cleaning

After checking the value_counts() for all these columns, below decisions were taken:

- 1) **next_pymnt_d**
(Next scheduled payment date)
Removed this column as it had 95pc nulls and is not of much business importance
- 2) **mths_since_last_record**
(The number of months since the last public record)
Removed this column as it had 90pc nulls and is not of much business importance.
- 3) **Id and member_id** : Removed these columns as these are just identifiers.
- 4) **collections_12_mths_ex_med**
(Number of collections in 12 months excluding medical collections)
Removed this column as it has only zeros
- 5) **mths_since_last_delinq**
(The number of months since the borrower's last delinquency)
Replaced nulls with zeros as zero as that would indicate no known last delinquency of the borrower.
- 6) **emp_length**
(Employment Length)
Filled nulls with "0 years" as that would indicate no employment. Such people may be small business owner or people having rental income etc.
- 7) **tax_liens**
(Number of tax liens)
Removed "tax_liens" as it has only one value (0)
- 8) **revol_util**
(Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit)
Filled nulls with "0%" as that would mean no utilization and is making more business sense
- 9) **desc, emp_title, title, zip_code, url, addr_state**
These columns can be removed as they do not hold much significance in the analysis and are only some descriptions of the borrower
- 10) **chargeoff_within_12_mths**
(Number of charge-offs within 12 months)
Removed it as it has only one value (0)
- 11) **pub_rec_bankruptcies**
(Number of public record bankruptcies)
Filled nulls with 0 as that would mean no publicly recorded bankruptcies.
- 12) **Removed Single Value Columns**
pymnt_plan, policy_code, initial_list_status, application_type

Data Cleaning

Standardized text for the categorical columns

Text for the categorical columns was standardised to unify the data and to be able to group the data properly when required. The code snippet below was used to lower the text and remove leading and trailing spaces.

```
▶ 05:26 PM (<1s) 25 Python
# getting the list of non-numeric columns and converting them to lower case and removing leading and trailing spaces
df_types=loanData.dtypes
list_object=list(df_types[df_types=="object"].index)
for x in list_object:
    loanData[x]=loanData[x].apply(lambda x: str(x).lower())
    loanData[x]=loanData[x].apply(lambda x: str(x).strip())
```

Standardized column formats

emp_length: Made it numeric for better analysis. Converted "10+" to 11 to distinguish them from 10 years. Converted "< 1" year to 0.5 to distinguish them from others.

int_rate: Made it numeric for better analysis and removed '%' sign.

```
▶ 05:26 PM (<1s) 21 Python
# emp_length(employment length) : Making it numeric for better analysis. Converting "10+" to 11 to distinguish then from 10 years. Converting "< 1" year to 0.5 to distinguish then from others.

loanData["emp_length"].value_counts()

loanData["emp_length"]=loanData["emp_length"].apply(lambda x: x[:-5])
loanData["emp_length"]=loanData["emp_length"].apply(lambda x: 0.5 if x=="< 1" else x)
loanData["emp_length"]=loanData["emp_length"].apply(lambda x: 11 if x=="10+ " else x)
loanData["emp_length"]=loanData["emp_length"].astype(int)

▶ 05:26 PM (<1s) 22 Python
# int_rate(rate Of Interest of the loan) : Making it numeric for better analysis and removing '%' sign
loanData["int_rate"].value_counts()
loanData["int_rate"]=loanData["int_rate"].apply(lambda x: float(x[:-1]))
```

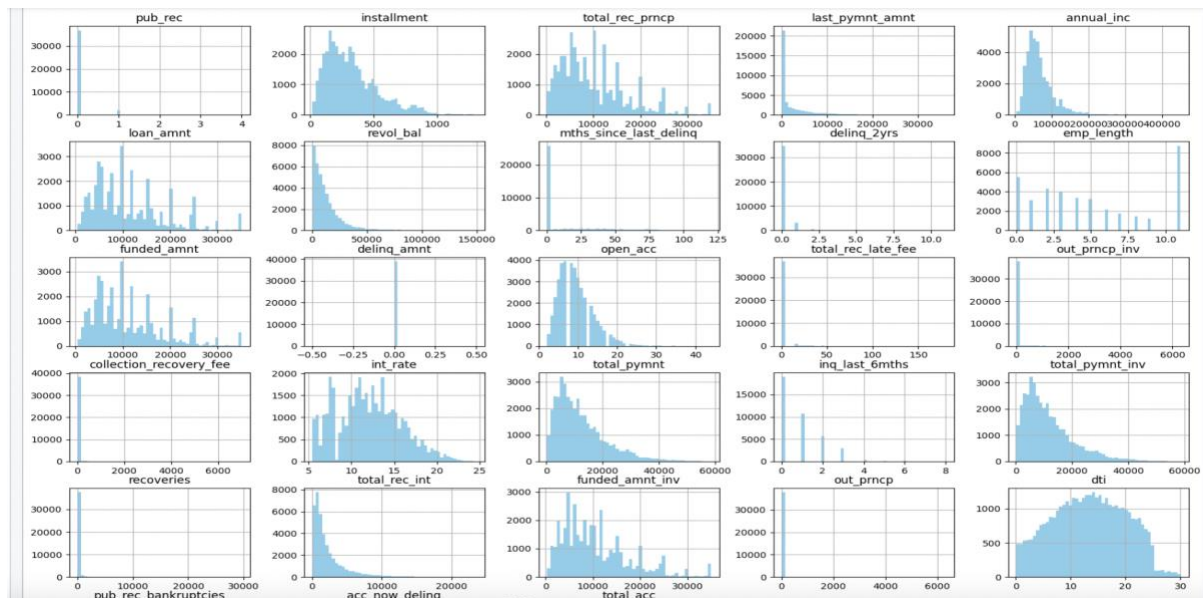
Outlier Removal

Removed outliers from the data using the code below. Used 0.99 as the upper quantile and .01 as lower quantile to remove only the extreme values. Around 1k records were removed by this activity.

Univariate Analysis

As a part of the univariate analysis, we employed techniques like creating histograms, scatter plots and box-plots to identify the trends in our data. This helped in finding some valuable insights to identify "Charged Off" loans.

Plotted histograms for checking data distribution of numeric columns

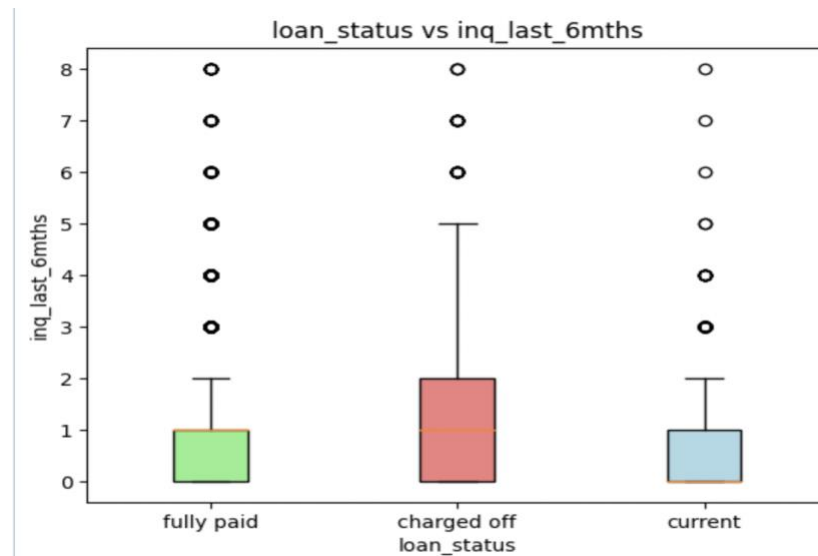


Observations

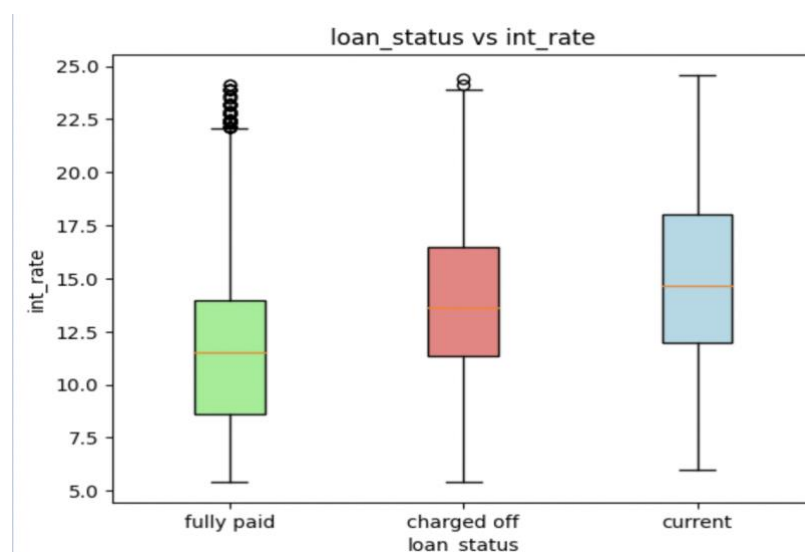
- **Grade Distribution:** The grade distribution reveals that a significant number of 'Charged Off' loans originate from the A, B, and C categories, suggesting a need to re-evaluate the company's grading criteria.
- **Popularity of Short-Term Loans:** Loans with a duration of 36 months are the most popular and show a higher likelihood of repayment.
- **2011 Charged Off Applications:** The year 2011 saw the highest number of 'Charged Off' loan applications, possibly due to economic or financial challenges during that period.
- **Verification Process Concerns:** The lending company has inadequately conducted the verification process, as most 'Charged Off' applicants are unverified.
- **8.Experience vs. Repayment Likelihood:** A high level of experience does not correlate with a greater likelihood of repayment; applicants with over 10 years of experience have the highest number of 'Charged Off' loans.

Plotted scatter plots and box plots for each feature against every loan status and comparing 'fully paid' and 'charged off'.

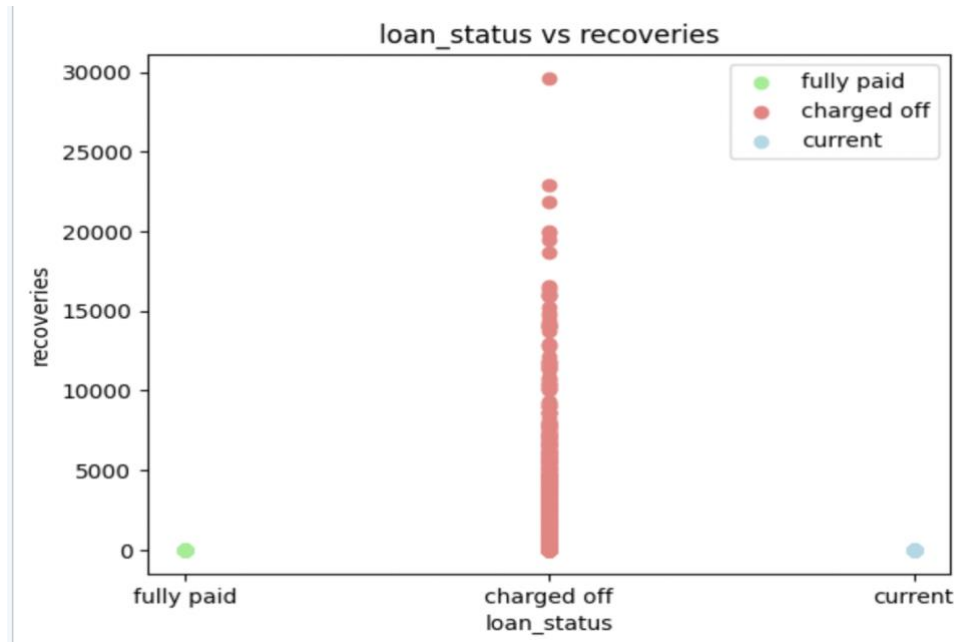
Finding 1. "inq_last_6mths" : Defaulters have higher number of inquiries 1-5 in the last 6 months whereas non-defaulters have mostly 0 or 1 inquiry. This can be useful in case we have history information for the customer or if the customer is already in the process of loan payment.



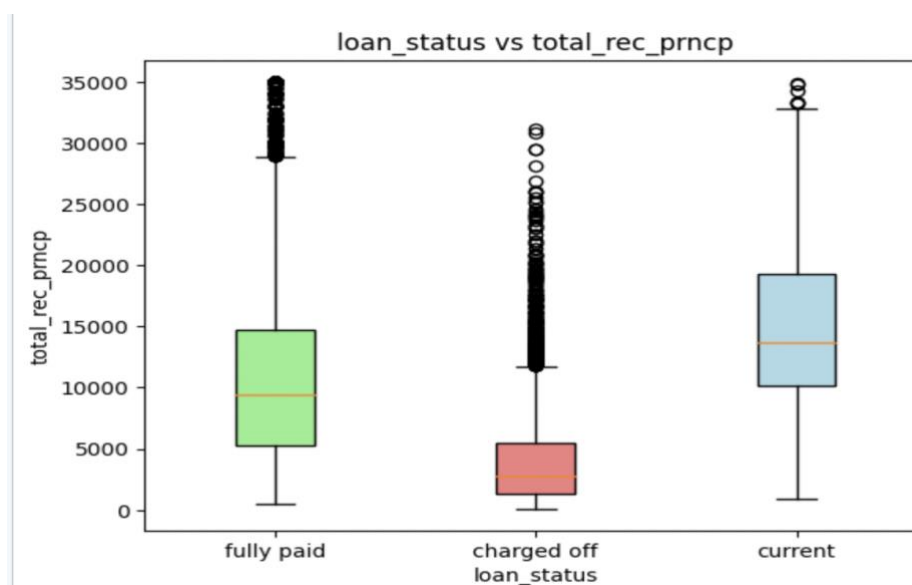
Finding 2. "int_rate" : It is also observed that defaulters have a relatively higher interest rate in the range of 11.5 to 16.5 whereas non-defaulters have lower interest rates ranging from 6 to 13.5 .This would mean that higher interest rates are more difficult to be paid off.



Finding 3. "recoveries" : post charge off gross recovery is much higher in the range of 100 to 16k for defaulters whereas it is almost 0 for non-defaulters. This can be useful in case we have history information for the customer or if the customer is already in the process of loan payment.

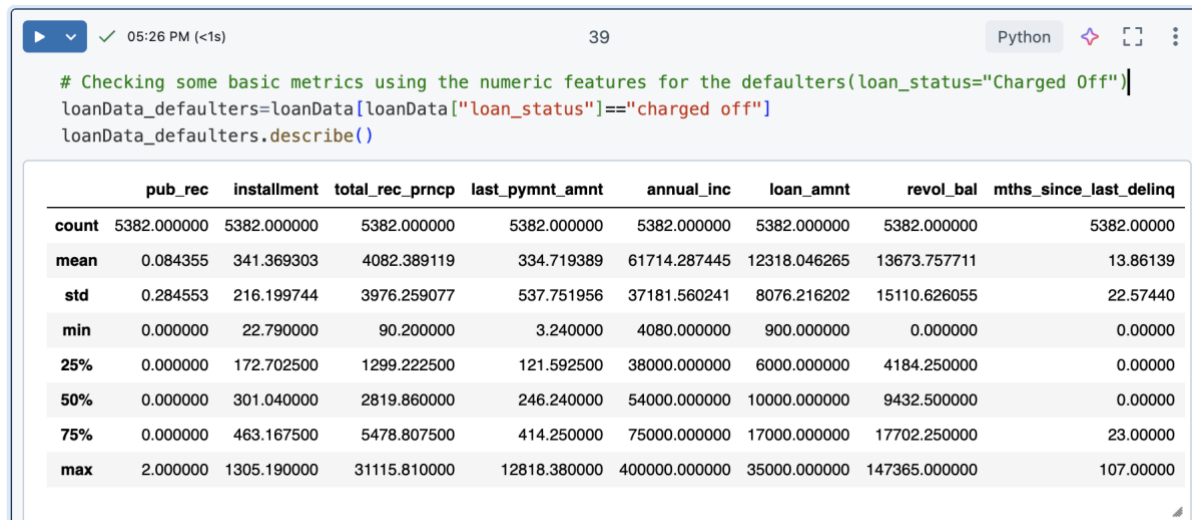


Finding 4. "total_rec_prncp" : Principal received to date for the loan defaulters is much less as compared to the non-defaulters. For defaulters it lies in the range of 1k to 5k whereas it lies in the range 5k to 14k for non-defaulters. This can be useful in case we have history information for the customer or if the customer is already in the process of loan payment.



Segmented Univariate Analysis

Checked some basic metrics using the numeric features for the defaulters(loan_status="Charged Off")



The screenshot shows a Jupyter Notebook interface. At the top, there's a status bar with a play button, a checkmark, the time '05:26 PM (<1s)', the cell number '39', and the language 'Python'. Below this, a code cell contains the following Python code:

```
# Checking some basic metrics using the numeric features for the defaulters(loan_status="Charged Off")
loanData_defaulters=loanData[loanData["loan_status"]=="charged off"]
loanData_defaulters.describe()
```

Below the code, a summary statistics table is displayed. The table has 9 columns: 'count', 'pub_rec', 'installment', 'total_rec_prncp', 'last_pymnt_amnt', 'annual_inc', 'loan_amnt', 'revol_bal', and 'mths_since_last_delinq'. The rows represent various statistical measures: 'count', 'mean', 'std', 'min', '25%', '50%', '75%', and 'max'.

	pub_rec	installment	total_rec_prncp	last_pymnt_amnt	annual_inc	loan_amnt	revol_bal	mths_since_last_delinq
count	5382.000000	5382.000000	5382.000000	5382.000000	5382.000000	5382.000000	5382.000000	5382.000000
mean	0.084355	341.369303	4082.389119	334.719389	61714.287445	12318.046265	13673.757711	13.86139
std	0.284553	216.199744	3976.259077	537.751956	37181.560241	8076.216202	15110.626055	22.57440
min	0.000000	22.790000	90.200000	3.240000	4080.000000	900.000000	0.000000	0.000000
25%	0.000000	172.702500	1299.222500	121.592500	38000.000000	6000.000000	4184.250000	0.000000
50%	0.000000	301.040000	2819.860000	246.240000	54000.000000	10000.000000	9432.500000	0.000000
75%	0.000000	463.167500	5478.807500	414.250000	75000.000000	17000.000000	17702.250000	23.000000
max	2.000000	1305.190000	31115.810000	12818.380000	400000.000000	35000.000000	147365.000000	107.000000

Did segmented analysis for all the 'loan_status' values against 'term' and 'home_ownership' and normalized the results to understand the percentage of each category in each segment



The screenshot shows a Jupyter Notebook interface. Below the code cell, a pivot table is displayed. The table has 8 columns: 'loan_status', 'home_ownership', 'other', 'own', 'rent', 'term', '36 months', and '60 months'. The rows represent the 'loan_status' categories: 'charged off', 'current', and 'fully paid'. The 'home_ownership' categories are 'mortgage', 'none', and 'other'. The 'term' categories are '36 months' and '60 months'.

loan_status	home_ownership			term			
	mortgage	none	other	own	rent	36 months	60 months
charged off	0.417689	0.000000	0.003159	0.079153	0.500000	0.56466	0.43534
current	0.558304	0.000000	0.000000	0.073322	0.368375	0.00000	1.00000
fully paid	0.446429	0.000093	0.002221	0.076487	0.474772	0.78371	0.21629

Finding 5.

Most of the non-defaulters take loans for smaller periods (36 months) whereas defaulters take loans for a mix of smaller(36 months) and longer periods (60 months).

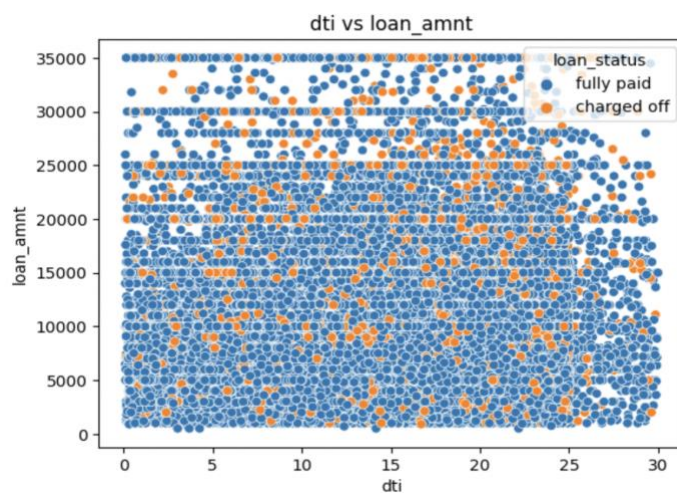
Hence we can say that "36 months" loans are more preferred if we need to reduce the number of defaults.

Bivariate Analysis

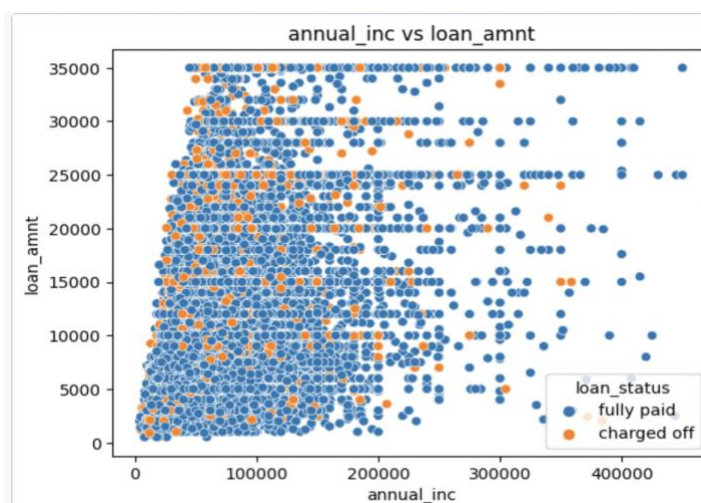
As a part of the bivariate analysis, we created bivariate scatter plots against the loan_status to help identify the combination of features that could identify "Charged Off" loans.

Bivariate analysis for numeric features

Finding 6. As seen in the graph of dti vs loan_amount, we can see that a combination of higher values of both leads to more loan defaults. Hence a person with higher dti(debt to income ratio) in the range of 15+ must not be given higher value loans(above 2lakhs).

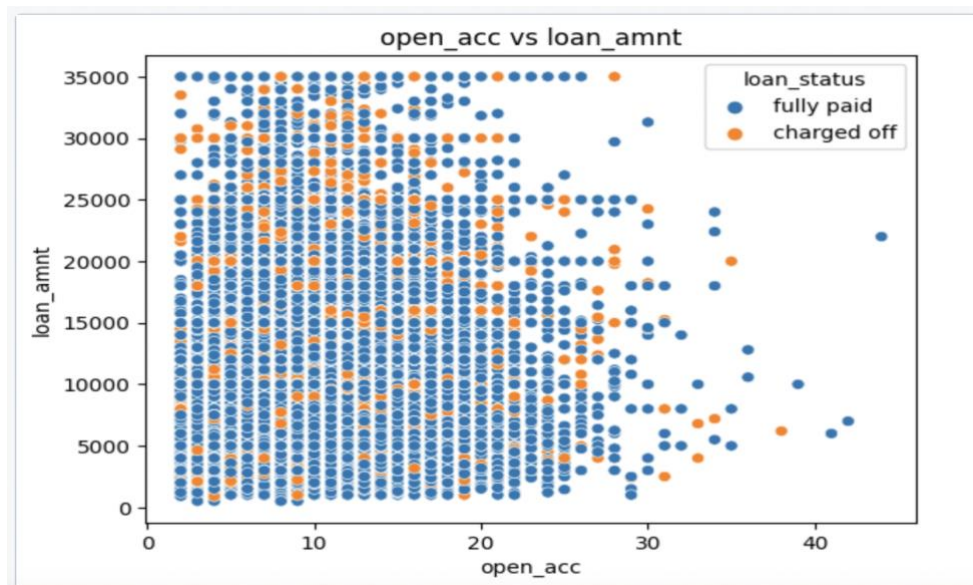


Finding 7. As seen in the graph of annual_inc vs loan_amount, we can see that a combination of high loan_amnt and low annual_inc, leads to higher loan defaults. . Hence a person with low annual_income(less than 1lakh) must not be given higher value loans(2 lakh+).

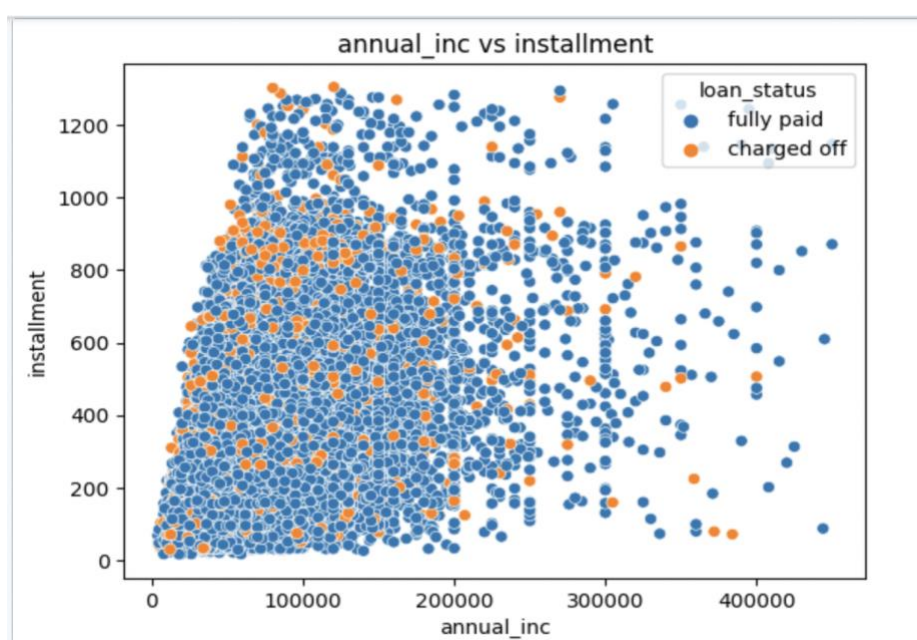


Finding 8. As seen in the graph of open_acc vs loan_amnt, we can see that a combination of high loan_amnt(20k +) and open_acc range of 7-20, leads to higher loan defaults.

Hence a person with higher open_acc(The number of open credit lines in the borrower's credit file) must not be given higher value loans.



Finding 9. As seen in the graph installment vs annual_inc, we can see that a combination of low annual_inc(less that 1.3 LPA) and higher installment(800 -1000) leads to higher defaults. Hence a person low annual_income must not be given loans with higher installments.



Bivariate analysis for categorical features

Checked the effect of 'home_ownership' on other features

	loan_status			purpose						
	charged off	current	fully paid	car	credit_card	debt_consolidation	educational	home_improvement	house	n
home_ownership										
mortgage	0.129530	0.036416	0.834054	0.042927	0.129127	0.445174	0.004955	0.132411	0.008124	
none	0.000000	0.000000	1.000000	0.000000	0.000000	0.333333	0.000000	0.000000	0.000000	
other	0.191011	0.000000	0.808989	0.022472	0.112360	0.438202	0.022472	0.056180	0.011236	
own	0.142523	0.027768	0.829709	0.050519	0.097022	0.432586	0.005688	0.107728	0.010371	
rent	0.145444	0.022538	0.832018	0.033510	0.136580	0.506324	0.010377	0.013782	0.010323	

Checked the effect of 'term' on other features by creating a pivot table.

Finding 9. '36 months' loans have a higher chance of getting fully paid off as compared to '60 months' term loans

```
# Bivariate analysis for the categorial columns. Checking the effect of 'term' on other features
loanData_cols=loanData[['loan_status', 'term', 'home_ownership', 'purpose', 'verification_status']]

loanData_pivot=(loanData_cols.set_index('term').stack()
                .groupby(level=[0,1])
                .value_counts(normalize=True)
                .unstack(level=[1,2])
                .fillna(0)
                .sort_index(axis=1))

loanData_pivot
```

	home_ownership					loan_status			purpose			
	mortgage	none	other	own	rent	charged off	current	fully paid	car	credit_card	debt_consolidation	e
term												
36 months	0.413146	0.000105	0.003093	0.078805	0.504851	0.106819	0.000000	0.893181	0.033814	0.141757	0.455466	
60 months	0.534039	0.000000	0.000095	0.071224	0.394641	0.223398	0.107933	0.668669	0.053013	0.098684	0.521453	

Recommendations

In-case we do-not have the loan history of the loan applicant:

- **dti** : A person with higher dti (debt to income ratio) in the range of 15+ must not be given higher value loans(above 2lakhs)
- **annual_income** : A person with low annual_income(less than 1lakh) must not be given higher value loans(2 lakh+)
- **term** : '36 months' loans have a higher chance of getting fully paid off as compared to '60 months' term loans
- **int_rate** : Higher interest rates(11.5% and above) are more difficult to be paid off as compared to lower interest rates(6% to 12%)
- **installment** : A person low annual_income(less than 1.3 LPA) must not be given loans with higher instalments(800 +)

In-case we have the loan history of the applicant, we can also use the below findings in addition to the above ones:

- **"inq_last_6mths"** : Defaulters have higher number of inquiries 1-5 in the last 6 months whereas non-defaulters have mostly 0 or 1 inquiry.
- **"recoveries"** : Post charge off gross recovery is much higher in the range of 100 to 16k for defaulters whereas it is almost 0 for non-defaulters.
- **"total_rec_prncp"** : Principal received to date for the loan defaulters is much less as compared to the non-defaulters. For defaulters it lies in the range of 1k to 5k whereas it lies in the range 5k to 14k for non-defaulters.
- **"open_acc"** : A person with higher open_acc(The number of open credit lines in the borrower's credit file) in the range of 7-20 must not be given higher value loans(2.5 lakh +)

Steps to identify the defaulters

- Check the input loan application against all the above mentioned recommendations.
- Mark the application as potential defaulter or non-defaulter against each recommendation
- Take majority voting and that could be our prediction for potential defaulter or potential non-defaulter.