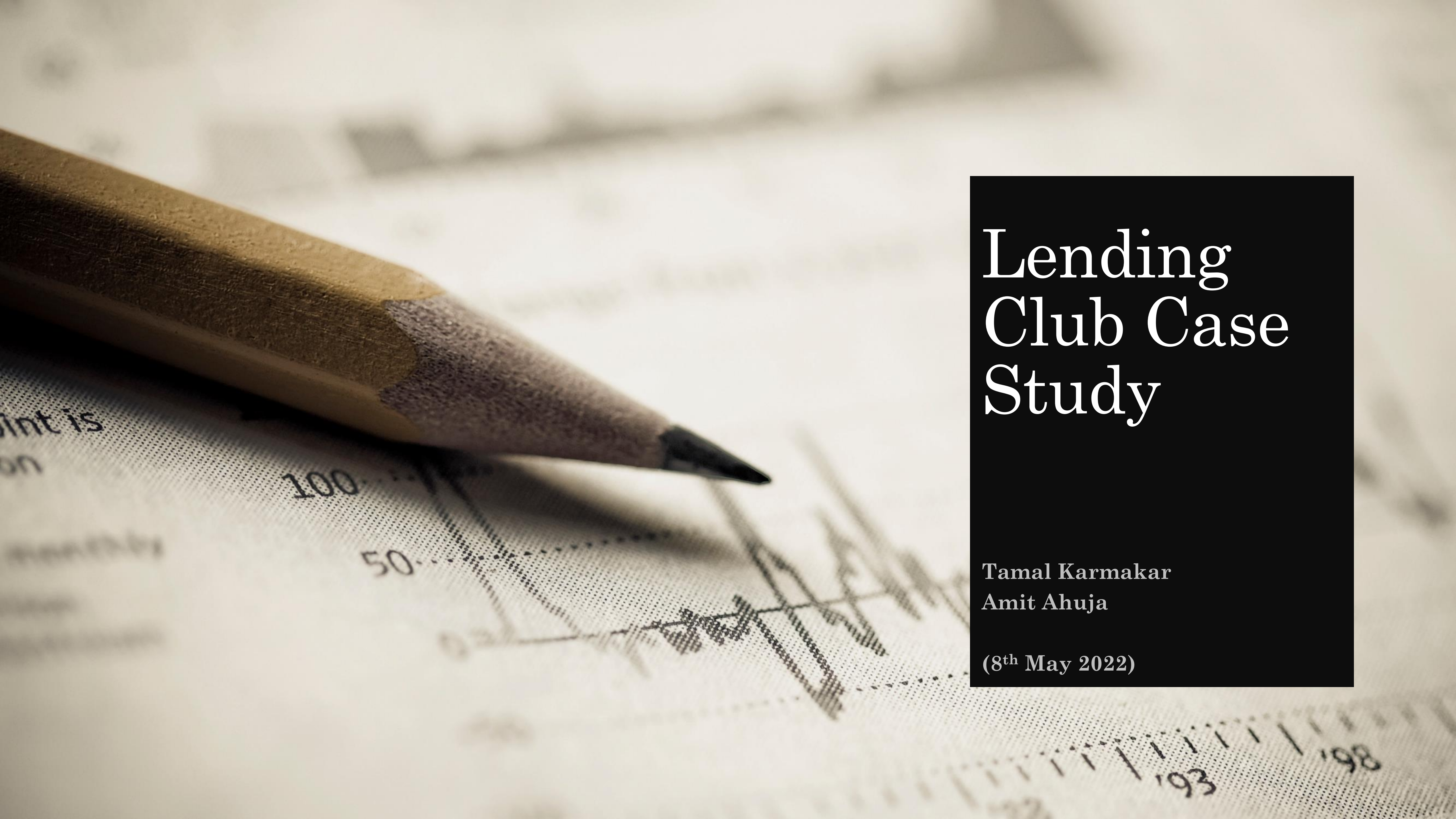


Lending Club Case Study

Tamal Karmakar
Amit Ahuja

(8th May 2022)

Table of Content

Case Study – Abridgment

Problem Statement:

Consumer finance company specializes 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 applicants profile. Two **types of risks** are associated with the banks decision:

- If the applicant is **likely to repay the loan**, then not approving the loan results in a **loss of business** to the company
- If the applicant is **not likely to repay the loan,** i.e. he/she is likely to default, then approving the loan may lead to a **financial loss** for the company

To do or the overall activity:

Leading Club has provided the data from 2007 to 2011, which contains a huge amount of data of various applications. So, as a part of our job before we derive any meaningful data that will be presented to Lending club management would be to

- Understand the data provided.
- Identify the attributes/features in the data which will be used for analysis.
- Cleanse the identified Data by removing the duplicate, erroneous or unwanted data.
- Do various analysis like Univariate, Bivariate, Derived and build/identify patterns on the plot built post clean-up.

Business Objective / Recommendations / Outcome:

This analysis will be used for taking actions:

- Denying the loan for the people fitting into the category of identified "defaulters"
- Reducing the amount of loan
- Lending (to risky applicants) at a higher interest rate.
- Or may be other uses.

Understanding and Analysis via Python

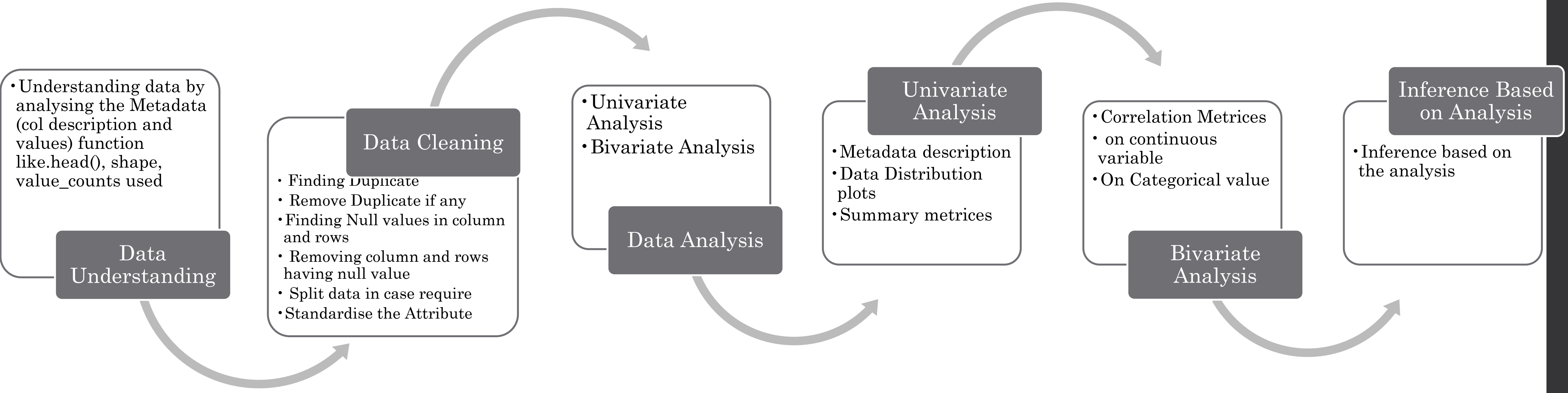
Important Libraries that were used:

- **numpy** – one of the most important and basic file to do all number/data analysis
- **pandas** – To play around data, this is library which facilitate everything.
- **matplotlib.pyplot** – For plotting of data
- **seaborn** – Library which give amazing graphs/charts
- **Warning** – is used to stop showing warning messages

Important Graphs Used:

- Bar
- Pie
- Clustermap
- Barplot
- Boxplot
- Heatmap
- lineplot

Process



Data Understanding

- Understanding data by analysing the Metadata (col description and values) function like.head(), shape, value_counts used

Data Cleaning

- Finding Duplicate
- Remove Duplicate if any
- Finding Null values in column and rows
- Removing column and rows having null value
- Split data in case require
- Standardise the Attribute

Data Analysis

- Univariate Analysis
- Bivariate Analysis

Univariate Analysis

- Metadata description
- Data Distribution plots
- Summary metrices

Bivariate Analysis

- Correlation Metrices
- on continuous variable
- On Categorical value

Inference Based on Analysis

- Inference based on the analysis

Data Understanding (Assumptions and Appendix)

As a part of Data Understanding, there are various assumptions that we need to consider and do the data analysis. Below are few important assumptions that bring meaning to the analysis done by the data analyst.

Assumptions

The provided data is quite self explanatory and thus post review of all the attributes and post data analysis, we could confidently say that no assumptions were required to derive any of the analysis.

Appendix

1. Columns will be referred to as Attributes.
2. Two files has been referred/consumed for this analysis
    1. Data_Dictionary.xlsx
    2. Loan.csv

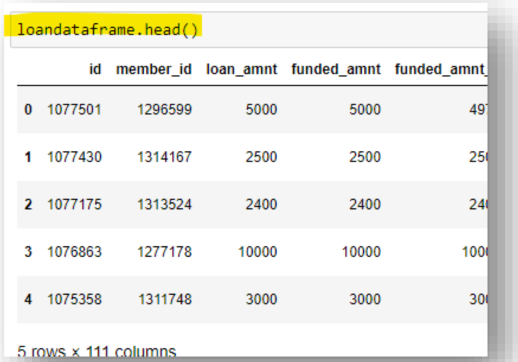
Data Understanding

Data analysis require *clear understanding* of data, for example, in this case, one should understand the volumetric of data being analyzed (higher the volume of data, probability of deriving meaning analysis would be high). Analyst should also identify those attributes which will be helpful, rather consuming unwanted data and deriving to over-complex and non-meaningful results.

Team has used different pandas methods and properties to analyze and understand the data. Naming few of the techniques/codes used.

- df.**shape** – to understand how many rows and columns the data have
- df.**head**() – to understand the columns header (Shows top 5 rows by default as seen in image)
- df.**dtypes** – shows datatype of each attribute
- df.**isnull().sum**() – to count null or empty value in all the columns of dataframe
- df.**columns** – to see the columns in the data
- df.**info**() – detailed information about the overall data
- df.<column>.**describe**() – basic athematic analytical data.

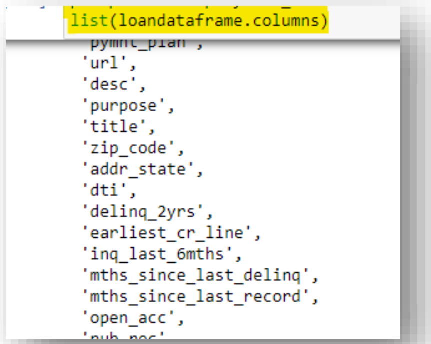
*df refers to data frame which is generated by csv load, in our case the dataframe name is "*loandataframe*"



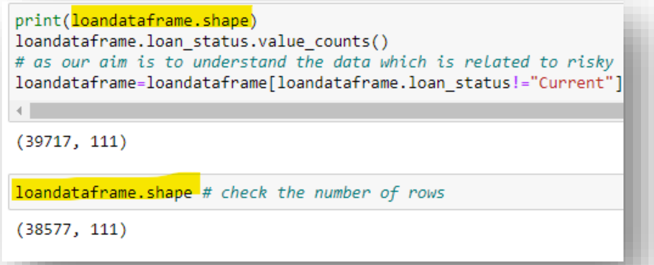
```
loandataframe.head()
```

	id	member_id	loan_amnt	funded_amnt	funded_amnt_
0	1077501	1296599	5000	5000	49
1	1077430	1314167	2500	2500	25
2	1077175	1313524	2400	2400	24
3	1076863	1277178	10000	10000	100
4	1075358	1311748	3000	3000	30

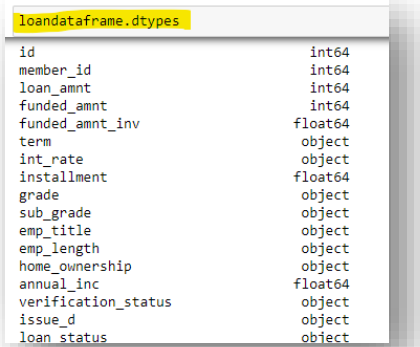
5 rows × 111 columns



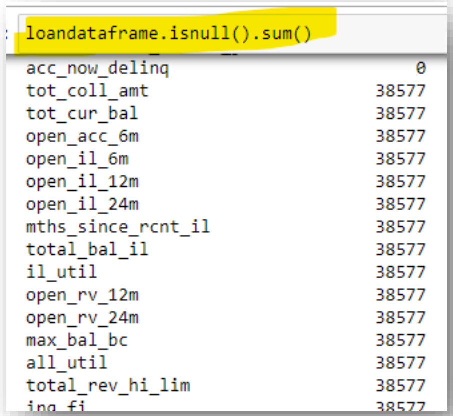
```
list(loandataframe.columns)
 'pymnt_plan',
 'url',
 'desc',
 'purpose',
 'title',
 'zip_code',
 'addr_state',
 'dti',
 'delinq_2yrs',
 'earliest_cr_line',
 'inq_last_6mths',
 'mths_since_last_delinq',
 'mths_since_last_record',
 'open_acc',
 'pub_rec',
```



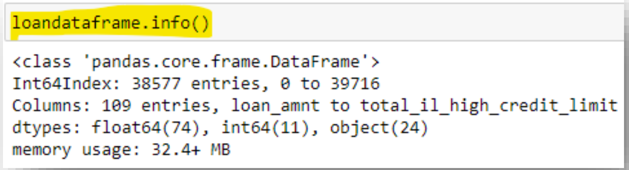
```
print(loandataframe.shape)
loandataframe.loan_status.value_counts()
# as our aim is to understand the data which is related to risky
loandataframe=loandataframe[loandataframe.loan_status!="Current"]

(39717, 111)

loandataframe.shape # check the number of rows

(38577, 111)
```



```
loandataframe.dtypes

id                          int64
member_id                   int64
loan_amnt                   int64
funded_amnt                 int64
funded_amnt_inv           float64
term                       object
int_rate                   object
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
```



```
loandataframe.isnull().sum()

acc_now_delinq                 0
tot_coll_amt               38577
tot_cur_bal                38577
open_acc_6m                38577
open_il_6m                 38577
open_il_12m                38577
open_il_24m                38577
mths_since_rcnt_il         38577
total_bal_il               38577
il_util                    38577
open_rv_12m                38577
open_rv_24m                38577
max_bal_bc                 38577
all_util                   38577
total_rev_hi_lim           38577
inq_fi                     38577
```

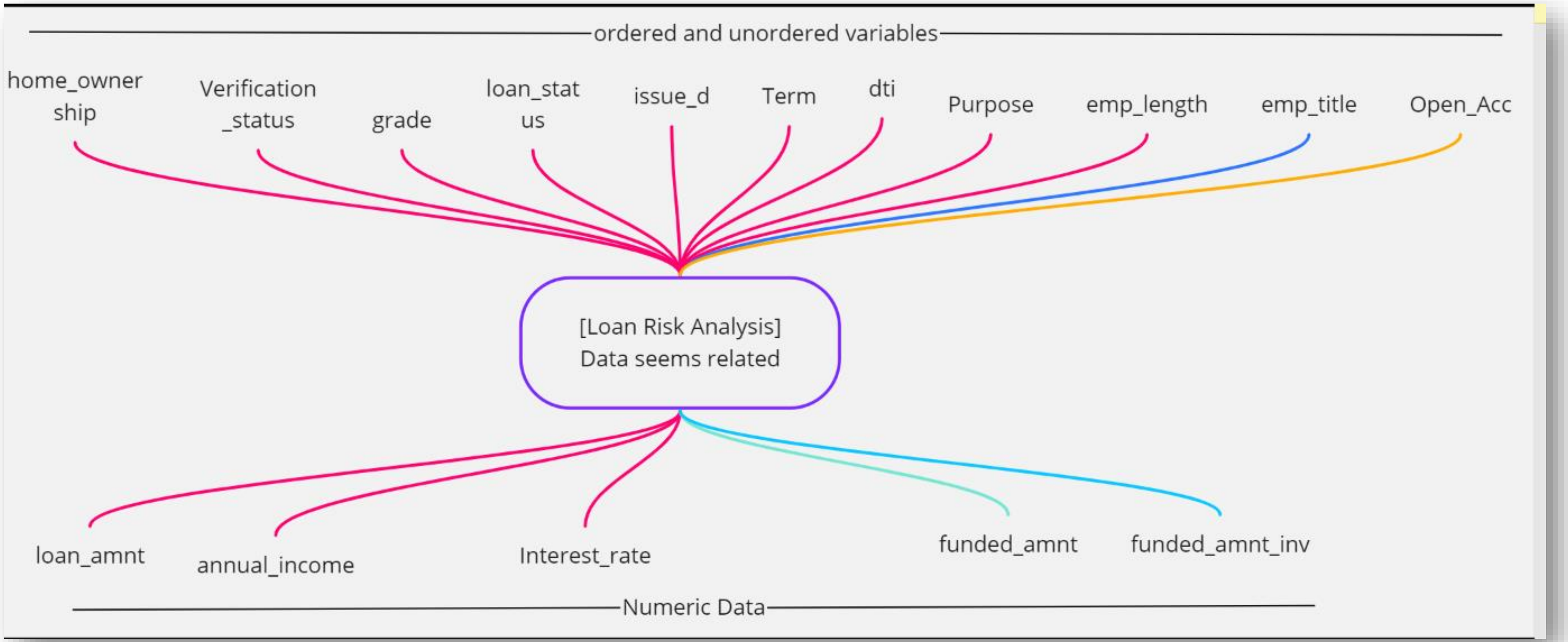


```
loandataframe.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 38577 entries, 0 to 39716
Columns: 109 entries, loan_amnt to total_il_high_credit_limit
dtypes: float64(74), int64(11), object(24)
memory usage: 32.4+ MB
```

Data Understanding (Quick Data Summary)

After understanding the source data provided by Lending Club, below are few details **(listing very limited information).**

1. **Original Source contains:** 39717 Rows and 111 attributes (Columns)
2. **Relevant Attributes**: 50 attributes contains the good/relevant data other 61 columns were without data (either contains 0 or NA or blank)
3. **Key attributes required for** analysis (below are most critical ones and not listing all):

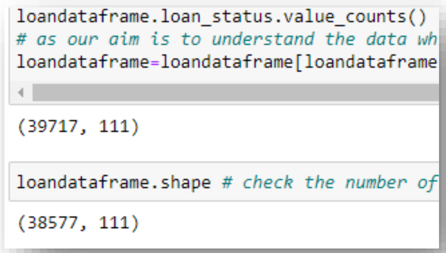


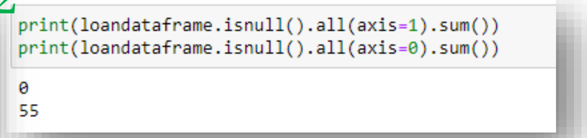
Data Cleansing (Data Standardization)

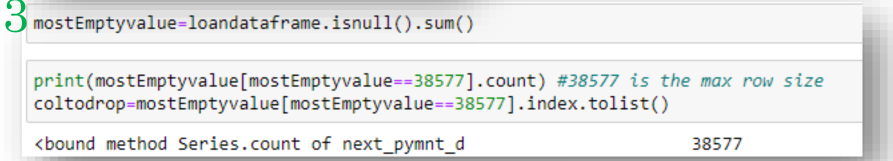
One of **the most important and very critical activity** performed by any analyst is Data cleaning. With cleaning of data, we make sure that whatever analysis that will be done by the analyst would be the most accurate, most meaningful, most important for the business; which helps them to take the required business/financial decisions. Any mistake that is done at the time of data cleaning may result to un-realistic or non-efficient data results, which might not be useful at all for any business.

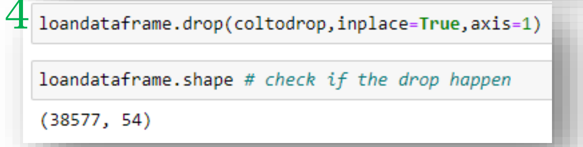
Basic things to look for data cleaning:

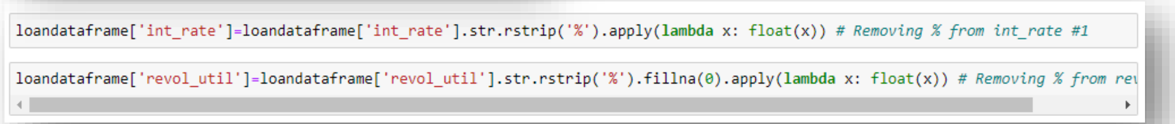
- Fix rows and columns
  - Remove all columns which having empty, NA & Null values, command used df.drop([‘col1’,’col2’…],inplace=True,axis=1)
  - Split issue_m attribute to issue_y & issue_m
  - Loan_status which have string value “Fully paid or “Charged off” is converted to binary 0 and 1.
- Fix missing values
  - emp_tile = empty values can be replaced with “notunknow” or can be removed altogether. However this specific column isn’t of much use as lot of data which doesn’t make much sense
  - Emp_length missing value is replaced with 0 value
- Standardize values
  - Rounding of the dataframe float value to 2 digits using .round(2) function
  - Remove % from int_rate, revol_util
  - Convert the emp_length from string to integer and replace the missing values
- Filter data
  - Check for duplicate data using .duplicate() method.
  - Create new columns which have data related to ‘Group the loan_amnt , interest_rate, annual_income and installment in bins
  - Remove the rows which have loan_status as current, as analysis around this data doesn’t have any inference as these are current loan data.

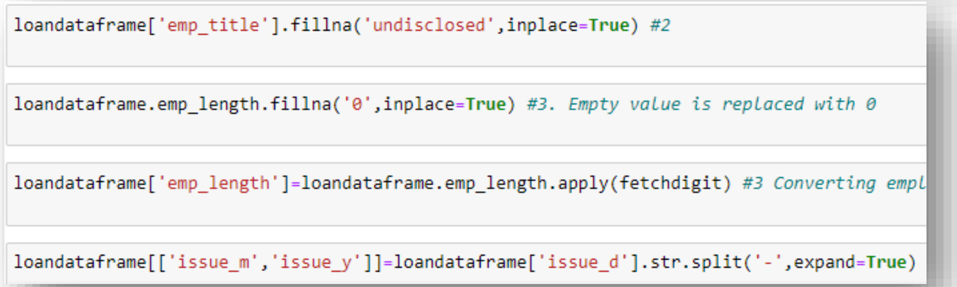
1

```
loandataframe.loan_status.value_counts()
# as our aim is to understand the data wh
loandataframe=loandataframe[loandataframe
```
```
(39717, 111)
```
```
loandataframe.shape # check the number of
```
```
(38577, 111)
```

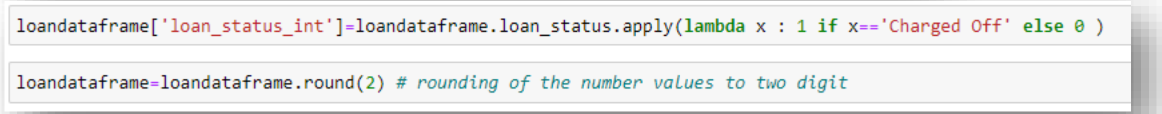
2

```
print(loandataframe.isnull().all(axis=1).sum())
print(loandataframe.isnull().all(axis=0).sum())
```
```
0
55
```

3

```
mostEmptyvalue=loandataframe.isnull().sum()
```
```
print(mostEmptyvalue[mostEmptyvalue==38577].count) #38577 is the max row size
coltodrop=mostEmptyvalue[mostEmptyvalue==38577].index.tolist()
```
```
<bound method Series.count of next_pymnt_d                    38577
```

4

```
loandataframe.drop(coltodrop,inplace=True,axis=1)
```
```
loandataframe.shape # check if the drop happen
```
```
(38577, 54)
```

5

```
loandataframe['int_rate']=loandataframe['int_rate'].str.rstrip('%').apply(lambda x: float(x)) # Removing % from int_rate #1
```
```
loandataframe['revol_util']=loandataframe['revol_util'].str.rstrip('%').fillna(0).apply(lambda x: float(x)) # Removing % from rev
```

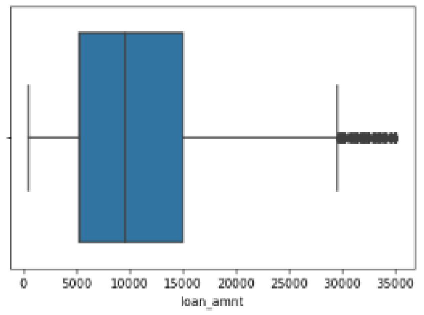
6

```
loandataframe['emp_title'].fillna('undisclosed',inplace=True) #2
```
```
loandataframe.emp_length.fillna('0',inplace=True) #3. Empty value is replaced with 0
```
```
loandataframe['emp_length']=loandataframe.emp_length.apply(fetchdigit) #3 Converting empl
```
```
loandataframe[['issue_m','issue_y']]=loandataframe['issue_d'].str.split('-',expand=True)
```

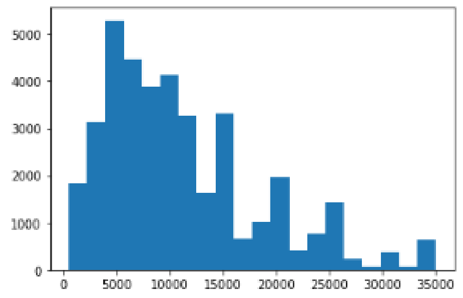
7

```
loandataframe['loan_status_int']=loandataframe.loan_status.apply(lambda x : 1 if x=='Charged Off' else 0 )
```
```
loandataframe=loandataframe.round(2) # rounding of the number values to two digit
```

Data Analysis (Univariate Use Case 1 – Loan Amount)

loan_amt

```
In [40]: loandataframe.loan_amnt.describe() # maximum loan amount requests as 35000 and most of the data belongs 5500 - 15000

Out[40]: count    38577.000000
         mean     11047.025430
         std       7348.441646
         min        500.000000
         25%       5300.000000
         50%       9600.000000
         75%      15000.000000
         max      35000.000000
         Name: loan_amnt, dtype: float64

In [41]: loandataframe.loan_amnt.value_counts(bins=[0,500,5500,10000,15000,25000,35000])

Out[41]: (5500.0, 10000.0]     12246
         (500.0, 5500.0]       10023
         (10000.0, 15000.0]     7683
         (15000.0, 25000.0]     7185
         (25000.0, 35000.0]     1435
         (-0.001, 500.0]           5
         Name: loan_amnt, dtype: int64

In [42]: sns.boxplot(loandataframe.loan_amnt)
         plt.show()
         plt.hist(loandataframe.loan_amnt,bins=20)
         plt.show()
```



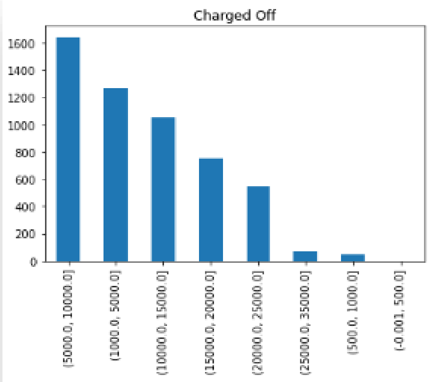


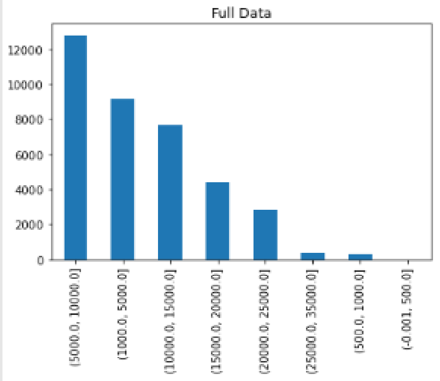
Analysis Explanation:

Attribute Loan Data: This attribute explains about the loan amount requested by any applicant.
Minimum loan Amount applied: 500
Maximum Loan Amount Applied for 35000
When categorized Loan Data into 8 such bins (post removal of outliners), shows that maximum loan that has been requested is between the range of 5000-15000.

Inference

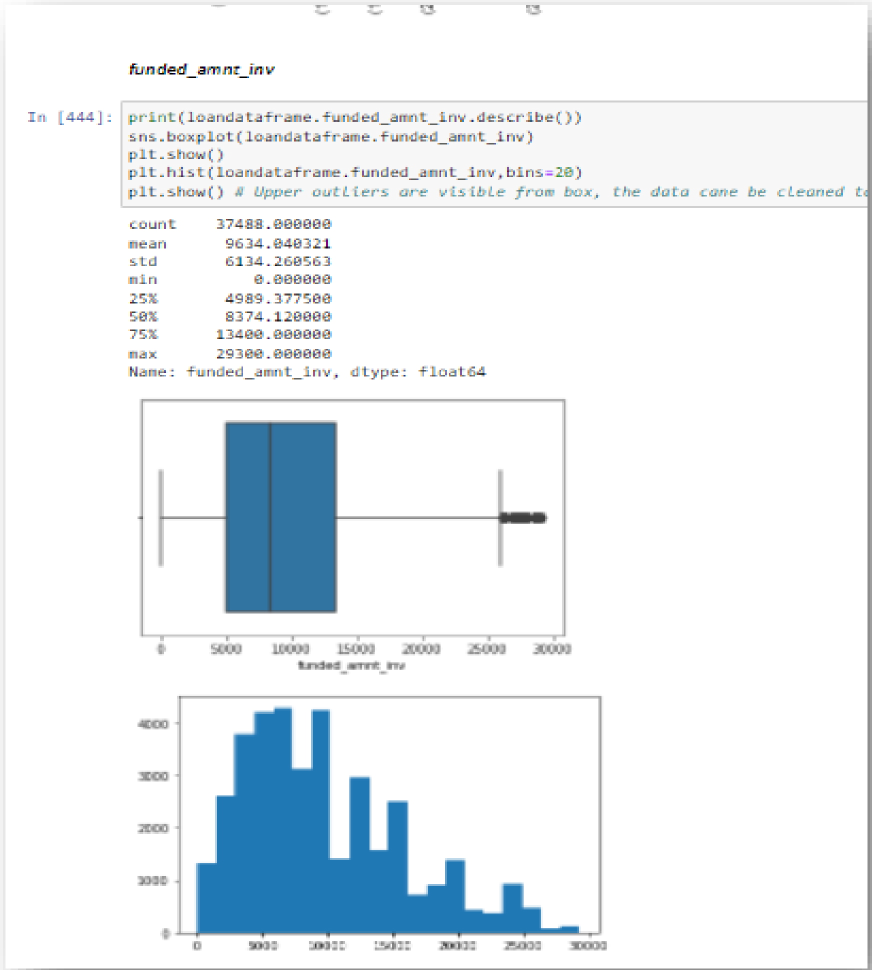
Maximum defaulters are those who has taken loan between 5,000 to 10,000.
Second Set of Defaulters are in the range of 1000 to 5,000.





Data Analysis (Univariate Use Case 2 – funded_amt_inv)

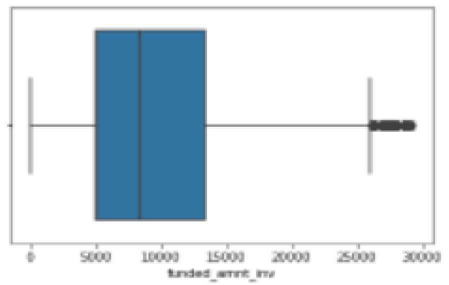
- Based on the boxplot, it is evident that there are outliers which needs to be fixed before doing an inference around it. IQR method is used to remove the outlier
- Data is binned to understand the amount distribution across different binned groups.
- Data is also categorized based on charged off status to understand which funded group having more defaulters

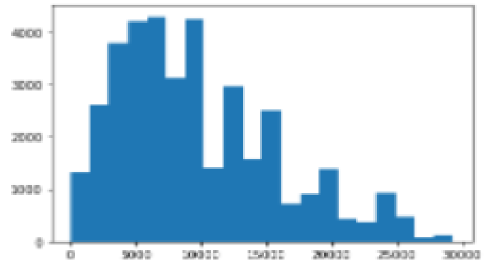


funded_amnt_inv

```
In [444]: print(loandataframe.funded_amnt_inv.describe())
          sns.boxplot(loandataframe.funded_amnt_inv)
          plt.show()
          plt.hist(loandataframe.funded_amnt_inv,bins=20)
          plt.show() # Upper outliers are visible from box, the data cane be cleaned t
```

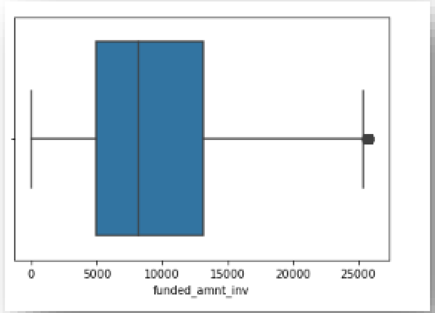
```
count    37488.000000
mean      9634.040321
std       6134.260563
min          0.000000
25%       4989.377500
50%       8374.120000
75%      13400.000000
max      29300.000000
Name: funded_amnt_inv, dtype: float64
```

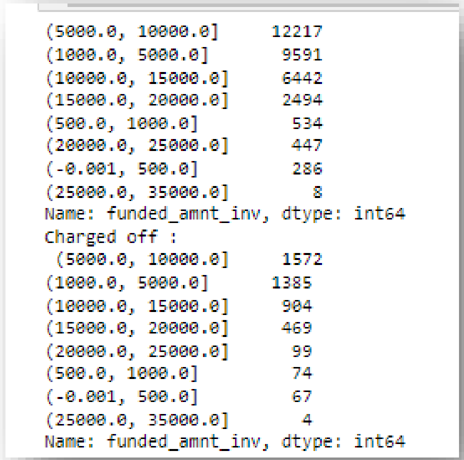




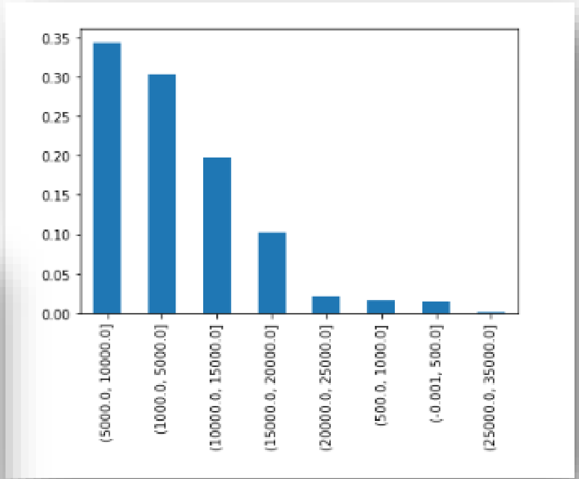
Analysis Explanation:

Attribute funded_amt_inv: This attribute explains about the total amount committed by investor at that point
Minimum funded amount approved: 0
Maximum funded amount approved: 29300
When categorized data into 8 such bins, shows that maximum loan that has been requested is between the range of 5000-10000.
Maximum defaulters are those who has taken loan between 5,000 to 10,000.
Second Set of Defaulters are in the range of 1,000 to 5,000.



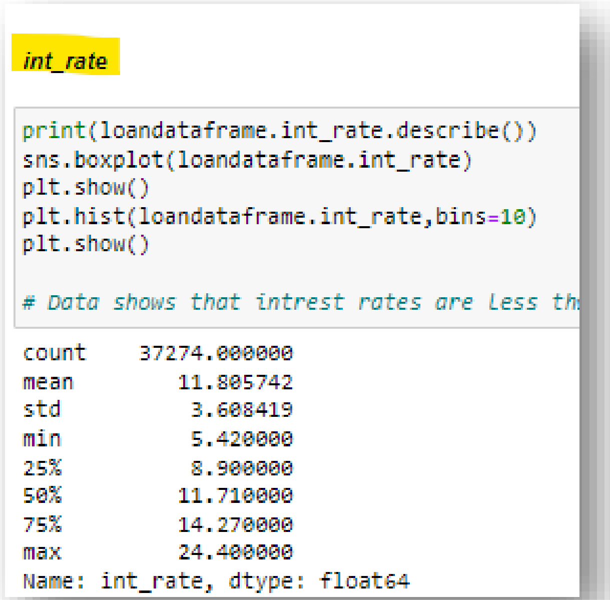


```
(5000.0, 10000.0]     12217
(1000.0, 5000.0]       9591
(10000.0, 15000.0]     6442
(15000.0, 20000.0]     2494
(500.0, 1000.0]         534
(20000.0, 25000.0]      447
(-0.001, 500.0]         286
(25000.0, 35000.0]        8
Name: funded_amnt_inv, dtype: int64
Charged off :
 (5000.0, 10000.0]     1572
(1000.0, 5000.0]      1385
(10000.0, 15000.0]     904
(15000.0, 20000.0]     469
(20000.0, 25000.0]      99
(500.0, 1000.0]         74
(-0.001, 500.0]         67
(25000.0, 35000.0]       4
Name: funded_amnt_inv, dtype: int64
```



Data Analysis (Univariate Use Case 3 – int_rate)

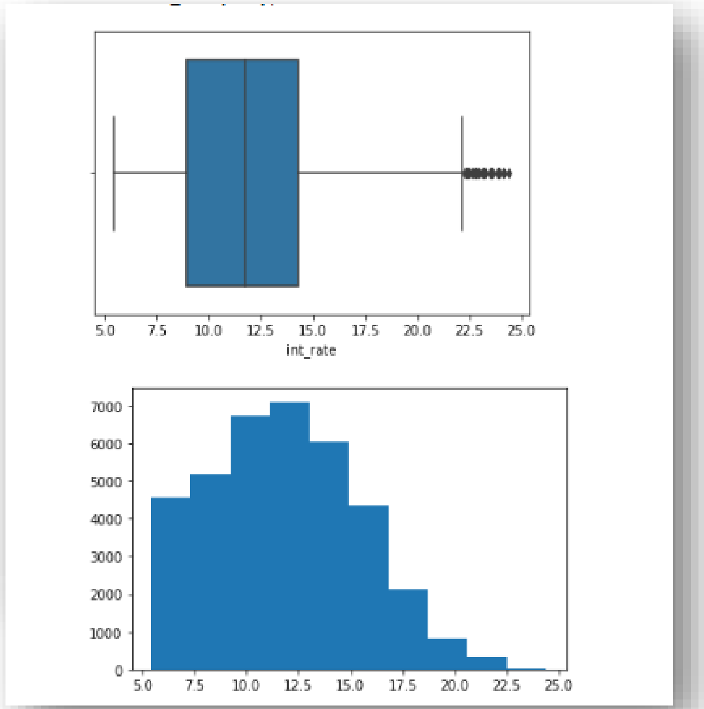
- Mean and Median of data is almost same, which suggested that the data is well distributed with very less standard deviation.
- When the data is binned in 9 different binned group namely bins=[0,2,4,8,10,12,14,20,24]), maximum number of applicant applied for loan with interest rate in 4-8% followed 10-20%. Similarly most of the defaulters are for loans having interest rate 14-20%



int_rate

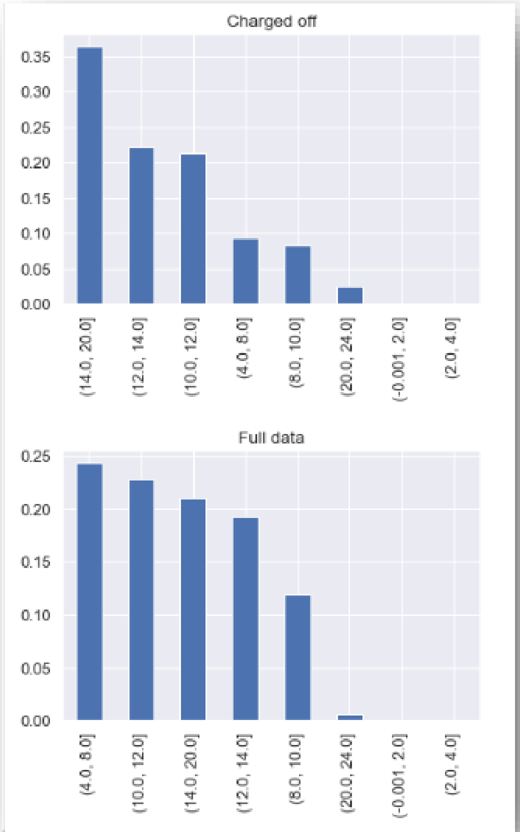
```
print(loandataframe.int_rate.describe())
sns.boxplot(loandataframe.int_rate)
plt.show()
plt.hist(loandataframe.int_rate,bins=10)
plt.show()

# Data shows that intrest rates are less th
```

```
count    37274.000000
mean        11.805742
std          3.608419
min          5.420000
25%          8.900000
50%         11.710000
75%         14.270000
max         24.400000
Name: int_rate, dtype: float64
```



Analysis Explanation:

Attribute Loan Data: This attribute explains about the interest rate on the loan.
Minimum interest rate on loan: 5.42%
Maximum interest rate on loan :24.40%
When categorized Loan Data into 9 such bins, shows that maximum interest rate that has been requested is between the range of 4-8%.
Maximum defaulters are those who has taken loan between 14-20%
Second Set of Defaulters are in the range of 12-14%.

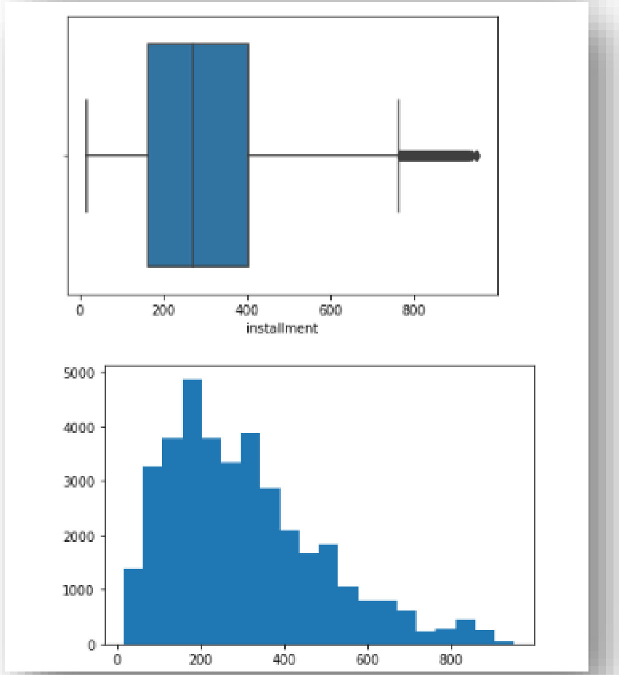


Data Analysis (Univariate Use Case 4 – installment)

- To analyse this boxplot, binning is being used.
- There is difference in mean and median; with higher value of std.devation, which points to large number of outliers present in data
- Next step is to remove the outlier from data for further analysis
- Binning the data for full data and for charge off data was done to identify the maximum number of defaulter in any range.

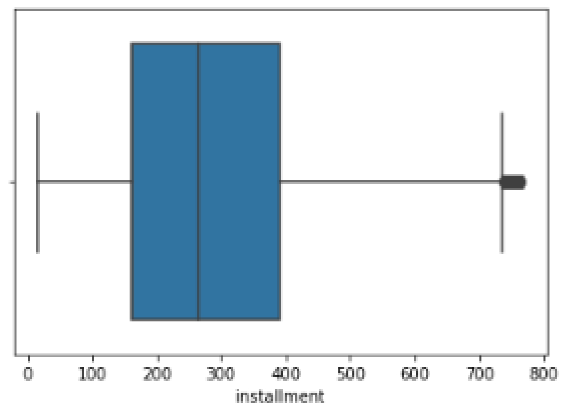
installment

```
sns.boxplot(loandataframe.installment)
plt.show()
plt.hist(loandataframe.installment,bins=20)
plt.show()
loandataframe.installment.describe()
```



Data with outlier

```
sns.boxplot(loandataframe.installment)
plt.show()
print(loandataframe.installment.describe())
```



```
count    37258.000000
mean       290.795545
std        163.596423
min         15.690000
25%        162.190000
50%        264.660000
75%        391.547500
max        765.990000
Name: installment, dtype: float64
```

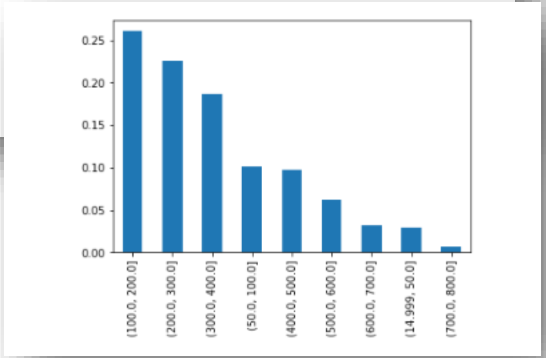
Data without outlier

Analysis Explanation:

Attribute Loan Data: This attribute explains about the monthly payment owned by applicant once the loan approved.
Minimum installment for a loan: 15
Maximum installment for a loan: 765
When categorized Loan Data into 8 such bins, shows that maximum installment for a particular loan is in between the range of 100-300.
Maximum defaulters are those who has installment in between 100 to 200.
Second Set of Defaulters are in the range of 200 to 300.

```
print(loandataframe.installment.value_counts(bins=[15,50,100,200,300,400,500,600,700,800],normalize=True))
print("Charged off :\n",loandataframe[loandataframe.loan_status=='Charged Off'].installment.value_counts(
loandataframe[loandataframe.loan_status=='Charged Off'].installment.value_counts(bins=[15,50,100,200,300,
```

```
(100.0, 200.0]    0.277367
(200.0, 300.0]    0.224211
(300.0, 400.0]    0.187701
(50.0, 100.0]     0.096661
(400.0, 500.0]    0.094506
(500.0, 600.0]    0.054999
(600.0, 700.0]    0.032387
(14.999, 50.0]    0.025766
(700.0, 800.0]    0.006402
Name: installment, dtype: float64
Charged off :
 (100.0, 200.0]    0.260603
(200.0, 300.0]    0.225623
(300.0, 400.0]    0.186489
(50.0, 100.0]     0.100568
(400.0, 500.0]    0.097289
(500.0, 600.0]    0.061653
(600.0, 700.0]    0.031482
(14.999, 50.0]    0.029515
(700.0, 800.0]    0.006777
Name: installment, dtype: float64
```



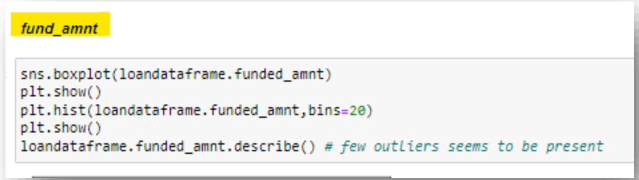
Charged off data

Data Analysis (Univariate Use Case 5 – fund_amnt)

- To analyse this boxplot, binning is being used.
- The data seems to be well distributed with minimum number of upper outlier.
- Binning the data in bins of [0,500,1000,5000,10000,15000,20000,25000,35000] for full data and as well for charge-off data was done to find out any relevant information from the data.

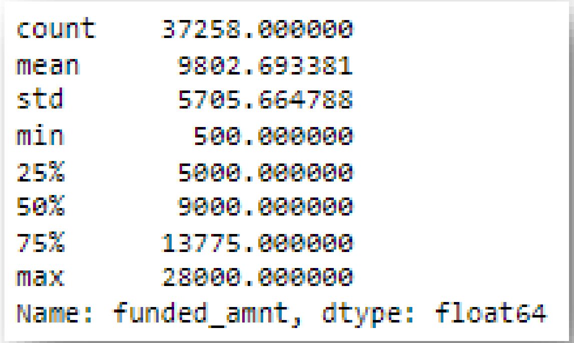
Analysis Explanation:

Attribute Loan Data: This attribute explains about the total amount committed to the loan.
Minimum committed amount : 500
Maximum committed amount: 28000
When categorized committed amount data into 9 such bins, shows that maximum committed amount for a particular loan is in between the range of 5k to 10k.
Maximum defaulters for committed amount for a loan 5k to 10k
Second Set of Defaulters are in the range of 1k to 5k.

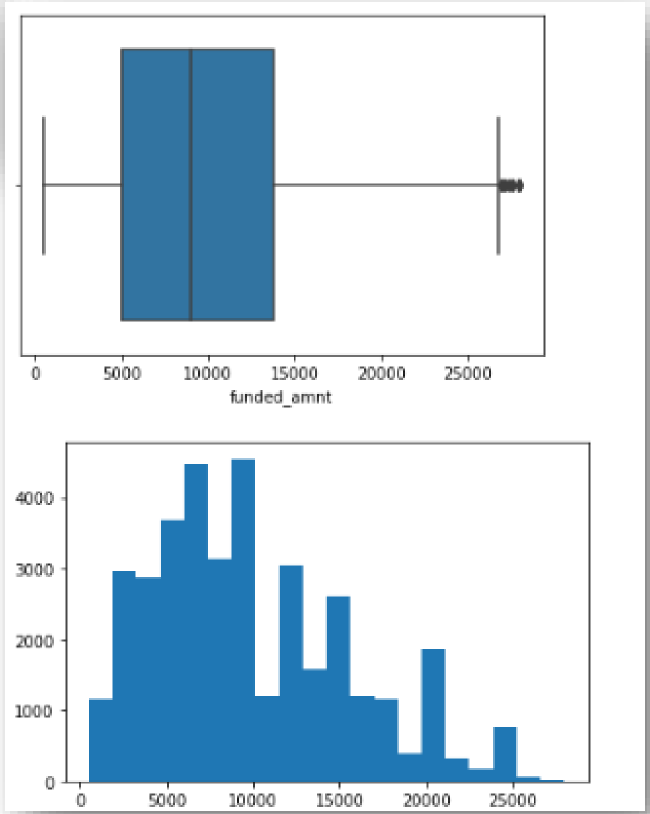


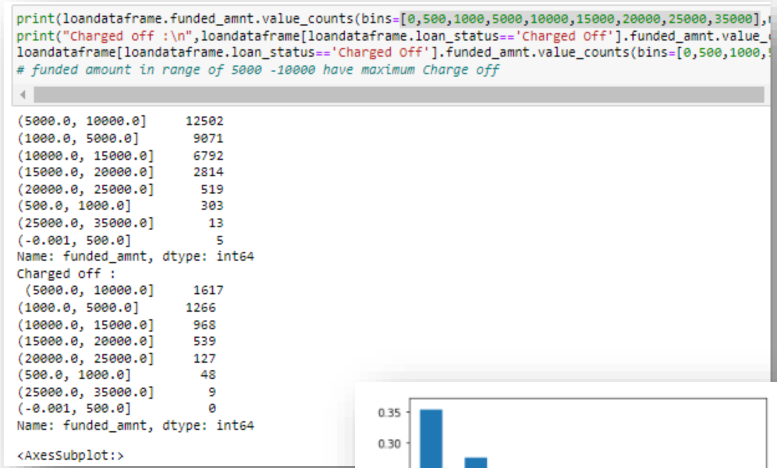
fund_amnt

```
sns.boxplot(loandataframe.funded_amnt)
plt.show()
plt.hist(loandataframe.funded_amnt,bins=20)
plt.show()
loandataframe.funded_amnt.describe() # few outliers seems to be present
```

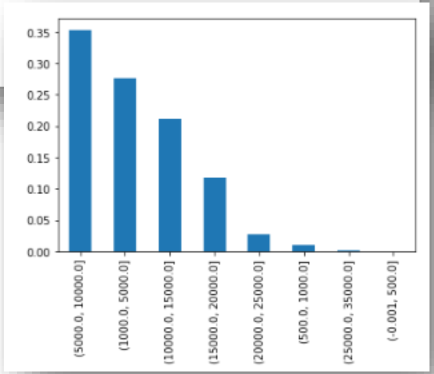


```
count    37258.000000
mean      9802.693381
std       5705.664788
min        500.000000
25%       5000.000000
50%       9000.000000
75%      13775.000000
max      28000.000000
Name: funded_amnt, dtype: float64
```





```
print(loandataframe.funded_amnt.value_counts(bins=[0,500,1000,5000,10000,15000,20000,25000,35000],
print("Charged off :\n",loandataframe[loandataframe.loan_status=='Charged Off'].funded_amnt.value_
loandataframe[loandataframe.loan_status=='Charged Off'].funded_amnt.value_counts(bins=[0,500,1000,
# funded amount in range of 5000 -10000 have maximum Charge off

(5000.0, 10000.0]     12502
(1000.0, 5000.0]       9071
(10000.0, 15000.0]     6792
(15000.0, 20000.0]     2814
(20000.0, 25000.0]      519
(500.0, 1000.0]         303
(25000.0, 35000.0]       13
(-0.001, 500.0]           5
Name: funded_amnt, dtype: int64
Charged off :
 (5000.0, 10000.0]     1617
(1000.0, 5000.0]      1266
(10000.0, 15000.0]     968
(15000.0, 20000.0]     539
(20000.0, 25000.0]     127
(500.0, 1000.0]         48
(25000.0, 35000.0]       9
(-0.001, 500.0]          0
Name: funded_amnt, dtype: int64

<AxesSubplot:>
```



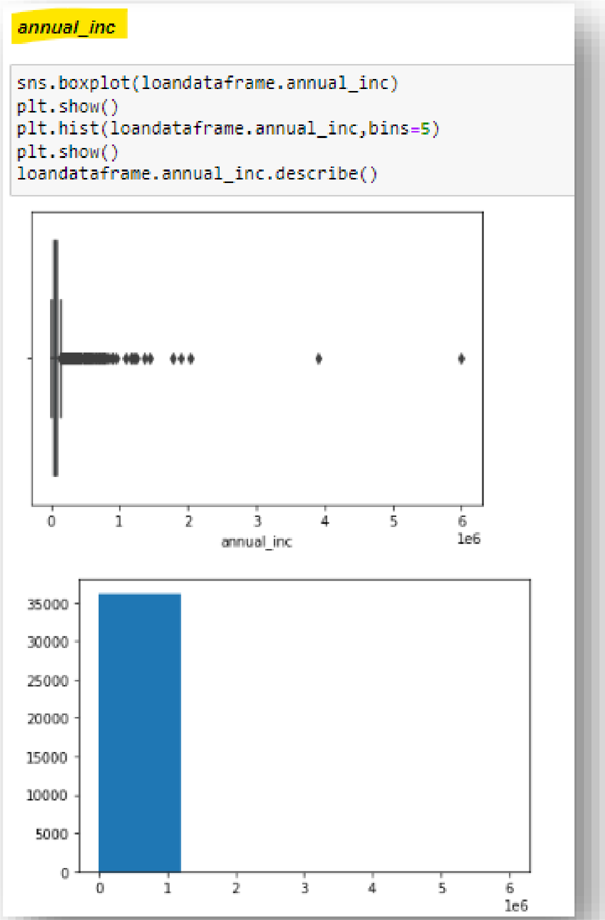
Charged off data

Data Analysis (Univariate Use Case 6 – annual_inc)

- To analyse this boxplot and binning is being used
- The data seems to fragmented having upper outlier
- Binning the data again in bins of [4000, 10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000, 120000, 140000] for full data and as well for charge-off data and find out any relevant information from the data

Analysis Explanation:

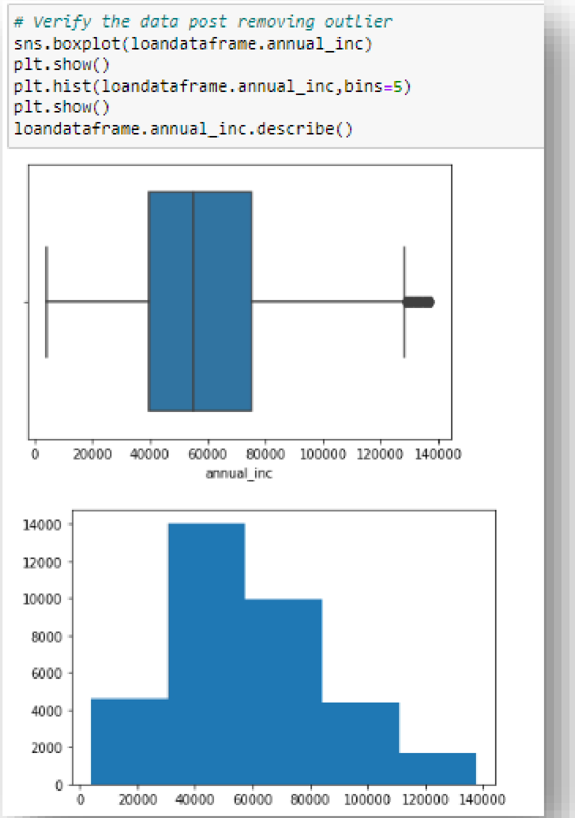
Attribute Data: This attribute explains about the annual income of applicant.
Minimum Annual income: 500
Maximum Annual income: 28000
When categorized Annual income data into 13 such bins, shows that maximum annual income of the applicant is in between the range of 40k to 50k.
Maximum defaulters are done by applicant whose annual income 30k to 40k
Second Set of Defaulters are in the range of 1k to 5k.



annual_inc

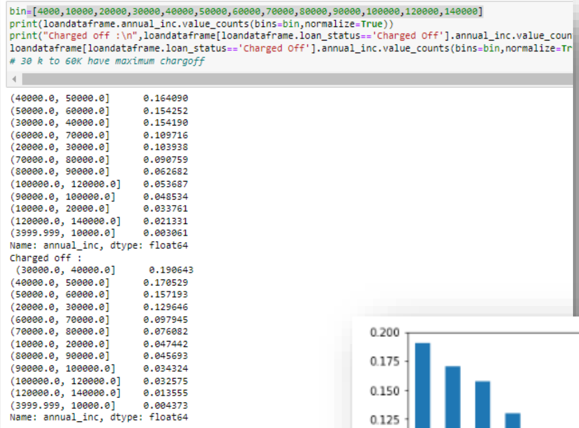
```
sns.boxplot(loandataframe.annual_inc)
plt.show()
plt.hist(loandataframe.annual_inc,bins=5)
plt.show()
loandataframe.annual_inc.describe()
```

Data with outlier



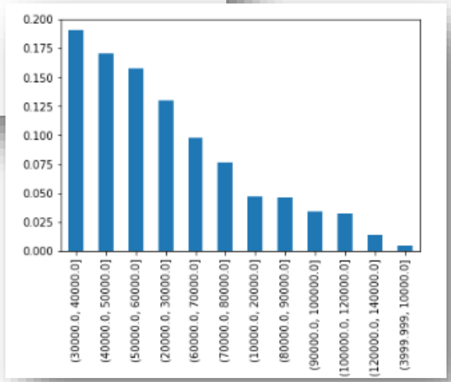
```
# Verify the data post removing outlier
sns.boxplot(loandataframe.annual_inc)
plt.show()
plt.hist(loandataframe.annual_inc,bins=5)
plt.show()
loandataframe.annual_inc.describe()
```

Data without outlier



```
bin=[4000,10000,20000,30000,40000,50000,60000,70000,80000,90000,100000,120000,140000]
print(loandataframe.annual_inc.value_counts(bins=bin,normalize=True))
print("Charged off :\n",loandataframe[loandataframe.loan_status=='Charged Off'].annual_inc.value_coun
loandataframe[loandataframe.loan_status=='Charged Off'].annual_inc.value_counts(bins=bin,normalize=Tr
# 30 k to 60K have maximum chargoff
```

```
(40000.0, 50000.0]      0.164090
(50000.0, 60000.0]      0.154252
(30000.0, 40000.0]      0.154190
(60000.0, 70000.0]      0.109716
(20000.0, 30000.0]      0.103938
(70000.0, 80000.0]      0.090759
(80000.0, 90000.0]      0.062682
(100000.0, 120000.0]    0.053687
(90000.0, 100000.0]     0.048534
(10000.0, 20000.0]      0.033761
(120000.0, 140000.0]    0.021331
(3999.999, 10000.0]     0.003061
Name: annual_inc, dtype: float64
Charged off :
 (30000.0, 40000.0]      0.190643
(40000.0, 50000.0]      0.170529
(50000.0, 60000.0]      0.157193
(20000.0, 30000.0]      0.129646
(60000.0, 70000.0]      0.097945
(70000.0, 80000.0]      0.076082
(10000.0, 20000.0]      0.047442
(80000.0, 90000.0]      0.045693
(90000.0, 100000.0]     0.034324
(100000.0, 120000.0]    0.032575
(120000.0, 140000.0]    0.013555
(3999.999, 10000.0]     0.004373
Name: annual_inc, dtype: float64
```

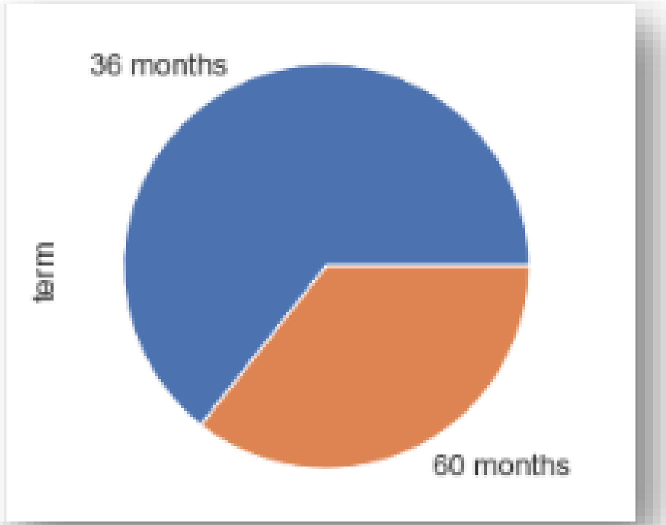


Charged off data

Data Analysis (Univariate few additional uses cases)

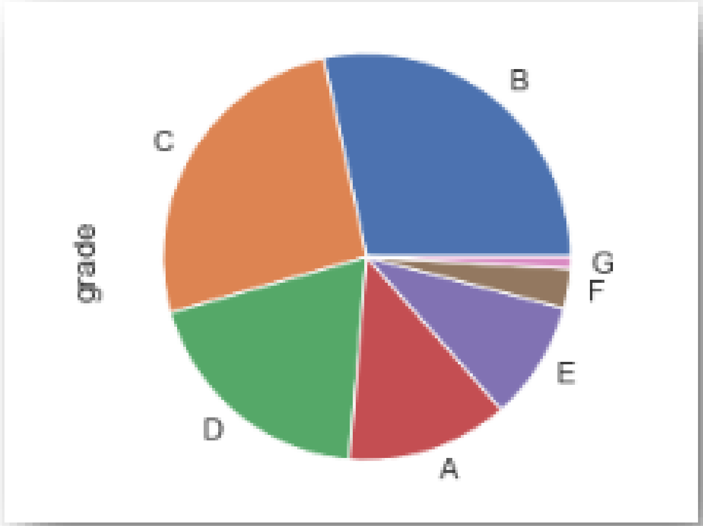
TERM

Attribute Term: Duration for which loan has been taken.
Maximum defaulters are those who has taken loan for 36 months.



GRADE

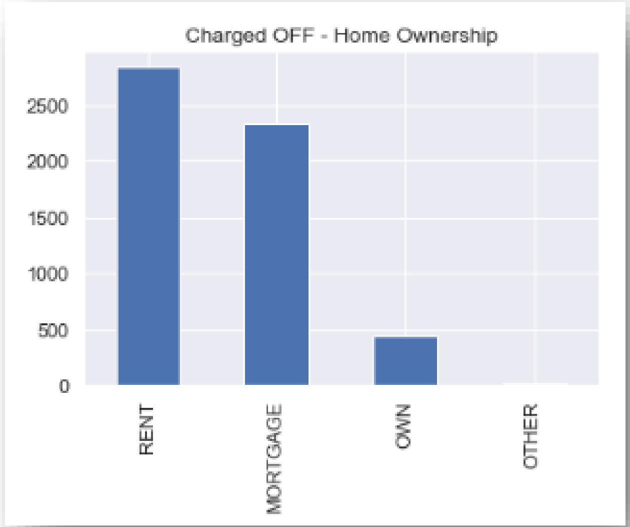
Attribute Grade: LC assigned loan grade.
Maximum defaulters are in the category with Grade B and C.



HOME_OWNERSHIP

Attribute home_ownership: The home ownership status provided by the borrower during registration. Our values are: RENT, OWN, MORTGAGE, OTHER.

Maximum defaulters are those who lives in Rented house.



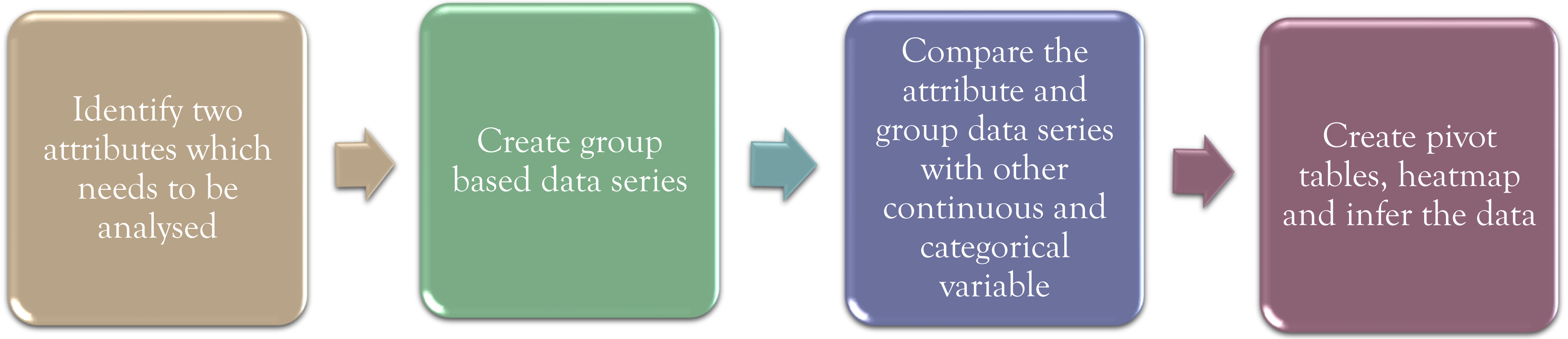
Data Analysis (**Univariate** Summary)

Observation based on univariate analysis for each associated variable.

Probability of having defaulters are more if they belongs to below mentioned points

- "Loan_amount" or "funded_amount_inv" in 5000 to 10000 range.
- Interest rates are heigher than 14% mainly 14-20%
- Installment range in between 100 -200
- Anuual income in range between 30k to 60K
- DTI value in range of 12-16
- Applicant who have 5-10 Accounts.
- Applicant who select 36 months of loan term, however the number are high for Fully paid as well.
- Grade is B & C
- Applicant with employee length 10 Or 1
- Applicant with home ownership as Rent
- Loan issued at year end mostly in the month of DEC
- Purpose is to debt_consolidation & Credit card payment
- Loan issued to people from CA state.
- Verification status is not verified

Data Analysis (Bivariate)


Identify two attributes which needs to be analysed
Create group based data series
Compare the attribute and group data series with other continuous and categorical variable
Create pivot tables, heatmap and infer the data

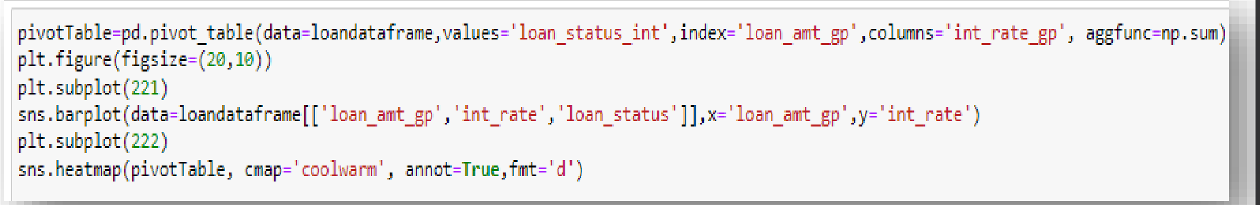
Bivariate analysis done on below mentioned attributes (Green tick means comparison done between these attributes)

	loan_amnt	Int_rate	instalment	grade	Emp_length	Home_ownership	Issue_m	Issue_y	verification	purpose	Addr_state
Loan_amt		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Int_rate	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
amount_inc	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

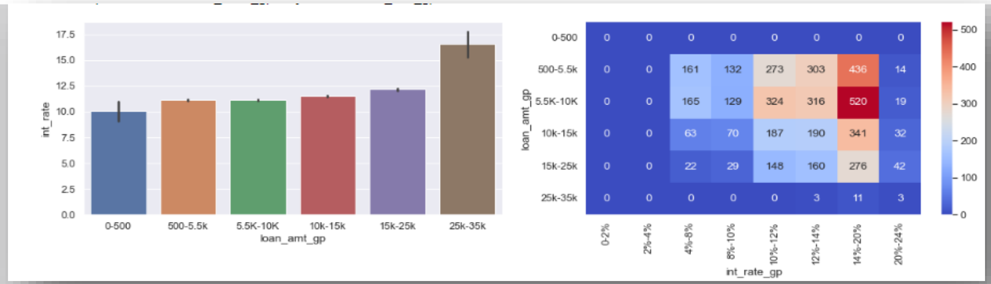
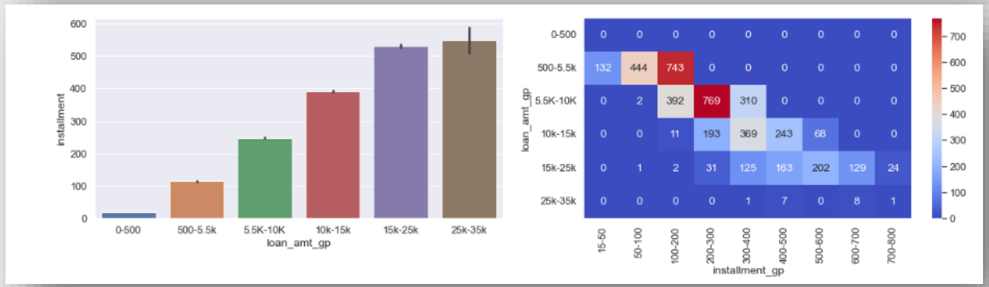
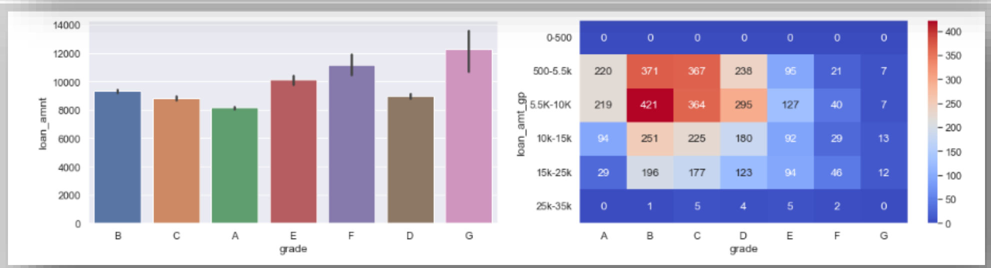
Data Analysis (Bivariate Analysis - Loan)

Bivariate analysis done on loan_amt

- The heat map analysis is done by creating a pivot table with index as loan_amt_gp and columns as respective other attribute, with value on loan_status_int.
- Bar char is created across loan_amnt_gp vs [different attribute]

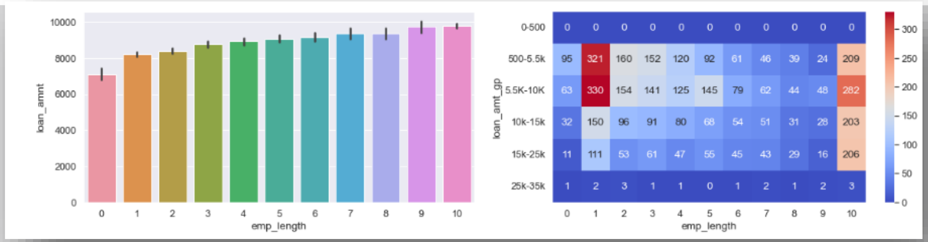
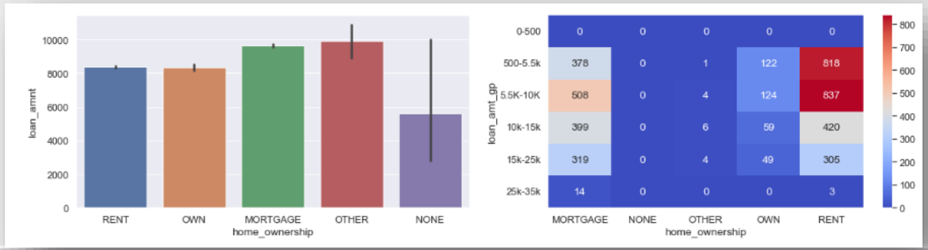
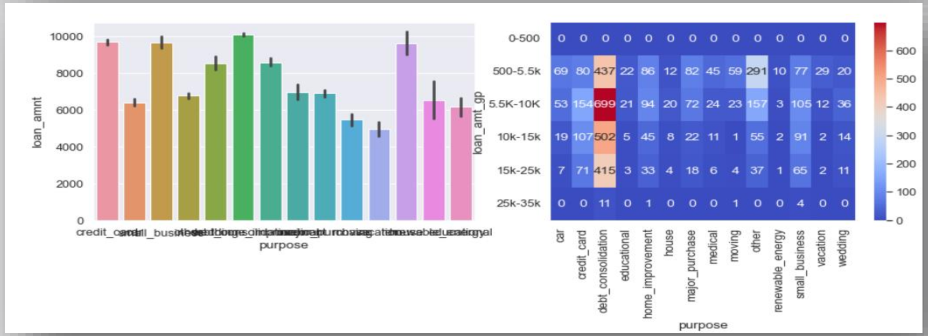


```
pivotTable=pd.pivot_table(data=loandataframe,values='loan_status_int',index='loan_amt_gp',columns='int_rate_gp', aggfunc=np.sum)
plt.figure(figsize=(20,10))
plt.subplot(221)
sns.barplot(data=loandataframe[['loan_amt_gp','int_rate','loan_status']],x='loan_amt_gp',y='int_rate')
plt.subplot(222)
sns.heatmap(pivotTable, cmap='coolwarm', annot=True,fmt='d')
```

Attribute	Graph	Inference
Int_rate		1. With increase in loan amount interest rate is also increasing 2. Maximum number of Charge –off happens when annual income is 5.5K -10K range having intreste rate 14%-20%
instalment		1. With increase in loan amount, number of instalment is also increases. 2. Maximum numbers of defaults happen when loan amount is in 500 -10k range and instalment is in 100-300.
grade		1. G grade have maximum number of loan amount approved, followed by F. 2. Most defaults happen when loan amount is 5.5k -10k range and belongs to B and C grade

Data Analysis (Bivariate – Loan cont..)

Similar kind of analysis is done for issue_y, verification and address

Attribute	Graph	Inference
Emp_length		1. Loan_amt also increases with increase in employment length. 10 years have maximum number of loan amount. 2. Most defaulters are there for loan amount 5.5 k to 10k for employee length is 1 year and 10 years.
Home_ownership		1. Loan amount is more for people who have Mortgage or others. 2. Most of the defaulters- are there with ownership marked as Rent and Loan amount is 5.5k -10k
purpose		1. Maximum Loan_amnt is taken for debt_consolidation and charged off by small business.. 2. Defaults are also mainly happen for debt_consolidation when loan amount is in between 5.5k-10k

Overall analysis –

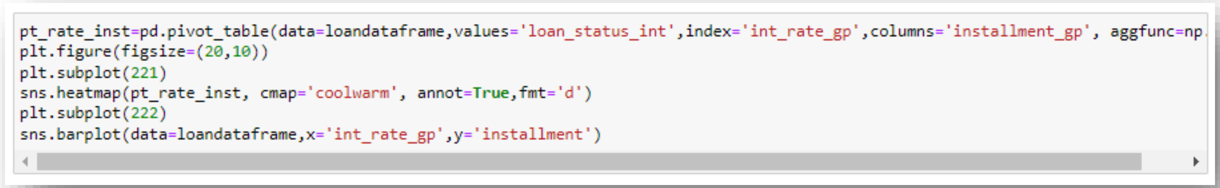
Loan Amount Maximum number of "charged off" when

1. Loan amount 5.5k and interest rate is in between 14-12%
2. Loan amount 5.5k and grade is B
3. Loan amount 5.5k-10K and installments are between 100-300
4. Loan amount 5.5k-10K and emp-length 1 or 10 Years
5. loan amount 5.5k-10K and homeownership as Rent
6. loan amount 5.5k-10K and purpose is debt_consolidation
7. loan amount 5.5k-10K and state CA
8. Maximum loan amount charged off by small business
9. Maximum loan amount charged off by state NE

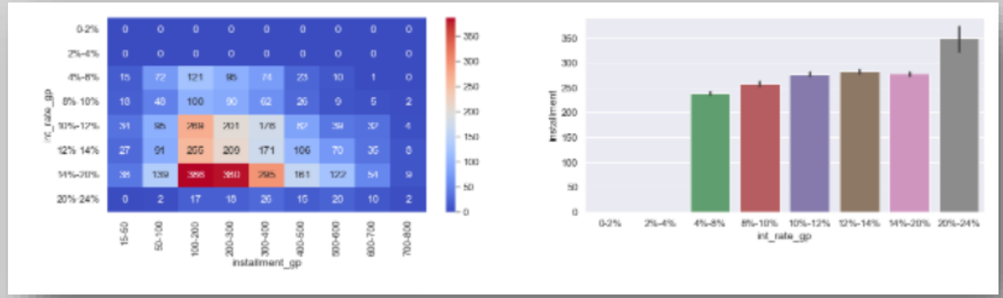
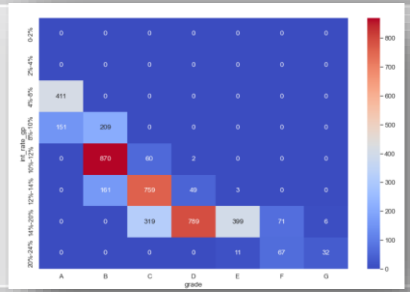
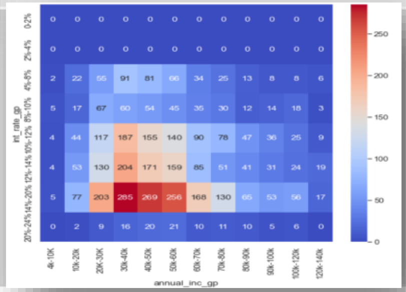
Data Analysis (Bivariate –int_rate)

Bivariate analysis done on int_rate

- The heat map analysis is done by creating a pivot table with index as *int_rate_gp* and columns as respective other *attributes*, with value on *loan_status_int*.
- Bar char is created across int_rate_gp vs [[different attribute]]

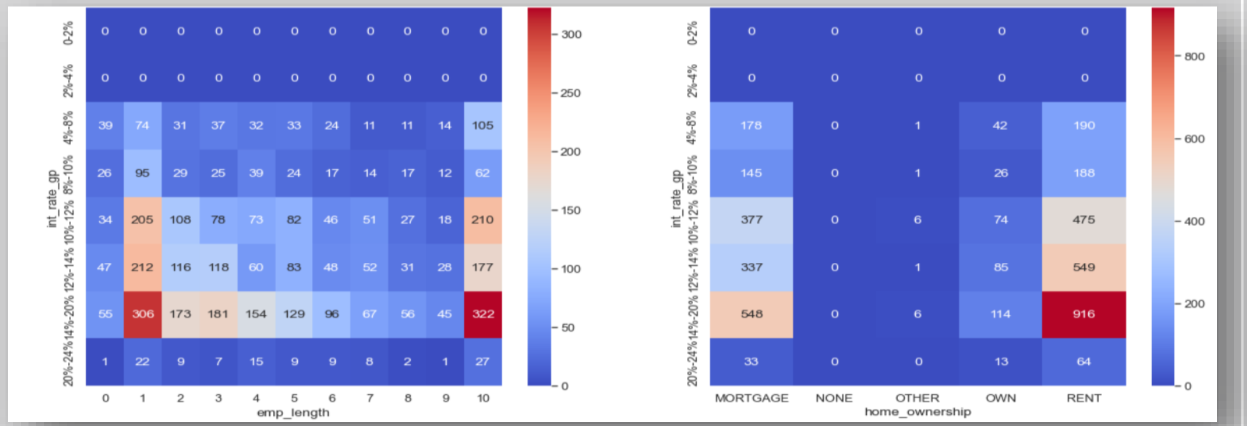


```
pt_rate_inst=pd.pivot_table(data=loandataframe,values='loan_status_int',index='int_rate_gp',columns='installment_gp', aggfunc=np.
plt.figure(figsize=(20,10))
plt.subplot(221)
sns.heatmap(pt_rate_inst, cmap='coolwarm', annot=True,fmt='d')
plt.subplot(222)
sns.barplot(data=loandataframe,x='int_rate_gp',y='installment')
```

Attribute	Graph	Inference
instalment		1. With increase in interest rate , number of instalment is also increases. 2. Maximum numbers of defaults happen when interest rate is in 14-20% range and instalment is in 100-300.
grade		1. Most charge-off happens when interest rate is in between 12-14% and grade in [B,C,D)
annual_inc		1. Most charge-off happens when interest rate is in between 14-20% and annual income in range of 30k-40k

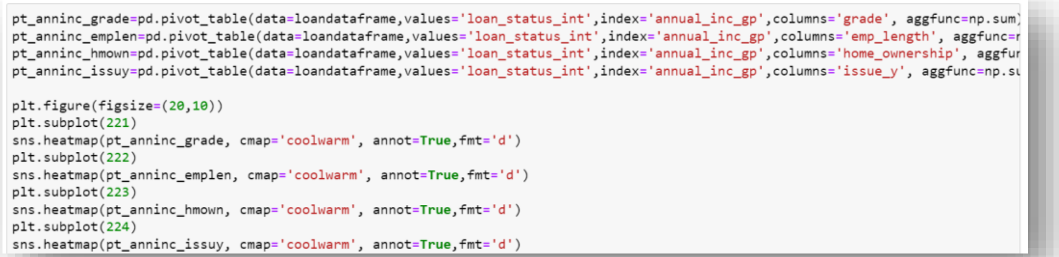
Data Analysis (Bivariate –int_rate.)

Similar kind of analysis is done for , purpose issue_y, verification and address

Attribute	Graph	Inference
Emp_length & Home_Ownership		1. Emp_length: Most of the charged-off happen when employment length is either 1 or 10 years and interest rate is in range 14-20%. 2. Home_ownership : Most of the charged-off happen when home ownerships is Rent and interest rate is in range 14-20%.

Overall analysis –
Loan Amount Maximum number of "charged off" when
1.for 14-20% interest rate, if annualincome is in 30k-40k range.
2.for 12-20% interest rate, if grade is in B,C,D.
3.for 14-20% interest rate, if emp-length is eithier 1,10.
4.for 14-20% interest rate, if home_ownership is at Rent.
5.for 14-20% interest rate, if month is Nov,Dec
6.for 14-20% interest rate, if issue_y is 2011

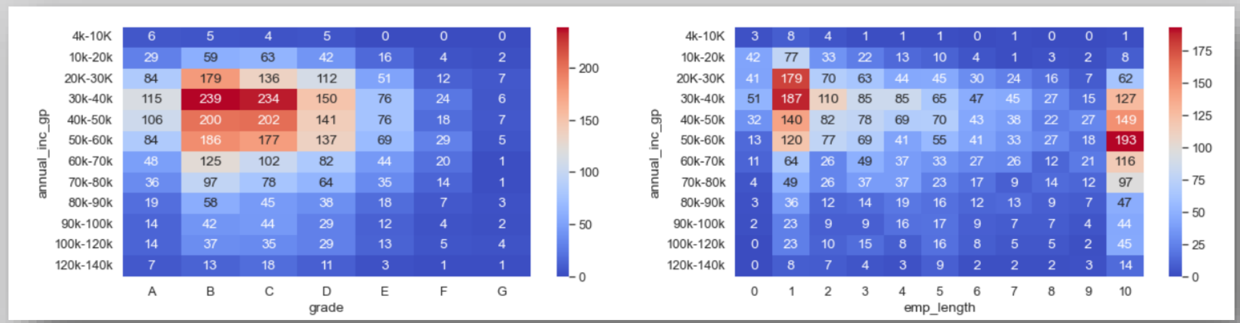
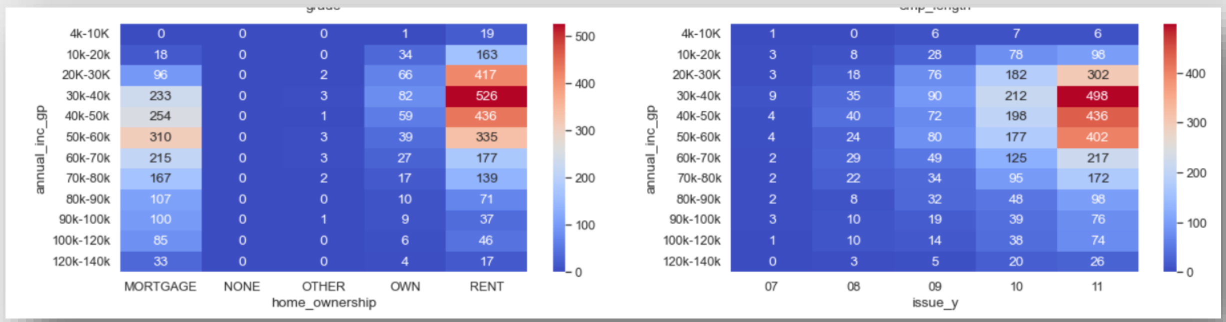
Data Analysis (Bivariate –annual_inc)



```
pt_anninc_grade=pd.pivot_table(data=loandataframe,values='loan_status_int',index='annual_inc_gp',columns='grade', aggfunc=np.sum)
pt_anninc_emplen=pd.pivot_table(data=loandataframe,values='loan_status_int',index='annual_inc_gp',columns='emp_length', aggfunc=r
pt_anninc_hmown=pd.pivot_table(data=loandataframe,values='loan_status_int',index='annual_inc_gp',columns='home_ownership', aggfur
pt_anninc_issuy=pd.pivot_table(data=loandataframe,values='loan_status_int',index='annual_inc_gp',columns='issue_y', aggfunc=np.su

plt.figure(figsize=(20,10))
plt.subplot(221)
sns.heatmap(pt_anninc_grade, cmap='coolwarm', annot=True,fmt='d')
plt.subplot(222)
sns.heatmap(pt_anninc_emplen, cmap='coolwarm', annot=True,fmt='d')
plt.subplot(223)
sns.heatmap(pt_anninc_hmown, cmap='coolwarm', annot=True,fmt='d')
plt.subplot(224)
sns.heatmap(pt_anninc_issuy, cmap='coolwarm', annot=True,fmt='d')
```

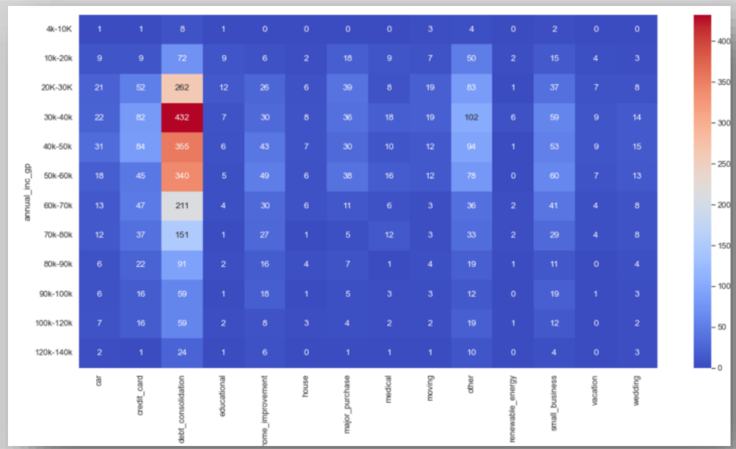
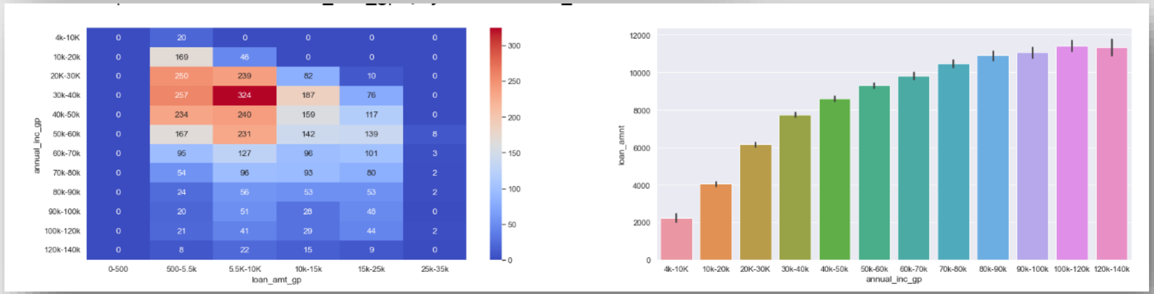
Bivariate analysis done on annual_inc

- The heat map analysis is done by creating a pivot table with index as *annual_inc_gp* and columns as respective other *attributes*, with value on *loan_status_int*.

Attribute	Graph	Inference
Grade & Emp_length		1. Grade : Most charge-off happens when grade is in B & C and annual income is in 30K to 40K range. 2. Emp_length : Most charge-off happens • Income is in 30-40K range and employee length 1 year • Income is in 50-60K range and employee length 10 year
Home_ownership & Issue_y		1. HomeOwnerShip – Most of the charge-off happens when Home-ownership is on Rent and annual income is in 30K-40K range. 2. 2011 have more charge-off compared to any other year. That too in range of 30k-40k annual income

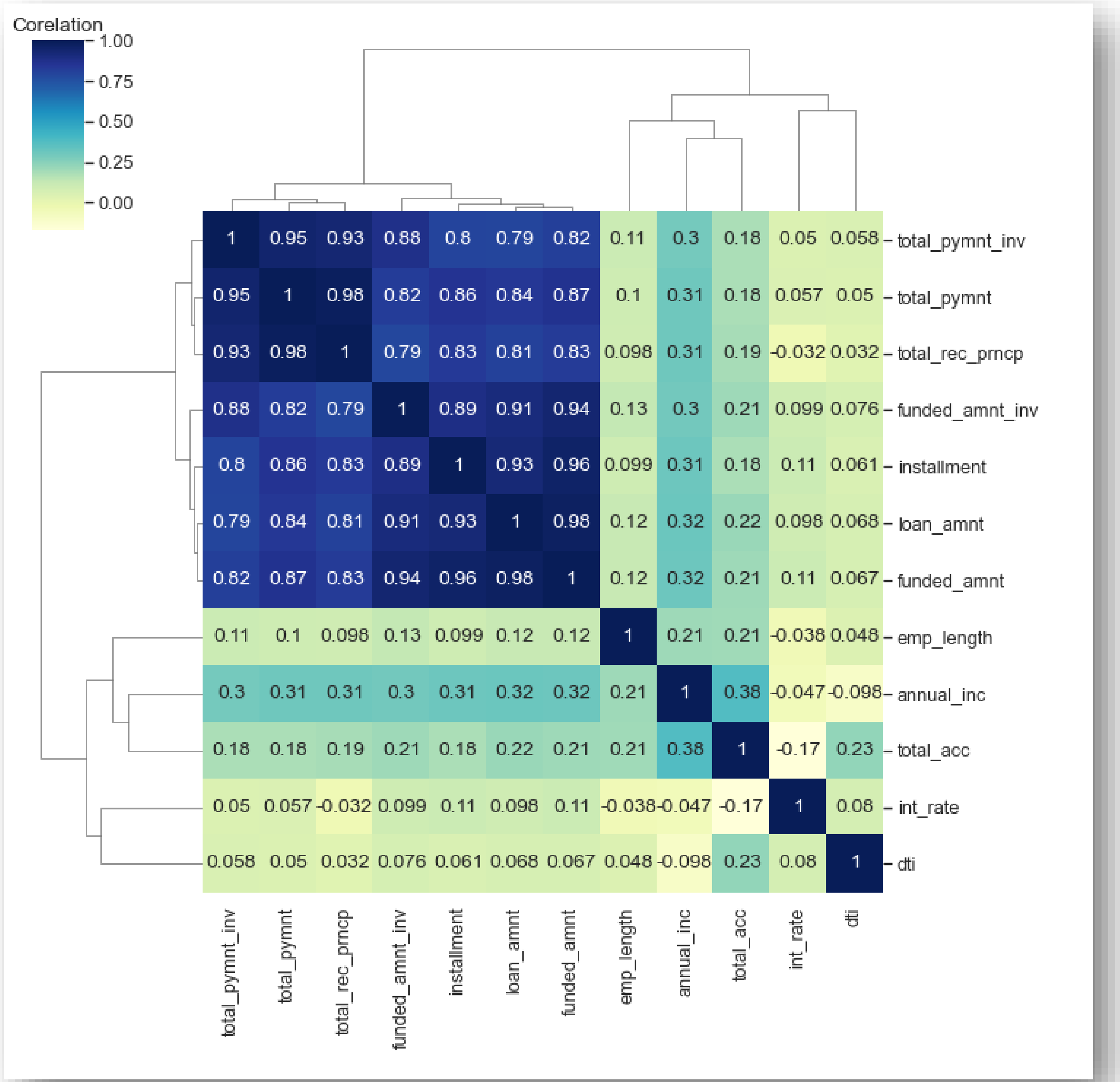
Data Analysis (Bivariate –int_rate..)

Similar kind of analysis is done for verification, address, # of accounts

Attribute	Graph	Inference
purpose		1. Charge-off mainly happens for debt_consolidation where the annual income is in range of 30k-40k
Loan_amount		1. Applicant with higher annual income value applied for higher loan amount. 2. Applicant with annual_income in range of 30-40K have higher rate of charged-off if they applied for loan in amount range of 5.5K to 10K

Overall analysis –
Annual Income
Maxmimum number of charge-off
1. for annual income in 30k-40k and grade is B &C
2. for annual income in30-40K, and emp-length 1 and for 50-60K, if emp-length is 10 years
3. for annual income in30-40K, and house ownership is Rent
4. for annual income in30-40k, and issue year is 2011
5. for annual income in30-40k, and verification status is 'Not-Verified'
6 for annual income in30-40k, and purpose is debt_consolidation
7. for annual income in20k-40K, and installment is in 100-300
8 for annual income in30-40k, and open_acc is 7

Data Analysis (Correlation Matrix)



Overall analysis from the given correlation Matrix

- Loan_amt is highly correlated with installments , funded_amnt, funded_amnt_inv, installments , which means with increase in loan amount other attributes will increase.
- Installments is also highly correlated with loan_amount, funded_amnt and funded_amnt_inv
- Int_rate is negatively correlated to open_acc, which means more open _acc leads to low interest rate.

Data Analysis (Bivariate Summary)

Observation based on Bivariate analysis for each associated variable.

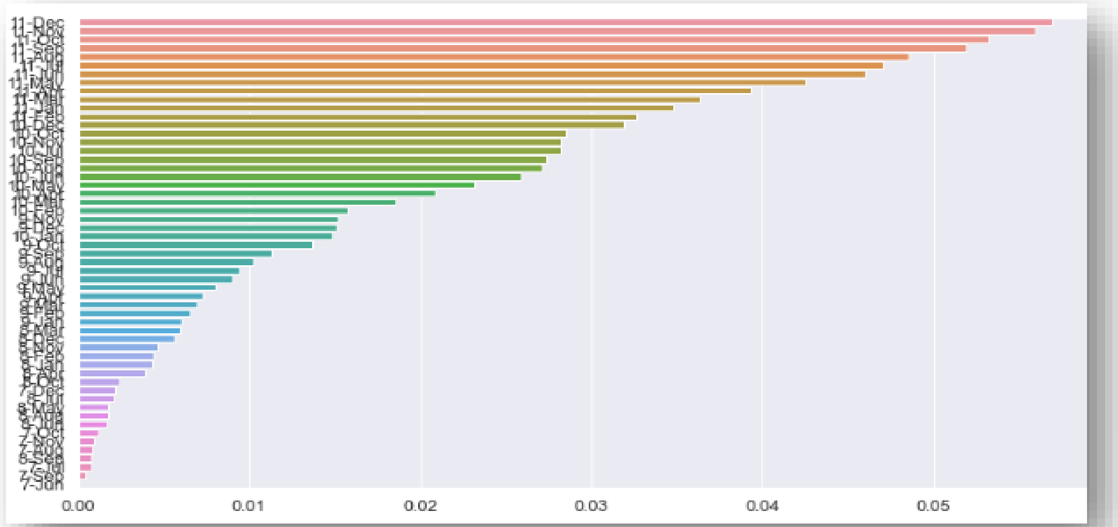
Probability of having Defaulters are more if they belong to below category in combination

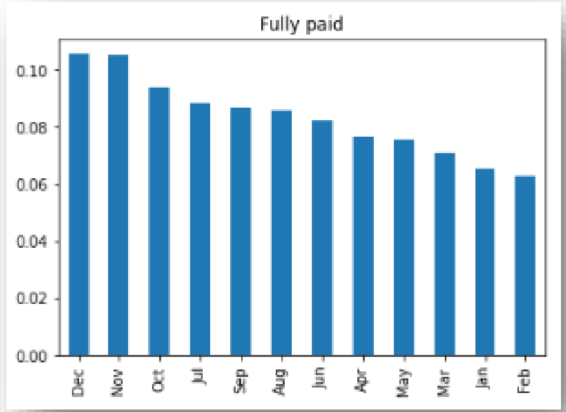
- When Applicant have annual income in range of 30K-40K.
- And applied for Loan amount in between 5.5K -10K
- And even chose higher interest rate of 14-20%,
- Also having an employment history of either 1 or 10 years.
- Also Chose to repay installment in 100-300 installments,
- Also the loan issued in the month of Nov. to Dec.
- As well if applicant is not verified.
- As well If applicant has 7-8 bank accounts
- And loan is taken for the purpose of “debt_considation”

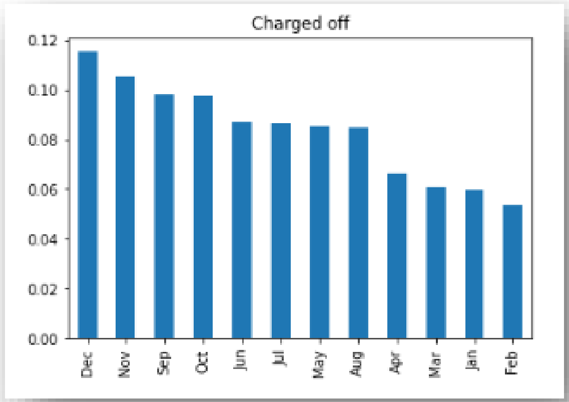
Data Analysis (Derived Analysis – issue_d)

Attribute issue_d: The month which the loan was funded.
Irrespective of year, loan is distributes across each month with more or less same value, However most of the defaults happens for the loan provided for the month of December.

- The data provided was having Month and the year of the loan issue date
- As part of analysis, date was split into Month and Year and new columns/attributes were derived out of it.
- Further all the loans were grouped on the basis of Month.
- Below bar chart shows that maximum defaulters are in the month of Dec.







Data Analysis (Derived Matrix Summary)

Observation based on Derived matrix analysis using loan issue date.

Probability of having Defaulters are more if they belong to below category

- Loan issued in the month of Nov & Dec.
- Probability is high that at the time of year end, applicant spend more money on luxury like vacation or buying gifts over new year or Christmas etc.

Recommendations

Business Stakeholders / Financial Decision makers should do due diligence using below analytics to make sure that their business growth arrow point towards the incremental side and debt / losses should be minimized.

To support in their decision making below are the top 10 recommendations derived by the analytics team. All below scenarios should be evaluated very carefully before lending the loan:

1. If an applicant belongs to Grade B, C or D and loan amount 5.5k to 10k, then chances of full-payment is very low.
2. Loan amount from 5.5K to 10K and interest rate is 14 to 20 %, these stats are strong indicators of defaulters.
3. One of the simplest analysis says that if the loan is taken for the duration of 36 months, overall count of defaulters are more as compared to those applicants taking loan for 60 months.
4. Another very important point that needs to be dig into is Employment durations is either 0 to 1 year or 10 years and above and the loan amount is between 30K to 40K, the loan may go into debt. Account of Business.
5. Any applicant with 7 or more active bank accounts along with annual income ranging 30K to 40K and loan amount between 5.5K to 10K, risk is high.
6. A very important contributor in deciding if the loan should be given or not is, if the application is falls in the category of "Not-Verified".
7. Avoid giving loan to those applicants who has requested for loan amount around 5.5 K to 10K and also requested for EMI length of 100 to 300.
8. Applications should be triple checked (to be careful), if income range is 30K to 40K and Loan amount is 5500 to 10,000 and interest rage is 14 % to 20%, request might be risky.
9. The risk is high if the purpose of the loan is to pay another debt.
10. Applications coming in for loan section, specially in the month of Dec. or November and loan amount between 5.5 K to 10K, this might be an indicator that requester is asking loan for fun/luxury – as Nov. and Dec. are holiday season.

THANK YOU