Lending Club Case Study

Done by:

Haresh Kumar K J

Prashant

Problem Statement:

- Given a dataset of a customer's profile of a consumer finance company's and we have to analyze from the dataset to make a decision for loan approval based on the applicant's profile. In this scenario, there are two possibilities: if the applicant is likely to repay the loan, then not approving the loan will result in loss to the company and if the applicant is not likely to repay the loan, then approving the loan may lead to a financial loss to the company. So, this case study should apply the EDA techniques to identify whether a customer is likely to default or not.

Problem Solution and approach:

The idea here is to follow all the EDA steps to arrive at a better solution.

Sourcing:

- The loan data set is provided along with the case study assignment (loan.csv).

Cleaning:

- The data cleaning is done step by step starting from fixing rows and columns, fixing missing values, standardizing values, fixing invalid values and filtering data.

Problem Solution and approach(cont..)

Univariate analysis:

- Univariate analysis is performed on various data columns.

Bivariate analysis:

- A set of bivariate analysis is performed mainly against ‘loan_status’ vs other columns which contribute to the loan defaulter analysis.

Derived metrics:

- Required derived metrics have been introduced wisely.

Data Cleaning:

- Initial number of rows and columns are: 39717 x 111
- First the null columns were removed and after that the count of columns got reduced to 57: 39717 x 57
- Then certain columns which don't contribute/help for this case study added below were removed: 'pymnt_plan','initial_list_status','collections_12_mths_ex_med','policy_code','acc_now_delinq', 'application_type', 'pub_rec_bankruptcies', 'tax_liens', 'delinq_amnt'
- Now the array matrix got further reduced to 39717 x 48
- There were still some columns which were duplicated and not related to this case study, and they were also removed. So, after this the final array matrix got reduced to 39717 x 21
- After this the null values rows and columns were replaced with mode values after analysis.

Univariate Analysis:

- Now, out of the 21 columns present “loan_status” column has to be analyzed properly as it determines whether a customer has paid/charged off or a loan is still in progress.
- So, from the “loan_status” column “Current” valued rows can be eliminated as it doesn’t contribute to the defaulter analysis.

```
loan_csv = loan_csv[loan_csv["loan_status"] != "Current"]
```

- So, now the “loan_status” column has only ‘Fully Paid’ or ‘Charged Off’ values.

```
loan_csv["loan_status"].unique()

array(['Fully Paid', 'Charged Off'], dtype=object)
```

Univariate Analysis(cont...)

- ”emp_length” column has some null values and we have identified that the max no of values/entries for the row is nearer to the mode value of that column. So, replaced the null values with the mode value.

```
[28]: # So from the above info we can see the data type of "emp_length" and "revol_util" is object
      # It's better to find the mode of emp_length and see whether the other data/values is nearer to mode
      loan_csv.emp_length.mode()

[28]: 0    10+ years
      Name: emp_length, dtype: object

[29]: # Find the frequency of data in emp_length columns
      loan_csv.emp_length.value_counts()

[29]: emp_length
      10+ years    8488
      < 1 year     4508
      2 years      4291
      3 years      4012
      4 years      3342
      5 years      3194
      1 year       3169
      6 years      2168
      7 years      1711
      8 years      1435
      9 years      1226
      Name: count, dtype: int64

[30]: # As from the above data, we clearly see the mode value can be filled for the missing values as it is having far higher frequency than
      # other values
      mode=loan_csv.emp_length.mode()
      loan_csv.emp_length.fillna(mode[0], inplace=True)
      loan_csv.emp_length.isna().sum()

[30]: 0
```

Univariate Analysis(cont...)

- "emp_length" column was still inconsistent with values 10+ years and <1year, so converted "< 1 year" into 0 and "10+ years" as 10.
- Also, removed % from the "int_rate" column.

```
[33]: # Firstly emp_length column is inconsistent with values 10+ years and <1year, so we can convert < 1 year as 0 and 10+ years as 10
loan_csv["emp_length"] = loan_csv["emp_length"].astype("string")
loan_csv.emp_length = pd.to_numeric(loan_csv.emp_length.apply(lambda x: 0 if "<" in x else (x.split('+')[0] if "+" in x else x.split()[
loan_csv.head()
```

[33]:

	loan_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_length	home_ownership	annual_inc	...	issue_d	loan_status	
0	5000	4975.0	36 months	10.65%	162.87	B	B2	10	RENT	24000.0	...	Dec-11	Fully Paid	
1	2500	2500.0	60 months	15.27%	59.83	C	C4	0	RENT	30000.0	...	Dec-11	Charged Off	
2	2400	2400.0	36 months	15.96%	84.33	C	C5	10	RENT	12252.0	...	Dec-11	Fully Paid	sr
3	10000	10000.0	36 months	13.49%	339.31	C	C1	10	RENT	49200.0	...	Dec-11	Fully Paid	
5	5000	5000.0	36 months	7.90%	156.46	A	A4	3	RENT	36000.0	...	Dec-11	Fully Paid	

5 rows × 21 columns

```
[34]: # Also we need to remove % from the int_rate column
loan_csv["int_rate"] = loan_csv["int_rate"].astype("string")
loan_csv.int_rate = pd.to_numeric(loan_csv.int_rate.apply(lambda x: x.split('%')[0]))
loan_csv.head()
```

[34]:

	loan_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_length	home_ownership	annual_inc	...	issue_d	loan_status	
0	5000	4975.0	36 months	10.65	162.87	B	B2	10	RENT	24000.0	...	Dec-11	Fully Paid	
1	2500	2500.0	60 months	15.27	59.83	C	C4	0	RENT	30000.0	...	Dec-11	Charged Off	
2	2400	2400.0	36 months	15.96	84.33	C	C5	10	RENT	12252.0	...	Dec-11	Fully Paid	sr
3	10000	10000.0	36 months	13.49	339.31	C	C1	10	RENT	49200.0	...	Dec-11	Fully Paid	
5	5000	5000.0	36	7.90	156.46	A	A4	3	RENT	36000.0	...	Dec-11	Fully Paid	

Univariate Analysis(cont…)

- Similarly, removed % from ”revol_util” column.
- Also “annual_inc” column had some outlier, so removed them for a better analysis.

Observations from Univariate Analysis:

- "annual_inc" column value is continuous/consistent for the quantile from 0.1 to 0.95 and it's varying heavily after 0.95. So, removed the records for more than 95%.
- Max annual_inc is 6000000.0 and Min annual_inc is 4000.0

```
[37]: # Find the max of 'annual_inc'
      loan_csv['annual_inc'].max()

[37]: 6000000.0

[38]: # Find the min of 'annual_inc'
      loan_csv['annual_inc'].min()

[38]: 4000.0

[39]: # To check from what % data can be removed
      per_range = loan_csv['annual_inc'].quantile([0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.95,0.96,0.97,0.98,0.99])
      per_range

[39]: 0.10     30000.0
      0.20     37200.0
      0.30     44700.0
      0.40     50004.0
      0.50     59000.0
      0.60     65004.0
      0.70     75000.0
      0.80     90000.0
      0.90    115000.0
      0.95    140004.0
      0.96    150000.0
      0.97    165000.0
      0.98    187000.0
      0.99    234000.0
      Name: annual_inc, dtype: float64

[40]: # From the above quantile info it's obvious that values/data after 95% is highly differing from the other data values, so we can
      # remove the records for those individuals whose 'annual_inc' is greater than 95%
      per_95_inc = loan_csv['annual_inc'].quantile(0.95)
      loan_csv = loan_csv[loan_csv['annual_inc'] <= per_95_inc]
      loan_csv
```

Observations from Univariate Analysis (cont..)

- ”loan_amnt” column is continuously spread across all the quantiles, so no need of omitting outlier for this column.

```
[43]: # Eventhough there looks to be some outliers, we cannot confirm as the outliers are continuous and we can confirm this by using %
      per_loan_amnt = loan_csv['loan_amnt'].quantile([0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.95,0.96,0.97,0.98,0.99,1.0])
      per_loan_amnt

[43]: 0.10     3000.0
      0.20     5000.0
      0.30     6000.0
      0.40     7500.0
      0.50     9250.0
      0.60    10550.0
      0.70    13000.0
      0.80    16000.0
      0.90    20000.0
      0.95    25000.0
      0.96    25000.0
      0.97    25475.0
      0.98    30000.0
      0.99    35000.0
      1.00    35000.0
      Name: loan_amnt, dtype: float64
```

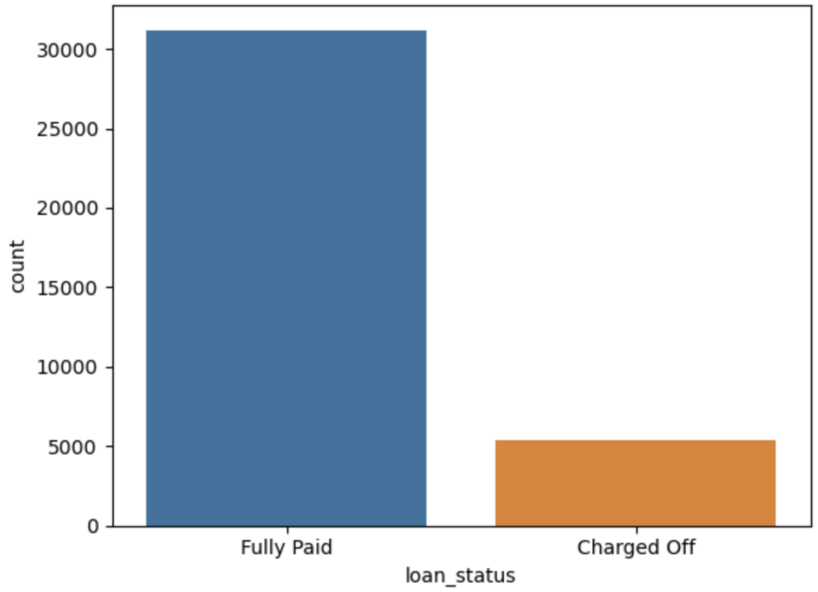
- Similarly other rows like ‘funded_amnt_inv’ , ‘dti’ etc., were checked for any outliers.

Observations from Univariate Analysis (cont..)

- The percentage of “Charged Off” loan_status is fairly less compared to the “Fully Paid” loan_status.

```
[50]: # Univariate Analysis - To check the ratio/count between Fully Paid and Charged Off loan_status
      # As we have to find out/analyze the lending for only defaulter, it's better to consider only the 'Charged Off' loan_status
      sns.countplot(data=loan_csv, x='loan_status')
```

```
[50]: <Axes: xlabel='loan_status', ylabel='count'>
```

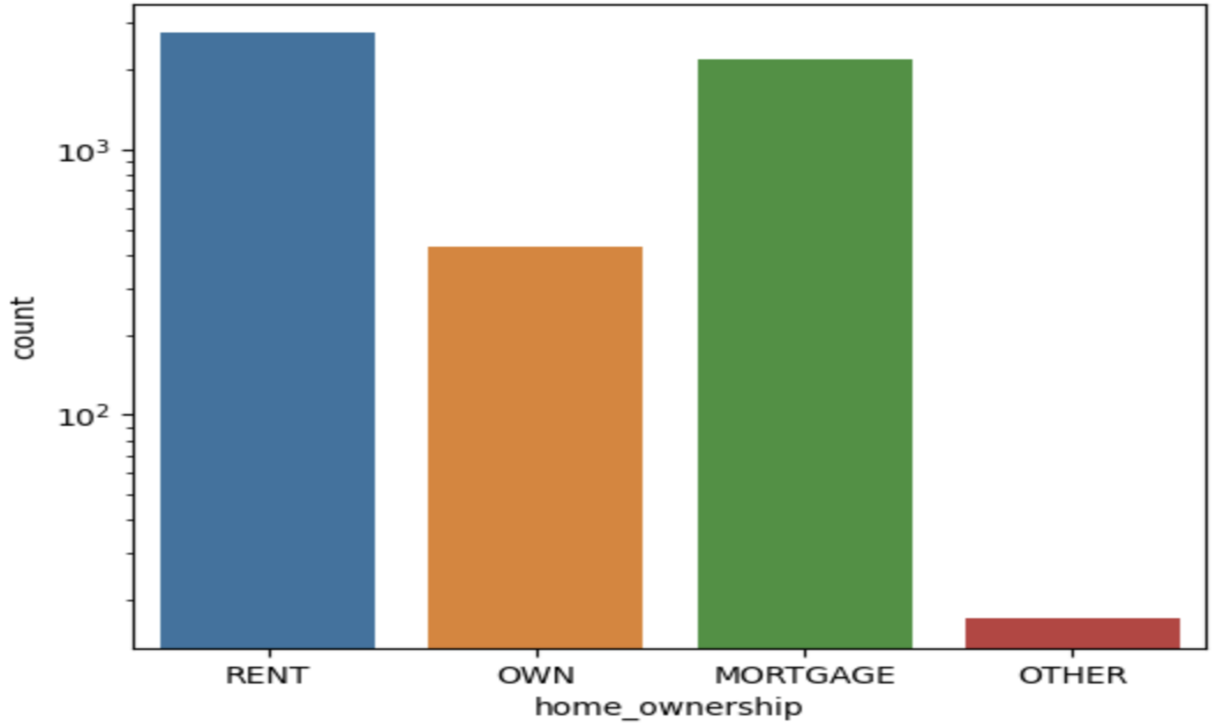


Observations from Univariate Analysis (cont..)

- The no of customer's live in rented and mortgage are more when compared to own house.

```
[160]: # We now need to find the no.of NONE values in the 'home_ownership' column, it should be zero
       fig, ax = plt.subplots(figsize = (6,5))
       ax.set(yscale = 'log')
       sns.countplot(x='home_ownership', data=loan_csv[loan_csv['loan_status']=='Charged Off'])

[160]: <Axes: xlabel='home_ownership', ylabel='count'>
```



Derived Metrics:

- Since the data set is huge to analyse, bins were created for many columns for better grouping.

[162]:
```
# Derived Metrics - Creating new columns with bins which will help for bi-variate analysis
# creating bins for int_rate,open_acc,revol_util,total_acc
loan_csv['int_rate_bin'] = pd.cut(loan_csv['int_rate'], bins=5,precision =0,labels=['5%-10%','10%-15%','15%-19%','19%-21%','21%-24%'])
loan_csv['open_acc_bin'] = pd.cut(loan_csv['open_acc'],bins = 5,precision =0,labels=['2-10','10-20','20-25','25-35','35-44'])
loan_csv['revol_util_bin'] = pd.cut(loan_csv['revol_util'], bins=5,precision =0,labels=['0-20','20-40','40-60','60-80','80-100'])
loan_csv['total_acc_bin'] = pd.cut(loan_csv['total_acc'], bins=5,precision =0,labels=['2-20','20-35','35-55','55-75','75-90'])
loan_csv['annual_inc_bin'] = pd.cut(loan_csv['annual_inc'], bins=5,precision =0,labels =['3k-31k','31k-58k','58k-85k','85k-112k','112k-
```

[163]:
```
loan_csv.head()
```

[163]:

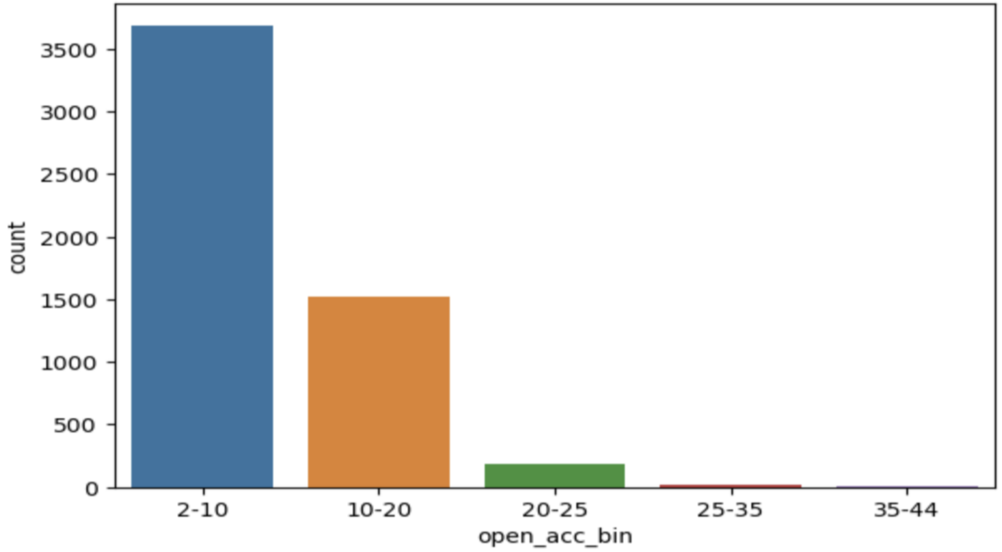
rship	annual_inc	...	inq_last_6mths	open_acc	pub_rec	revol_util	total_acc	int_rate_bin	open_acc_bin	revol_util_bin	total_acc_bin	annual_inc_bin
RENT	24000.0	...	1	3	0	83.7	9	10%-15%	2-10	80-100	2-20	3k-31k
RENT	30000.0	...	5	3	0	9.4	4	15%-19%	2-10	0-20	2-20	3k-31k
RENT	12252.0	...	2	2	0	98.5	10	15%-19%	2-10	80-100	2-20	3k-31k
RENT	49200.0	...	1	10	0	21.0	37	15%-19%	2-10	20-40	20-35	31k-58k
RENT	36000.0	...	3	9	0	28.3	12	5%-10%	2-10	20-40	2-20	31k-58k

Bivariate analysis:

- “open_acc” vs “charged off loan status”
- It is clear that open_acc between 2-10 is more and the probability of getting defaulted is more for that range.

[166]:
```
# We need to analyse the other bin values
# 'open_acc' vs 'open_acc_bin'
fig, ax = plt.subplots(figsize = (15,10))
plt.subplot(221)
sns.countplot(x='open_acc_bin', data=loan_csv[loan_csv.loan_status == 'Charged Off'])
```

[166]: <Axes: xlabel='open_acc_bin', ylabel='count'>



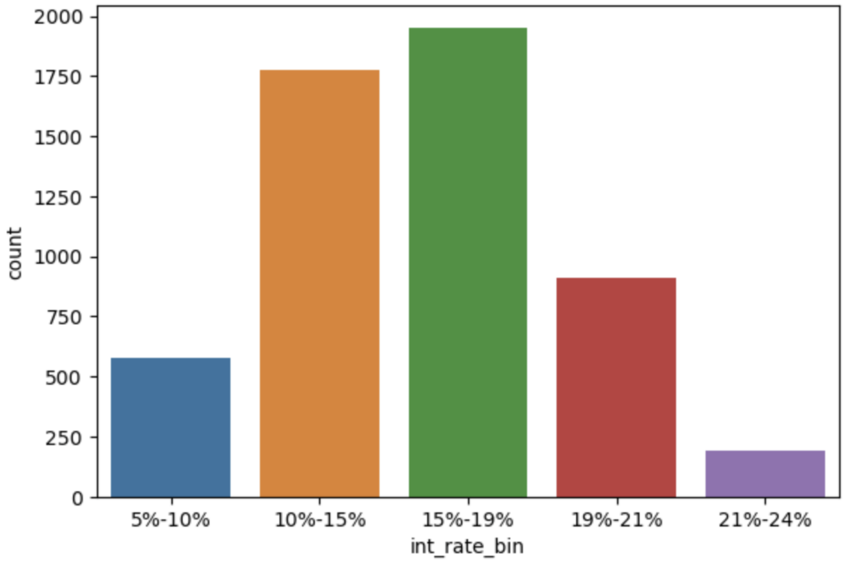
[167]:
```
# Obs-2: There are more 2-10 open credit lines when compared to others
# So, there is more probability of defaulting when the open credit lines falls between 2-10
```

Bivariate analysis(cont..)

- “int_rate_bin” vs “charged off loan status”
- It is clear that interest rate between 15-19% is more and the probability of getting defaulted is more for those interest rate.

```
[164]: # BiVariate Analysis
       # Now we can analyze the bins created with the actual columns
       # Let's first start with interest rate bin vs interest rate
       fig, ax = plt.subplots(figsize = (15,10))
       plt.subplot(221)
       sns.countplot(x='int_rate_bin', data=loan_csv[loan_csv.loan_status == 'Charged Off'])
```

```
[164]: <Axes: xlabel='int_rate_bin', ylabel='count'>
```



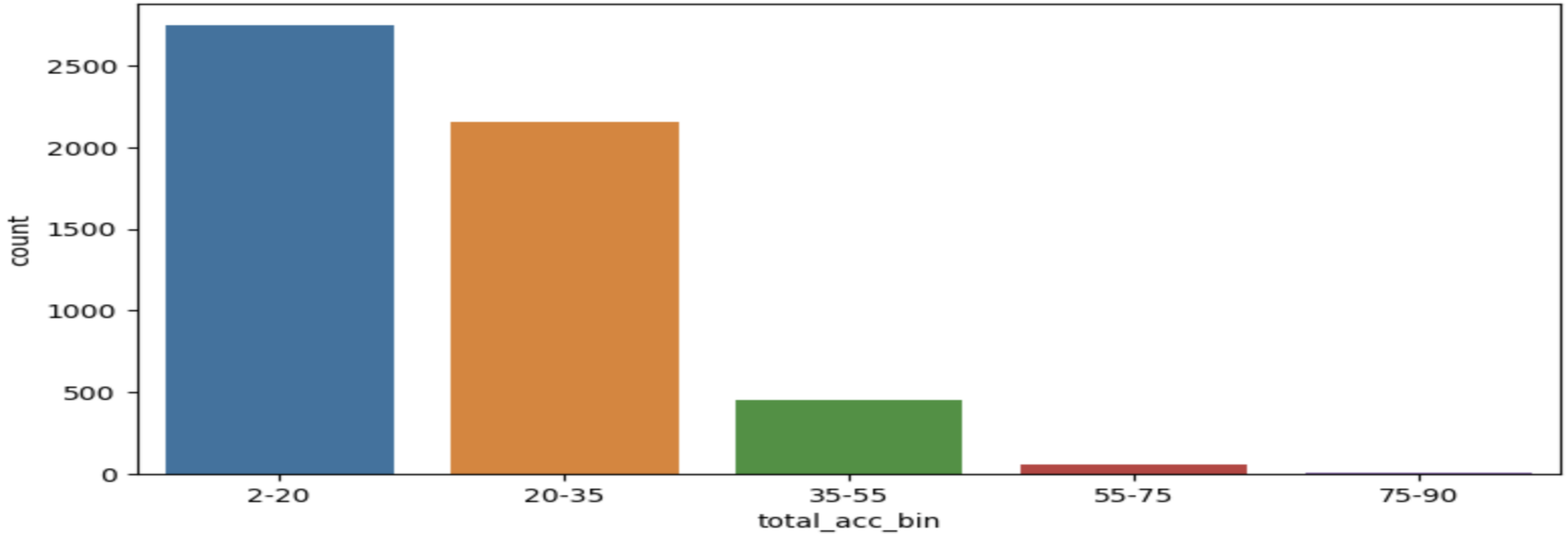

```
[165]: # Obs-1: So, the analysis from the above plot is that there are more people whose fall under the interest rate 15-19
       # So, there is more probability of defaulting when the interest_rate falls under 15-19%
```

Bivariate analysis(cont..)

- "total_acc_bin" vs "charged off loan status"
- There is more probability of defaulting when the total_acc (number of credit lines) is between 2-20

```
[170]: # # 'total_acc_bin' vs 'total_acc'
       fig, ax = plt.subplots(figsize = (20,10))
       plt.subplot(221)
       sns.countplot(x='total_acc_bin', data=loan_csv[loan_csv.loan_status == 'Charged Off'])

[170]: <Axes: xlabel='total_acc_bin', ylabel='count'>
```



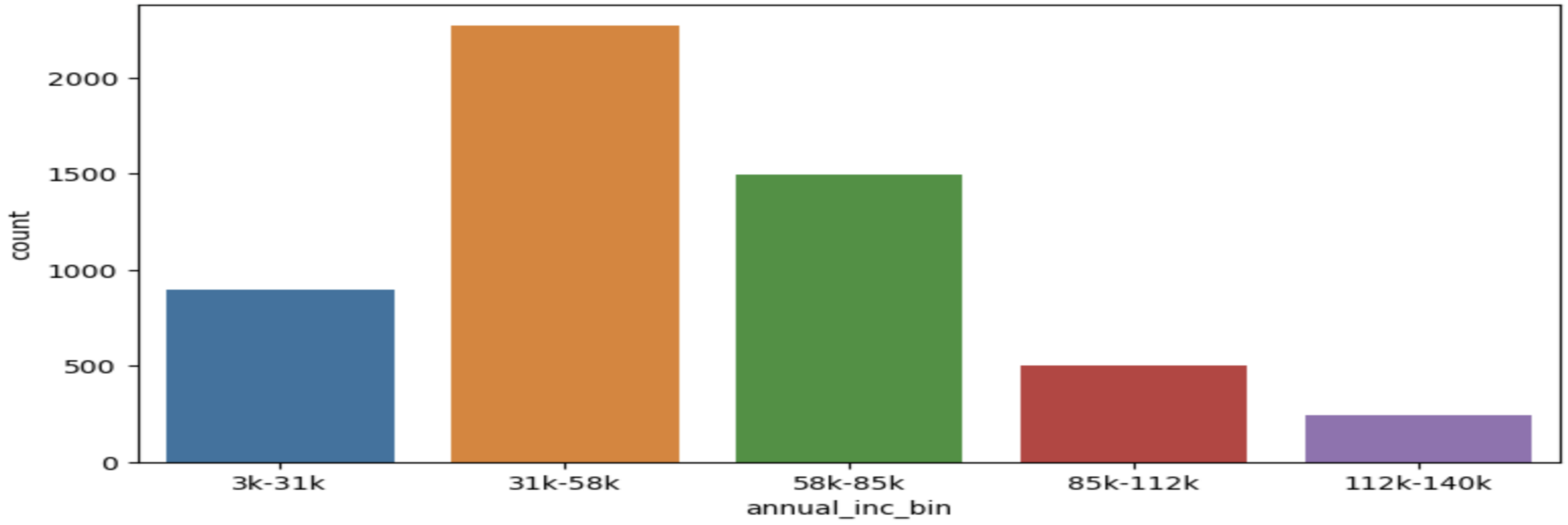

```
[171]: # Obs-4: There is more probability of defaulting when the total_acc (number of credit lines) is between 2-20
```

Bivariate analysis(cont..)

- “annual_inc_bin” vs “charged off loan status”
- There is more probability of defaulting when the annual income is between 58k-85k

[172]:
```
# 'revol_util' vs 'revol_util_bin'
fig, ax = plt.subplots(figsize = (20,10))
plt.subplot(221)
sns.countplot(x='annual_inc_bin', data=loan_csv[loan_csv.loan_status == 'Charged Off'])
```

[172]: <Axes: xlabel='annual_inc_bin', ylabel='count'>



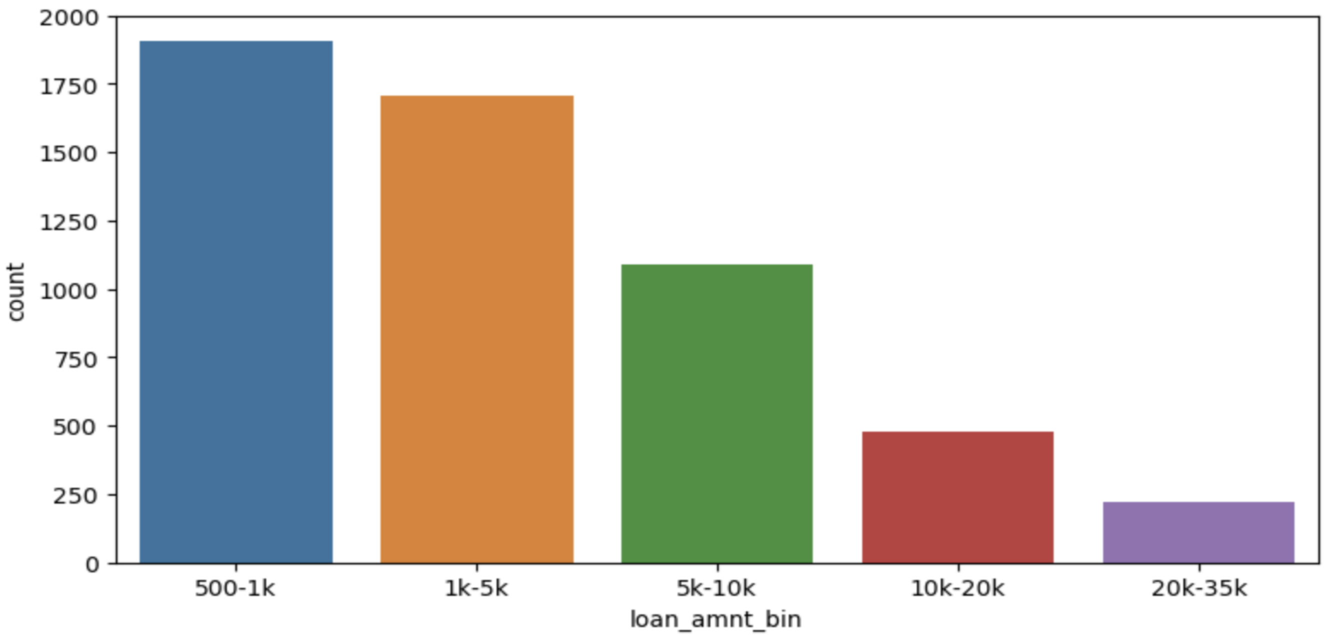
[173]:
```
# Obs-5: There is more probability of defaulting when the annual income is between 58k-85k
```

Bivariate analysis(cont..)

- “loan_amnt_bin” vs “charged off loan status”
- People with loan amount 500-1k will be defaulted mostly.

[176]:
```
# annual_inc_bin plot
fig, ax = plt.subplots(figsize = (20,10))
plt.subplot(221)
sns.countplot(x='loan_amnt_bin', data=loan_csv[loan_csv.loan_status == 'Charged Off'])
```

[176]:
```
<Axes: xlabel='loan_amnt_bin', ylabel='count'>
```



[177]:
```
# Obs-6: From the above plot it is obvious that people with loan amount 500-1k will be defaulted mostly
```

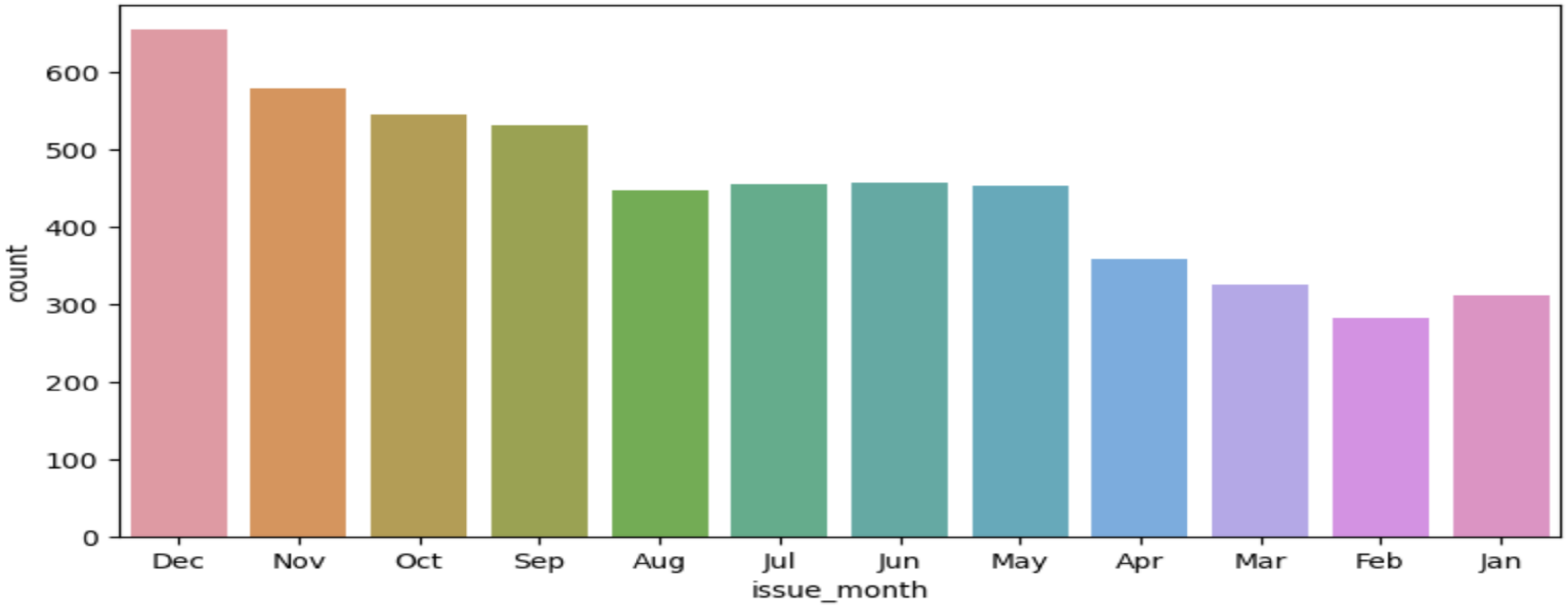
Bivariate analysis(cont..)

- “issue_d” vs “charged off loan status”
- The probability of getting charged off is more for the month of December and 2011 had mostly defaulted cases.

```
[181]: # Obs 7: So, the no of loans which has got charged off has been increasing gradually over year-on-year

       # issue_month analysis
       fig, ax = plt.subplots(figsize = (20,10))
       plt.subplot(221)
       sns.countplot(x='issue_month', data=loan_csv[loan_csv.loan_status == 'Charged Off'])
```

```
[181]: <Axes: xlabel='issue_month', ylabel='count'>
```




```
[182]: # Obs 8: The probability of getting charged off is more for the month of December
```

Key Observations and important results:

- There is more probability of defaulting when the interest_rate falls under 15-19%.
- There is more probability of defaulting when the open credit lines falls between 2-10.
- There is more probability of defaulting when the revol_util falls under 40-60.
- There is more probability of defaulting when the total_acc (number of credit lines) is between 2-20.
- There is more probability of defaulting when the annual income is between 58k-85k.
- People with loan amount 500-1k will be defaulted mostly.
- The no of loans which has got charged off has been increasing gradually over year-on-year.
- 2011 had most no of charged off loan_status.

Key Observations and important results:

- The probability of getting charged off is more for the month of December.
- The loan purpose stated as “home_improved” has been charged off more when compared to others mentioned loan purposes.
- Applicants with less salary has applied for education and moving loan.
- Applicants with mortgage as “home ownership” is more likely to be defaulted.
- The higher the salary range the more the interest rate.
- The higher the interest rate the higher the probability of getting defaulted.
- People with loan_amnt range 5k-10k is having interest rates between 12.5-15%