upGrad
#LifeKoKaroLift

Lending Club Case Study:
Pre-Assignment Session

1

Suman & Pruthvi consumer finance company

Company's Description:

Here we take loan applications for all type of loans(Personal/Car/Computer/Education/Credit card ETC). Analyse the RISK based on the customer metrics. Submit the application to the investor along with the FUND_AMOUNT.

Company's Goal:

Provide details about who can go as defaulters based on DRIVER variables or factors.

Here we also analyse the amount of risk to reduce/avoid the CREDIT LOSS to the COMPANY.

Case Study:

Based on HISTORIC DATA we get from LENDING CLUB. Going to analyse the data and provide some interesting patterns.

LC Data comprising LOAN records from borrowers during 2007 till 2011.

Step: 1:

Data Sourcing:

We will be using LC provided Data for analysing.

Step: 2:

Data Understanding:
Understand all the columns by going through Data Dictionary provided by LC.
Example Columns:
Loan_status: This would be main column analysing defaulters and non-defaulters.
Funded_amnt_Investor: This column says the amount of fund sanctioned to the applicant.

Step: 3:

Data Cleaning:

Check the percentage of columns that are high with missing values based on columns.
Columns with higher % of missing values will be removed from the analysing DF.
Analyse the columns which are redundant and not useful. Such columns will also be removed.
After removing the redundant and not useful columns . Once check the % of missing columns again to make sure that DF is clean.

Fixing Columns:

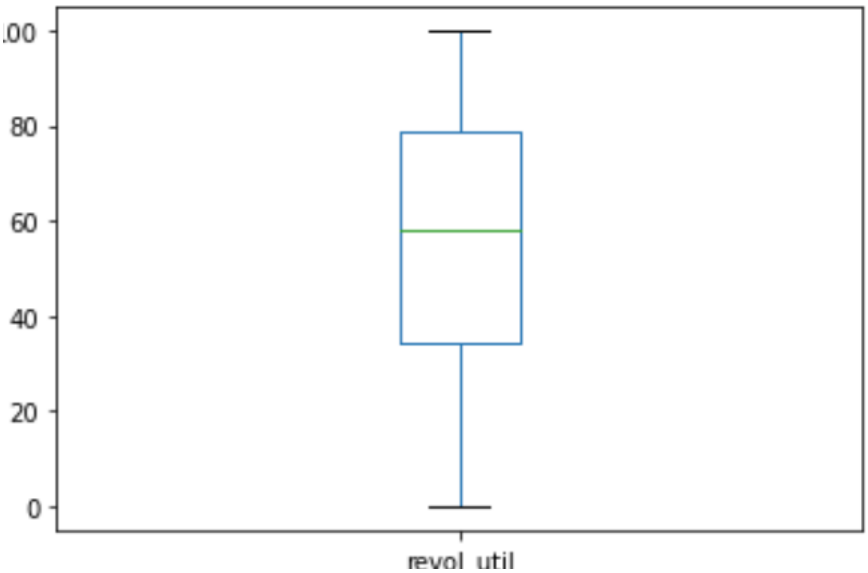
Example:

```
0]: # Dropping ID column as its redundant
df1 = df1.drop(columns='id')
# Removing columns :"loan_amnt" & "funded_amnt" since the column:"funded_amnt_inv" is the one which applicant is going
df1.drop(columns=['loan_amnt','funded_amnt'],inplace=True)
#Payment plan has all same values as "n" and of no use
df1.drop(columns=['pymnt_plan'],inplace=True)
#URL and Desc dosent have any relacvent values for out analysis.
df1.drop(columns=['url','desc'],inplace=True)
#Title is redundant with purpose and title values are too generic and hence dropping
df1.drop(columns=['title'],inplace=True)
#For sanctioning loans delinquished data is of no use.Hence dropping
df1.drop(columns=['delinq_2yrs'],inplace=True)
#Revolving util has percentage about customer behavior variables.
df1.drop(columns=['revol_bal'],inplace=True)
# Since this column has redundant values as :"f". Not usefull column to retaain .Hence dropping
df1.drop(columns=['initial_list_status'],inplace=True)
# Most of the elements are EMPTY . We cannot impute or compute using any derivative metrics.Hence dropping.
df1.drop(columns=['next_pymnt_d'],inplace=True)
# All the value in these columns are ZERO or N/A not a usefull column. Hence dropping
df1.drop(columns=['collections_12_mths_ex_med'],inplace=True)
#Both the columns having redundant values and not usefull.Hence dropping .
df1.drop(columns=['policy_code','application_type'],inplace=True)
```

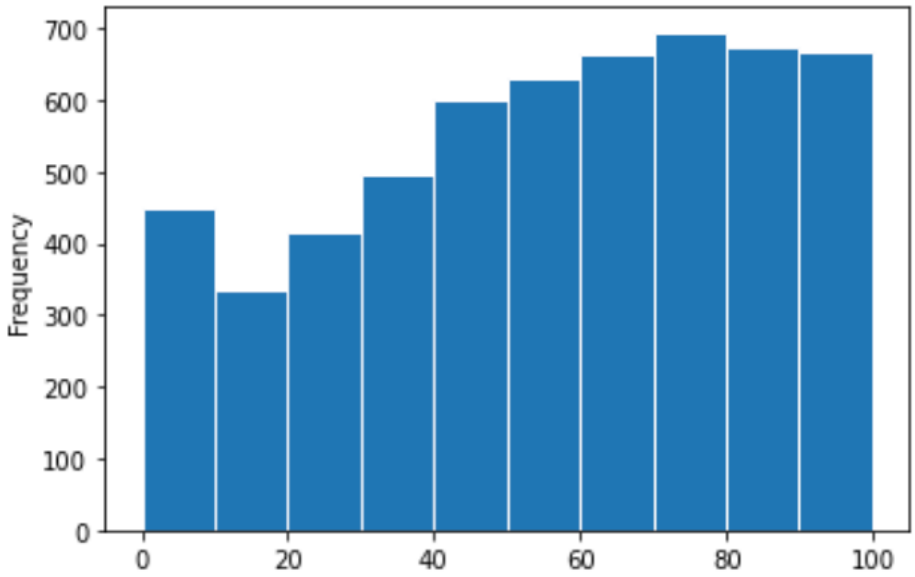
Step :4

Univariate Analysis:

1. To make sure there are no any outliers with 'Revol_util' wrt charged-off applicants. Using BOX plot as below:



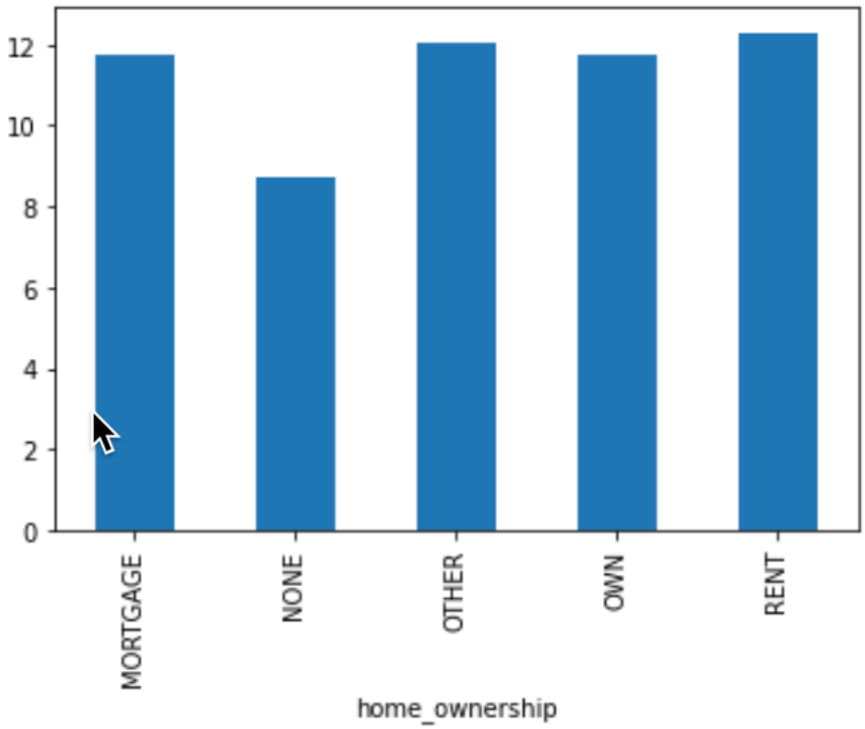
2. Based on the above HISTOGRAM plot its evident that the number of "charged-off" applicants increases with "revol_util" % increment.



Step: 5

Bivariate Analysis

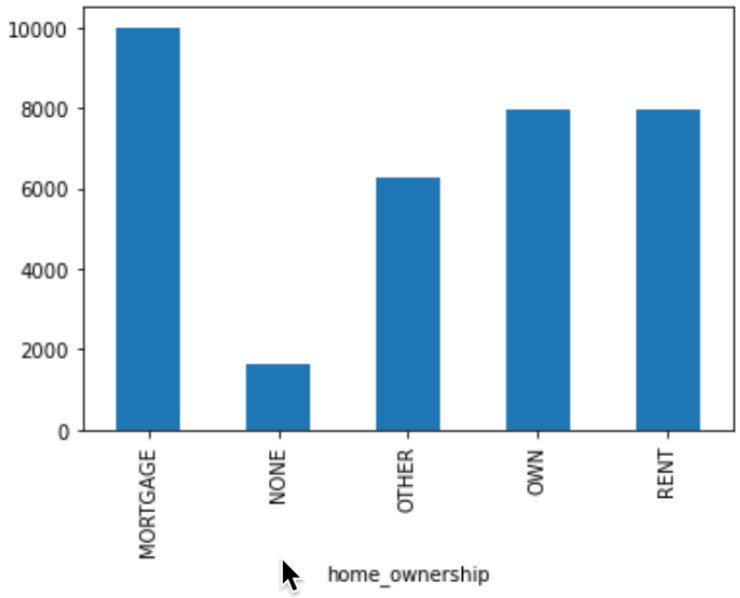
1. The LOAN amount and the INTEREST rates given to NONE category applicants are less on an average, compared with other category applicants.



2. When performing deep dive into the data to see NONE applicants . It observed that there are only THREE applicants part of that category.
   Instead of considering this type , its better to ignore as an OUTLIER's category.

```
df1.groupby(['home_ownership'])['funded_amnt_inv'].median().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa714db5c40>
```




```
df1[df1.home_ownership=='NONE']
```

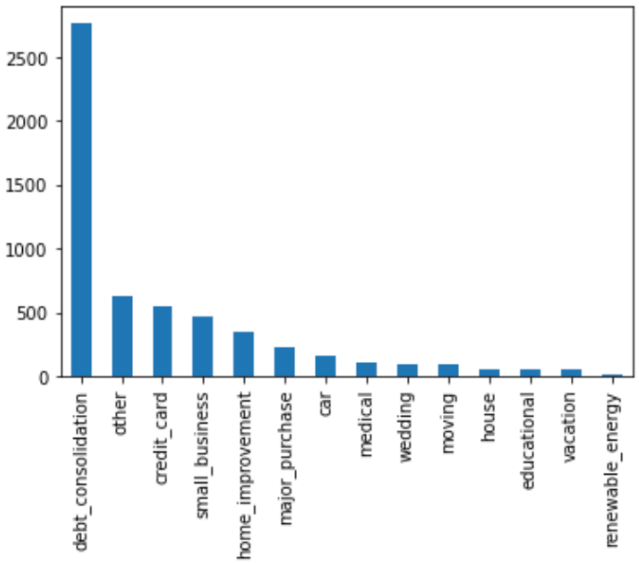
	member_id	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	...	recoveries	collection_recovery_
39318	214993	1228.06	36	7.75	312.22	A	A3	ups	5.0	NONE	...	0.0	
39659	121574	1925.00	36	9.64	128.41	B	B4	NaN	0.9	NONE	...	0.0	
39660	121373	1625.00	36	8.70	88.65	B	B1	NaN	0.9	NONE	...	0.0	

3 rows × 41 columns

3. Below analysis gives the information about how the purpose of loan is influencing the number of charged-off applicants.

```
# Plotting the bar graph ideltifying the spread of driving_factor:purpose
df2.purpose.value_counts().plot.bar()
```

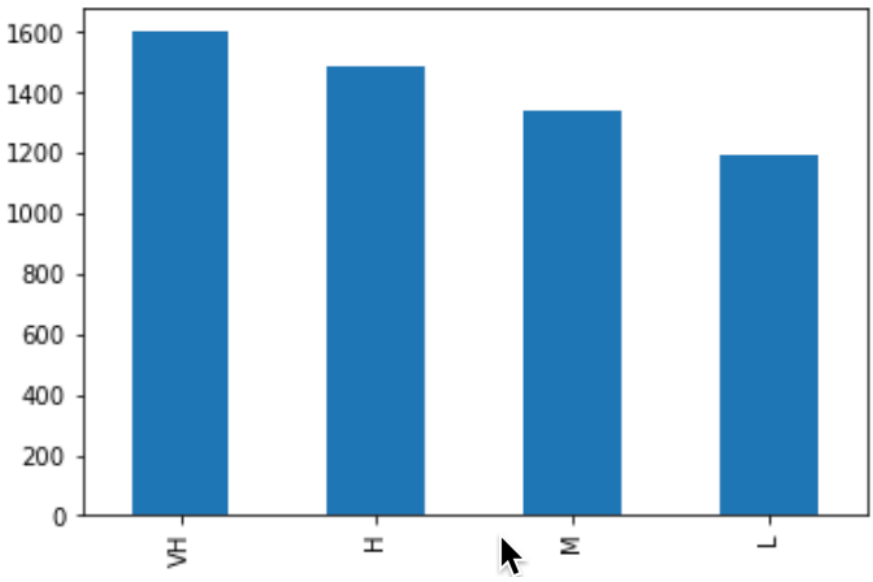
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa715b127c0>
```



4. As per the above PLOT the probability of charged-off applicants is increasing along with the raise in DTI ratio.

```
df2.dti_bucket.value_counts().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa7152fc1f0>
```



Step : 6

Standardising Values:

1. Fixing the “int_rate” by removing the % symbol and converting from object type to float as_ type.

2. Fixing the date column form strig to data format . This can help for derived metrics.

3. Modifying the columns to INT or FLOAT . Which can help in analysing using aggrfunc

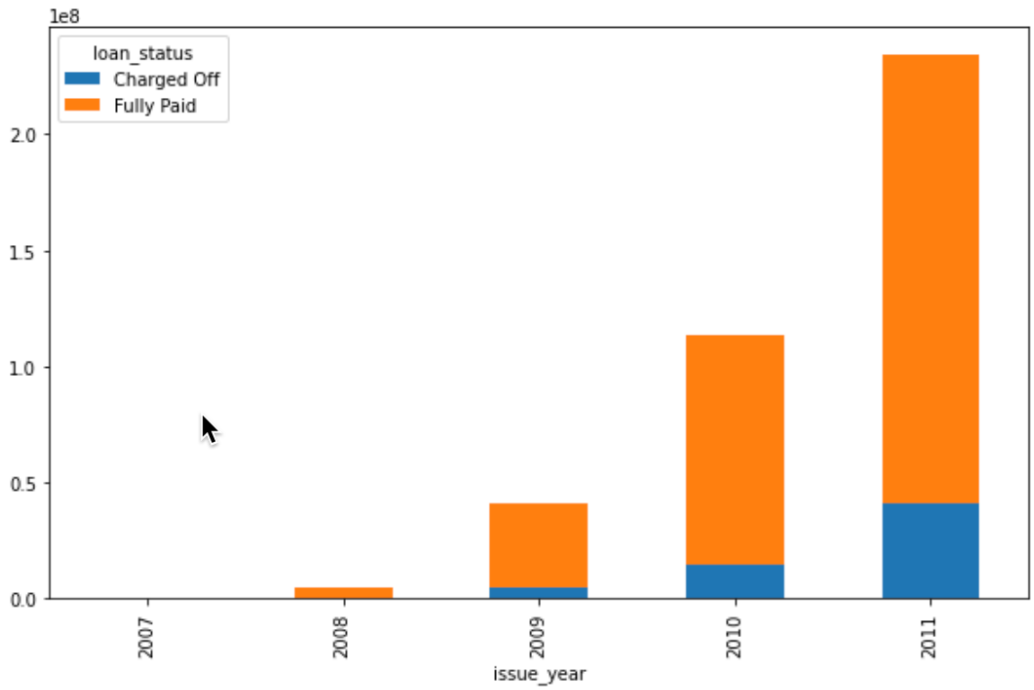
Step : 7

Multivariate Analysis

1. Observed that the LOAN amount is raising every year and charged-off and fully paid also raising equivalent. Charged off applicants are close to 1/4th of the fully paid applicants during 2011.Which is HIGH compared to other years.

```
: #Plottig a stacked plot on X DF.
  X.plot(kind="bar", stacked="True", figsize=[10,6])
```

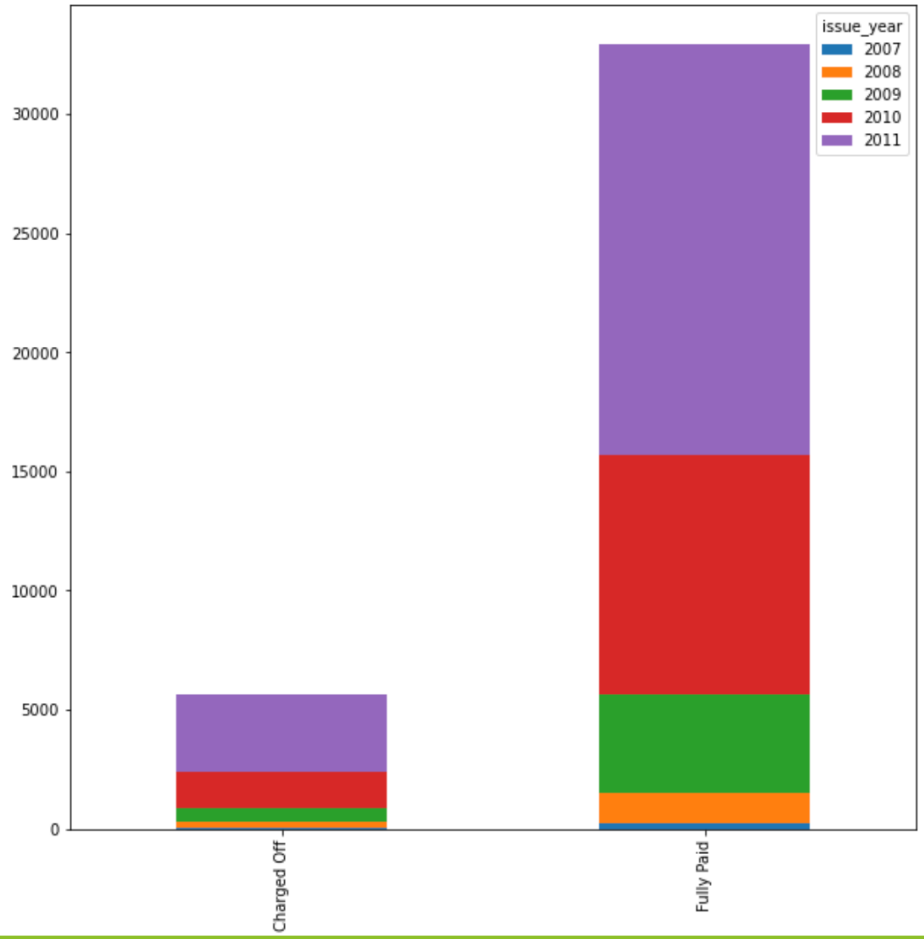
```
: <matplotlib.axes._subplots.AxesSubplot at 0x7fa718a62100>
```



2. Its observed that the number of applicants are raising every year which is +ve sign. Above analysis 4:1 is represented here with number of applicant's distribution.2011 has more charged-off and fully-paid based on number of applicants.

```
Y.plot(kind='bar',stacked='True',figsize=[10,10])
#Plottig stacked plot
```

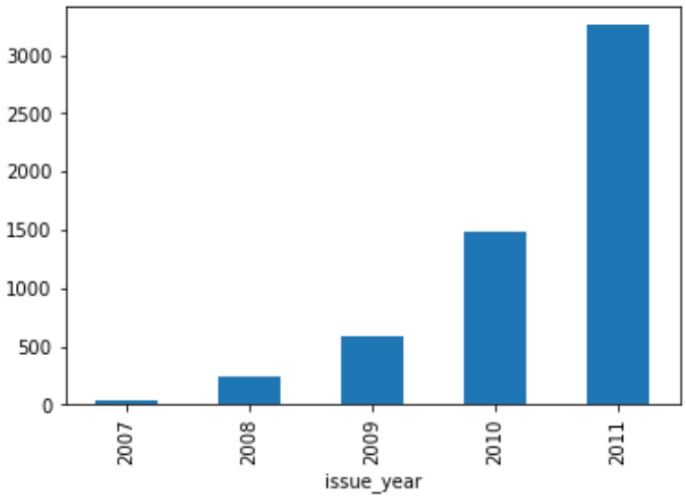
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa71ae0c6a0>
```



3. The number of charged-off applicants are evidenly clear that, They are growing every year more than 100%.

```
#Plotting a graph only to analyse charged-off applicants based on YEAR
Y.loc['Charged Off'].plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa71ac94af0>
```



Step : 8

Derived metric:
Deriving a new derived variable('issue_year') of type "INTERVAL" variable from ('issue_d') column.

upGrad
#LifeKoKaroLift

Thank You!