Lending Club Case Study

GARV DAGA

PUSHPENDRA HIRWANI

Problem Statement

Understanding the driving factors (or driver variables) behind loan default, i.e. the variables which are strong indicators of default. These variables can be then utilized by the company for its portfolio and risk assessment.

Approach

- Data Cleaning
  1. Remove the columns and Rows based on Business Understanding.
  2. Fix Column Values and Data Types .

- Derived Metrics
  1. Converting dates to Month and Year.
  2. Finding ratios and percentage.
- 3. Derive columns using Buckets.

- Univariate Analysis

- Bivariate Analysis

Data Cleaning

04

1. Remove the columns and Rows based on Business Understanding.

- Removed columns having all null values.
- Removed rows having loan_status as 'Current' as it is not required for this analysis.
- Removed columns having all same or, all unique values.
- Removed columns not relevant to the problem statement and requiring more data processing to give insights.
- Removed rows with null values as the number of such records were less.

```
# Drop fields having all NA Values
data = data.dropna(axis=1, how='all')

# Determines and drops columns which have all unique values
def drop_unique_cols():
    uniq_cols = [col for col in data.columns if data[col].nunique() == len(data.index)]
    print("Columns having all unique : ", uniq_cols)
    for cols in uniq_cols:
        del data[cols]

# Determines and drops columns having only 1 unique value
def drop_cols_having_only_1_unqiue_value():
    cols_having_only_1_unqiue_value = [col for col in data.columns if data[col].nunique() == 1]
    print("Columns having only 1 unique values: ", cols_having_only_1_unqiue_value)
    for cols in cols_having_only_1_unqiue_value:
        del data[cols]

drop_unique_cols()
drop_cols_having_only_1_unqiue_value()

# Drop columns requiring more processing or are not relevant for the given problem statement
data = data.drop(['desc', 'emp_title','title'], axis=1)
data = data.drop(['last_pymnt_d', 'earliest_cr_line', 'total_rec_int', 'total_rec_late_fee', 'total_rec_prncp'], axis=1)

# Drop mths_since_last_delinq which is having more than 65% NA values
data.drop(['mths_since_last_delinq'], axis=1, inplace=True)

# Drop all rows having NA values for below columns as they are small in number
data.dropna(subset=['pub_rec_bankruptcies','revol_util','last_credit_pull_d','emp_length'], axis=0, inplace=True)
data.info()
```

Data Cleaning

05

2. Fix Column Values and Data Types .

- Numerical values should take int or float, whereas categorical can even take object.
- Convert column having % to float
- Convert Date fields from Object to DateTime.
- Note we are not converting term to numerical as the data has only two possible values and is categorical.
- Remove Outliers and Duplicate Data if any.

```python
data['int_rate'] = data['int_rate'].str.rstrip('%').astype('float')
data['revol_util'] = data['revol_util'].str.rstrip('%').astype('float')

data = data.astype({ "loan_amnt": float, "funded_amnt": float, "funded_amnt_inv": float, "annual_inc": float})

data['issue_d'] = pd.to_datetime(data['issue_d'], format='%b-%y')

data.select_dtypes('object').apply(pd.Series.nunique, axis = 0)
```

```python
# Remove outliers using IQR for loan amount
loan_amnt_iqr = data.loan_amnt.quantile(0.75) - data.loan_amnt.quantile(0.25)
loan_amnt_lower_bound = data.loan_amnt.quantile(0.25) - 1.5*loan_amnt_iqr
loan_amnt_upper_bound = data.loan_amnt.quantile(0.75) + 1.5*loan_amnt_iqr

data = data[(data.loan_amnt >= loan_amnt_lower_bound) & (data.loan_amnt <= loan_amnt_upper_bound)]

# Remove outliers using IQR for funded amount invested
funded_amnt_inv_iqr = data.funded_amnt_inv.quantile(0.75) - data.funded_amnt_inv.quantile(0.25)
funded_amnt_inv_lower_bound = data.funded_amnt_inv.quantile(0.25) - 1.5*funded_amnt_inv_iqr
funded_amnt_inv_upper_bound = data.funded_amnt_inv.quantile(0.75) + 1.5*funded_amnt_inv_iqr

data = data[(data.funded_amnt_inv >= funded_amnt_inv_lower_bound) & (data.funded_amnt_inv <= funded_amnt_inv_upper_bound)]

# Remove outliers using IQR for funded amount invested
annual_inc_iqr = data.annual_inc.quantile(0.75) - data.annual_inc.quantile(0.25)
annual_inc_lower_bound = data.annual_inc.quantile(0.25) - 1.5*annual_inc_iqr
annual_inc_upper_bound = data.annual_inc.quantile(0.75) + 1.5*annual_inc_iqr

data = data[(data.annual_inc >= annual_inc_lower_bound) & (data.annual_inc <= annual_inc_upper_bound)]
data.shape
```

Derived Metrics

1. Converting dates to Month and Year will help in analysis

```python
data['issue_d_month'] = pd.to_datetime(data['issue_d'], format='%b-%y').dt.month
data['issue_d_year'] = pd.to_datetime(data['issue_d'], format='%b-%y').dt.year
data['last_pymnt_d_month'] = pd.to_datetime(data['last_pymnt_d'], format='%b-%y').dt.month
data['last_pymnt_d_year'] = pd.to_datetime(data['last_pymnt_d'], format='%b-%y').dt.year
data.head()
```

[17] ✓ 0.1s

2. % of Amount funded by Investor to the Amount Asked by Borrower

```python
data['funded_amnt_%'] = data['funded_amnt_inv']/data['funded_amnt'] *100
data.head()
```

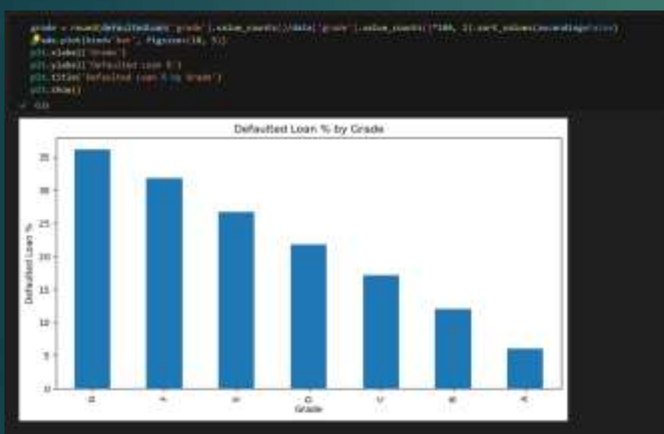
[18] ✓ 0.0s

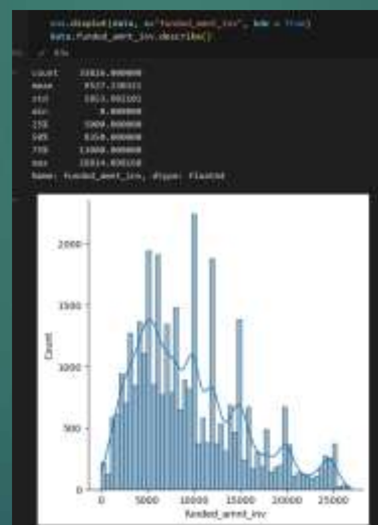
3. Derive Columns using Buckets

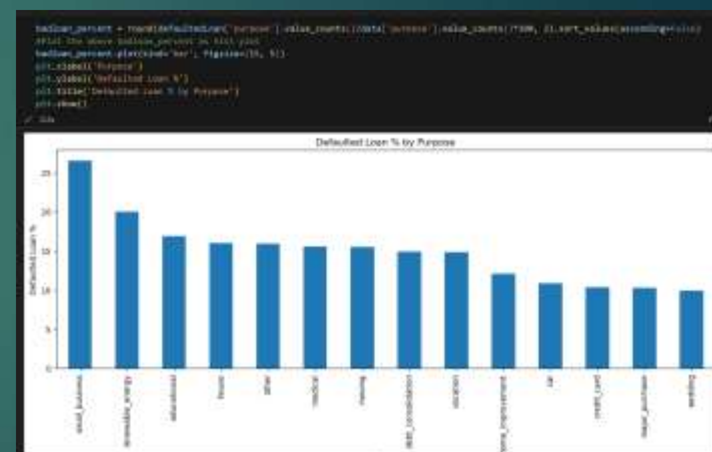
```python
# Convert dti into a categorical variable
def dti_bucketing(dti):
    if (dti <= 8):
        return 'VERY LOW'
    if (dti > 8 and dti <=13):
        return 'LOW'
    if (dti > 13 and dti <=17):
        return 'MEDIUM'
    if (dti >= 17 and dti < 19):
        return 'HIGH'
    return 'VERY HIGH'
data['dti_bucket'] = data.dti.apply(dti_bucketing)

# Convert interest rate into categorical variable:
int_rate_25_quantile = int(data.int_rate.quantile(0.25))
int_rate_50_quantile = int(data.int_rate.quantile(0.5))
int_rate_75_quantile = int(data.int_rate.quantile(0.75))

def bin_int_rate(int_rate):
    if(int_rate <= int_rate_25_quantile):
        return "LOW"
    if(int_rate > int_rate_25_quantile and int_rate <= int_rate_50_quantile):
        return "MEDIUM"
    if(int_rate > int_rate_50_quantile and int_rate <= int_rate_75_quantile):
        return "HIGH"
    return "VERY HIGH"

data['int_rates_bucket'] = data.int_rate.apply(bin_int_rate)
```

Univariate Analysis

1. The Risk increases with the increase of Grade.
2. We can say that the chances of bad loans is the most for small business.
3. Its evident that loan amount asked for and loan amount funded by investors at around multiples of 5000(5000, 10000, 12500, 15000, 20000, 25000).

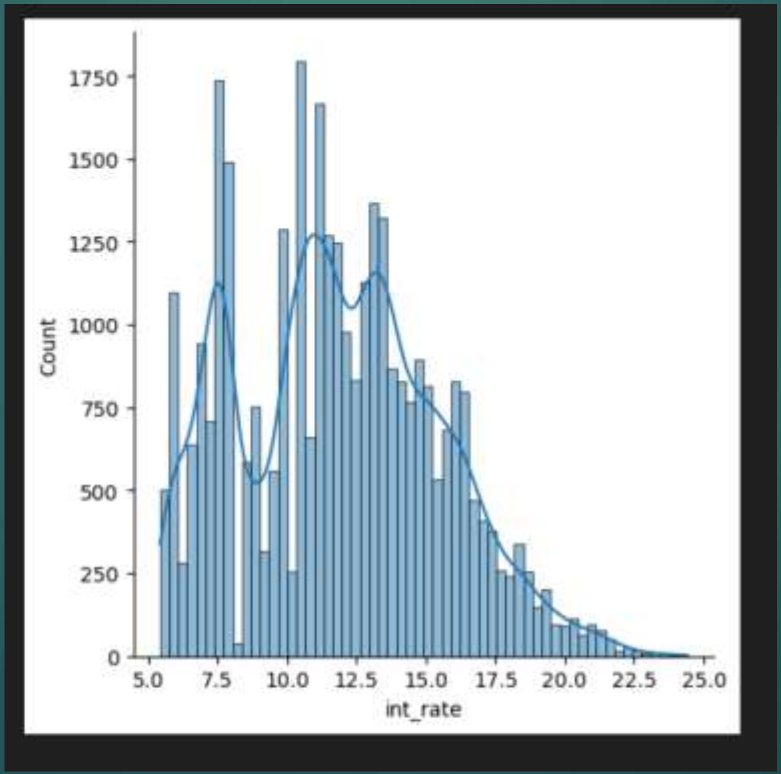






Univariate Analysis

4. 50% of folks are disbursed loan amount at around 8200
5. A lot of folks are issued interest at ~8% and between (10 - 14.5)%
6. Very less people are issued interest between (8-10)%
7. 50% of folks are issued loan at interest rate ~11.5%



Bivariate Analysis

09

1. An individual with home ownership status as "OTHER" and they generally take a higher loan amount and there is a very high tendency of their loan being "Charged off".
2. Loan amount value is inversely proportional to amount of folks asking for that particular loan amount.
3. As the Annual income increases the tendency of some to default decreases
4. The higher the Interest rates higher are the chances of an individual to default on the loan which suggests there is a higher likelihood of loan getting defaulted when loan is disbursed with higher interest rates.

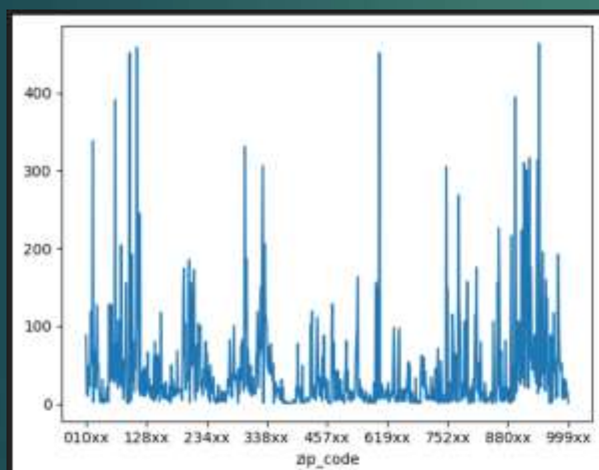
```
data.groupby(["home_ownership"])['loan_amnt'].mean()
✓ 0.0s

home_ownership
MORTGAGE     11146.602111
OTHER        10637.500000
OWN           9453.203343
RENT          9422.616726
Name: loan_amnt, dtype: float64
```

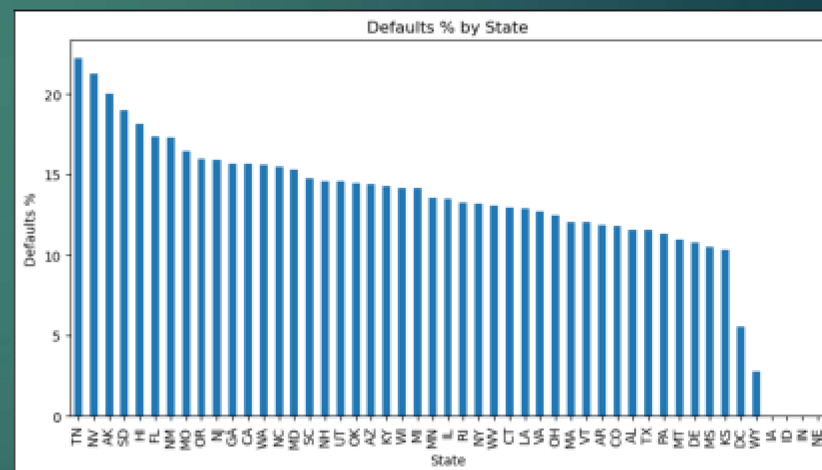
```
round(data[data['loan_status'] == 'Charged Off'].groupby(["int_rates_bucket"])['loan_status'].count()
    /data.groupby(["int_rates_bucket"])['loan_status'].count()*100,2)
✓ 0.0s

int_rates_bucket
HIGH         14.98
LOW           5.25
MEDIUM       10.22
VERY HIGH    23.92
Name: loan_status, dtype: float64
```

```
round(data[data['loan_status'] == 'Charged Off'].groupby(["annual_inc_category"])['loan_status'].count()
    /data.groupby(["annual_inc_category"])['loan_status'].count()*100,2)
✓ 0.0s

annual_inc_category
0-40000        17.64
40000-55146    14.88
55146-75821    13.65
>75821         10.70
Name: loan_status, dtype: float64
```

Bivariate Analysis

5. 2007, followed by 2011 and 2008 experienced most number of people with charged off loans, the amount of fully paid loan also increased, This is probably as after effect of recession happened
6. Folks with zipcode ranging from 010xx-12xx, living around 338xx or 754xx, 622xx, 882xx-90xx have higher tendency of defaulting the loans
7. We can infer that loans from States 'TN' , 'NV' and 'AK' have higher probability of Defaults.
8. As the dti keeps on increasing the Charge off percentage which suggests there is a higher likelihood of loan getting defaulted when loan is disbursed to individual with higher dti






```
dti_bucket
HIGH         15.30
LOW          13.49
MEDIUM       14.76
VERY HIGH    16.14
VERY LOW     12.23
Name: loan_status, dtype: float64
```

Bivariate Analysis

9. When grouped by grade, subgrades, It's found that as we move to lower grades the Charge off proportions keeps on increasing which suggests there is higher chance of loans being defaulted when disbursed to individual with lower employment grades

```
round(data[data['loan_status'] == 'Charged Off'].groupby(["grade", "sub_grade"])['loan_status'].count()
    /data.groupby(["grade", "sub_grade"])['loan_status'].count()*100,2)
```

✓ 0.0s

```
grade  sub_grade
A      A1           2.41
       A2           4.91
       A3           5.26
       A4           6.22
       A5           8.24
B      B1           9.15
       B2          10.78
       B3          11.91
       B4          13.41
       B5          13.83
C      C1          15.79
       C2          16.47
       C3          18.50
       C4          17.64
       C5          18.25
D      D1          17.84
       D2          20.81
       D3          22.24
       D4          23.20
       D5          25.67
E      E1          26.84
       E2          25.70
       E3          23.61
       E4          29.88
       E5          29.00
F      F1          27.59
       F2          29.61
       F3          30.08
       F4          33.61
```