Lending Club Case Study

Conclusion

Analysis

Analysis

Data

Problem

Team

LendingClub

Conclusion

Analysis

Analysis

Data

Problem

1. Kunj Lal
2. Navaneeth S Rao

Team

Conclusion

Analysis

Analysis

Data

Can we analyze the past loan data of Lending Club and identify the risky loan applicants who are prone to default?

Problem

Team

Data Cleaning

- The **loan.csv** data has a total of 39,717 rows and 111 columns.

- Only a few columns from this data help us in analyzing and coming up with the defaulter predictions.

Step 1: Drop the column with only NULL values

```
#Check the total no. of rows and columns with NULL values
print("No. of empty rows in the data set:")
print(loan_data.isnull().all(axis = 1).sum())

print("No. of empty columns in the data set:")
print(loan_data.isnull().all(axis = 0 ).sum())
```

```
No. of empty rows in the data set:
0
No. of empty columns in the data set:
54
```

- Totally 54 columns have NULL values and can be dropped.

Data Cleaning(Contd.)

Step 2: Dropping customer behaviour variables.

- These variables are not available at the time of application, and thus they cannot help in our analysis.

b) Dropping customer behavior variables

- Customer behavior variables are not available at the time of loan application, and thus they cannot help in our predictive analysis

```
loan_data.drop(['delinq_2yrs','earliest_cr_line','inq_last_6mths', 'open_acc', 'pub_rec', 'revol_bal', 'revol_util','total_acc',
```

```
# Check the shape now
print(loan_data.shape)
```

```
(39717, 36)
```

Now the number of columns have dropped from 57 to 36

- Now the numbers of columns have reduced to 36.

Data Cleaning(Contd.)

Step 3: Dropping additional columns that don't offer much to our analysis.

- Some of the key columns are "id", "member_id", "initial_list_status", "mths_since_last_delinq", "title" etc.

c) Dropping additional columns as they don't offer much to our analysis ¶

```
In [950]: loan_data.drop(['id', 'member_id', 'emp_title', 'mths_since_last_delinq', 'mths_since_last_record', 'initial_list_status', 'next_

In [951]: loan_data.shape
Out[951]: (39717, 17)
```

- Now the number of columns has reduced to 17.

Data Cleaning(Contd.)

Step 4: Check for missing values in the remaining columns.

- Emp_length column has 1075 missing values.
- The missing values will be imputed with the mode of the data.

```
In [955]: loan_data.isnull().sum()

Out[955]: loan_amnt                 0
          funded_amnt               0
          funded_amnt_inv           0
          term                      0
          int_rate                  0
          installment               0
          grade                     0
          sub_grade                 0
          emp_length             1075
          home_ownership            0
          annual_inc                0
          verification_status       0
          issue_d                   0
          loan_status               0
          purpose                   0
          addr_state                0
          dti                       0
          dtype: int64
```

```
In [956]: # Replace the missing values with mode
          loan_data.emp_length.fillna(loan_data.emp_length.mode()[0], inplace = True)

          #Check if any values are missing still
          loan_data.emp_length.isnull().sum()

Out[956]: 0
```

Conclusion Analysis Analysis Data Problem Team

Data Cleaning(Contd.)

Step 5: Standardize the data.

- emp_length column field should contain only numbers (need to remove <, +, years etc.
- Int_rate column should only contain numbers (need to remove %)

```
In [957]: #Convert to a numerical column
          loan_data['emp_length']=loan_data.emp_length.str.extract('(\d+)')

In [958]: #Display the emp_length column
          loan_data['emp_length']

Out[958]: 0        10
          1         1
          2        10
          3        10
          4         1
                   ..
          39712     4
          39713     3
          39714     1
          39715     1
          39716     1
          Name: emp_length, Length: 39717, dtype: object
```

```
In [959]: #Convert to a numerical column
          loan_data['int_rate'] = loan_data['int_rate'].str.rstrip('%')

In [960]: #Display the data
          loan_data['int_rate']

Out[960]: 0        10.65
          1        15.27
          2        15.96
          3        13.49
          4        12.69
                   ...
          39712     8.07
          39713    10.28
          39714     8.07
          39715     7.43
          39716    13.75
          Name: int_rate, Length: 39717, dtype: object
```

Data Cleaning(Contd.)

Step 5: Removing the current loan status data.

- Applicants who are still paying the loans cannot be considered for the analysis.

```
In [961]: loan_data = loan_data[loan_data.loan_status != "Current"]
          loan_data.loan_status.unique()

Out[961]: array(['Fully Paid', 'Charged Off'], dtype=object)

In [962]: loan_data.emp_length

Out[962]: 0        10
          1         1
          2        10
          3        10
          5         3
                   ..
          39712     4
          39713     3
          39714     1
          39715     1
          39716     1
          Name: emp_length, Length: 38577, dtype: object
```

Data Cleaning(Contd.)

Step 6: Converting the data type of int_rate variable.

- This will be used further in the analysis.

```
In [964]: loan_data.dtypes

Out[964]: loan_amnt                int64
          funded_amnt              int64
          funded_amnt_inv        float64
          term                    object
          int_rate                object
          installment            float64
          grade                   object
          sub_grade               object
          emp_length              object
          home_ownership          object
          annual_inc             float64
          verification_status     object
          issue_d                 object
          loan_status             object
          purpose                 object
          addr_state              object
          dti                    float64
          dtype: object
```

h) Converting the datatypes of variables

```
In [965]: loan_data['int_rate'] = loan_data['int_rate'].astype('float64')
```

Data Cleaning(Contd.)

Step 7: Deriving additional columns and binning the data.

- This will be used further in the analysis.

i) Deriving columns

In [966]:
```
# Derived columns

# categorise loan amounts into bins
loan_data['loan_amnt_category'] = pd.cut(loan_data ['loan_amnt'], [0, 7000, 14000, 21000, 28000, 35000], labels=['0-7000', '7000-

# categorise annual incomes into bins
loan_data['annual_inc_category'] = pd.cut(loan_data['annual_inc'], [0, 20000, 40000, 60000, 80000,1000000], labels=['0-20000', '2

# categorise intrest rates into bins
loan_data['int_rate_category'] = pd.cut(loan_data['int_rate'], [0, 10, 12.5, 16, 20], labels=['0-10', '10-13', '12.5-16', '16 +']

# categorise dti into bins
loan_data['dti_category'] = pd.cut(loan_data['dti'], [0, 5, 10, 15, 20, 25], labels=['0-5', '05-10', '10-15', '15-20', '25+'])
```

In [967]:
```
# Derived columns

# Lets create month and year columns separately

loan_data.issue_d = pd.to_datetime(loan_data.issue_d, format='%b-%y')
loan_data['year']=loan_data['issue_d'].dt.year
loan_data['month']=loan_data['issue_d'].dt.month
```

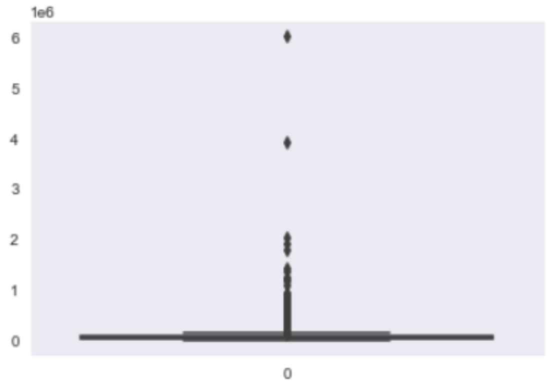
Data Cleaning(Contd.)

Step 8: Handling outliers.

- These variables could possibly have outliers: annual_inc, dti, loan_amnt, funded_amnt, funded_amnt_inv
- Box plots will help us in removing the outliers.

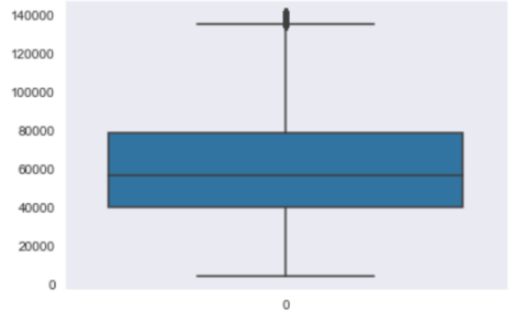
annual_inc

```
#Plot boxplot for annual_inc variable
sns.boxplot(loan_data['annual_inc'])
plt.show()
```




```
_95_percentile = loan_data['annual_inc'].quantile(0.95)
loan_data = loan_data[loan_data.annual_inc <= _95_percentile]
```

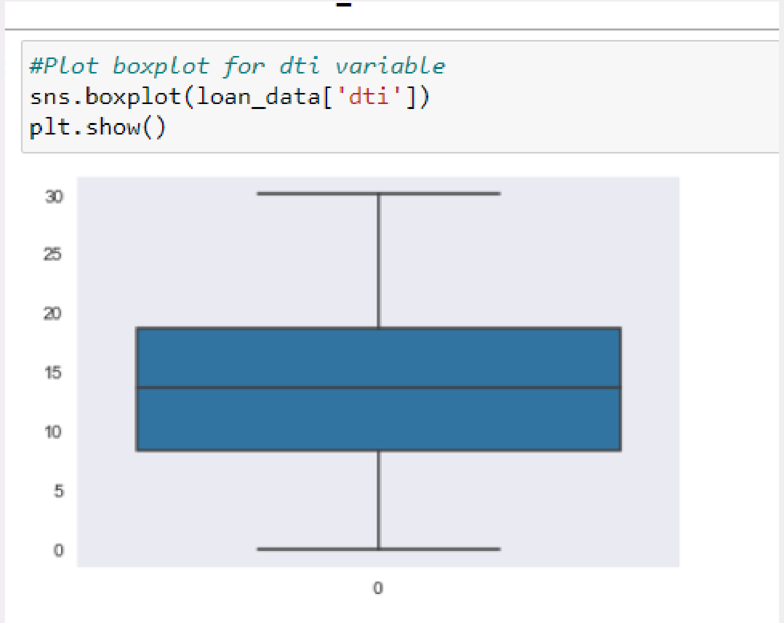
```
sns.boxplot(loan_data['annual_inc'])
plt.show()
```



- Values above 95th percentile are considered as outliers and removed from the data

Data Cleaning(Contd.)

dti

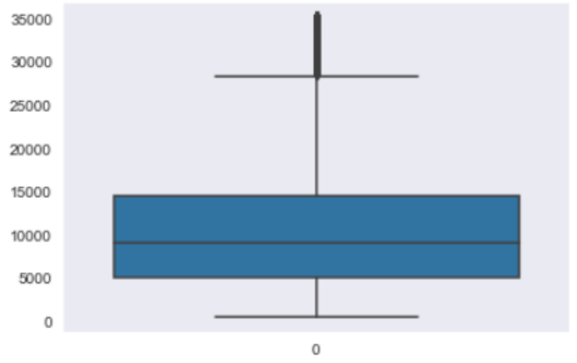


- There are no outliers in this variable.

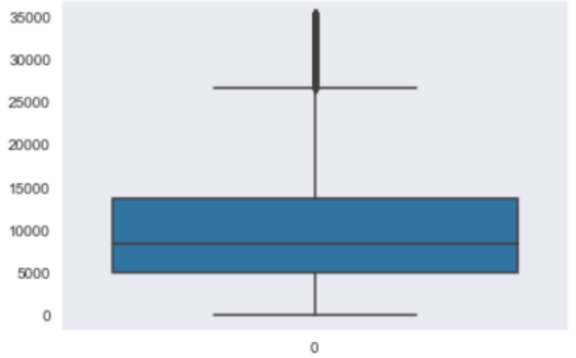
Data Cleaning(Contd.)

funded_amnt, funded_amnt_inv

```
#Plot boxplot for funded_amnt variable
sns.boxplot(loan_data['funded_amnt'])
plt.show()
```




```
#Plot boxplot for funded_amnt_inv variable
sns.boxplot(loan_data['funded_amnt_inv'])
plt.show()
```

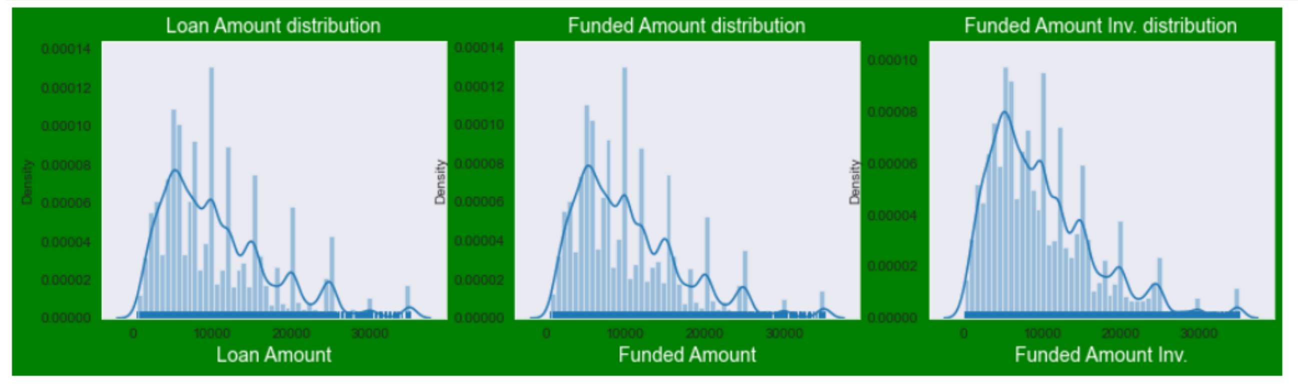


- There are very few outliers in these variables. Since the distribution is continuous we will keep these variables.

Univariate Analysis

1. Distribution plot of loan_amnt, funded_amnt, and, funded_amnt_inv.

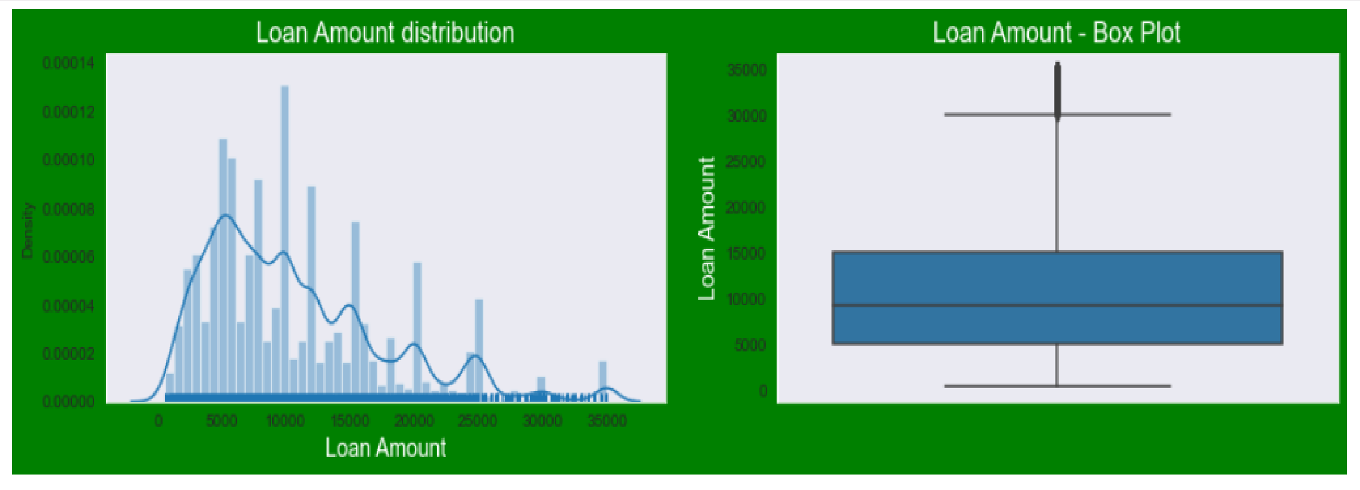
- The distribution of the amounts for all three looks very similar. We can use one variable for our analysis further.



- Loan_amnt variable will be considered.

Univariate Analysis

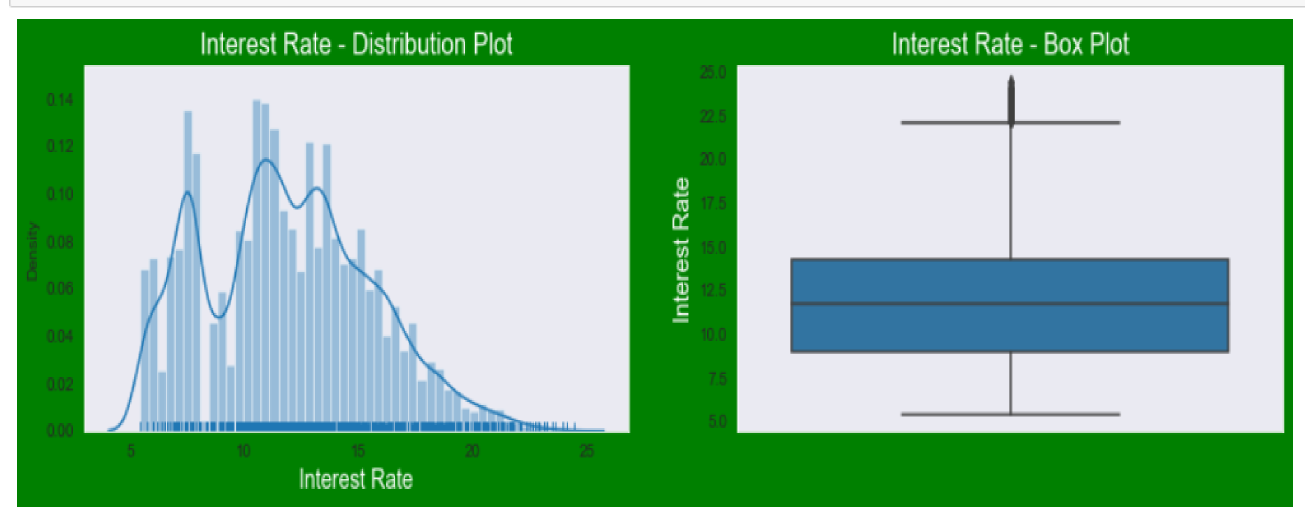
- Loan Amount distribution



- Observation: Most of the loan amounts are in the range of 5000 - 15000

Univariate Analysis

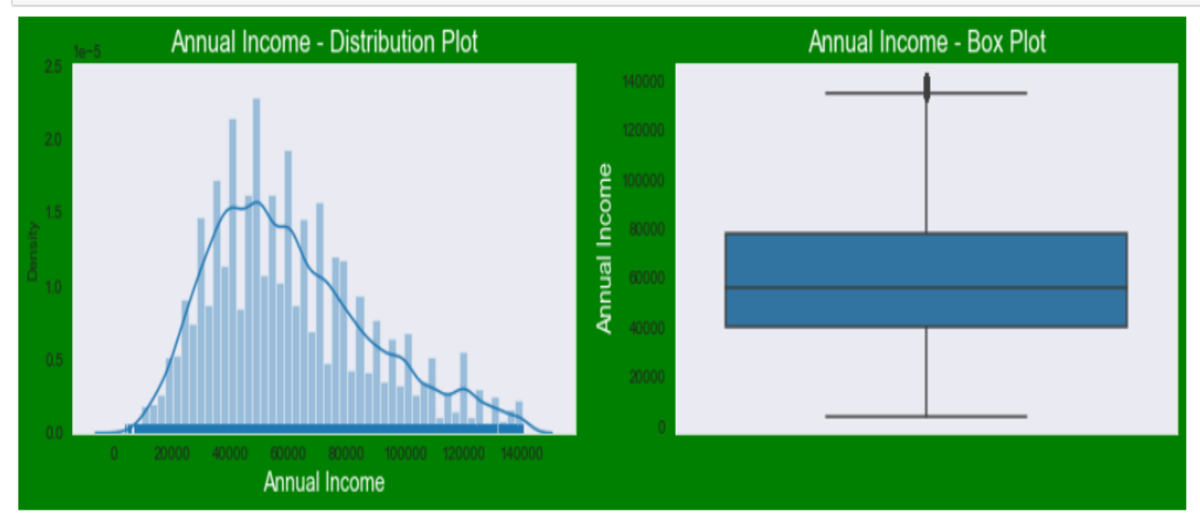
2. Distribution plot of interest rates.



- Observation: Most interest rates are in the range of 10 – 15%

Univariate Analysis

3. Distribution plot of annual income.

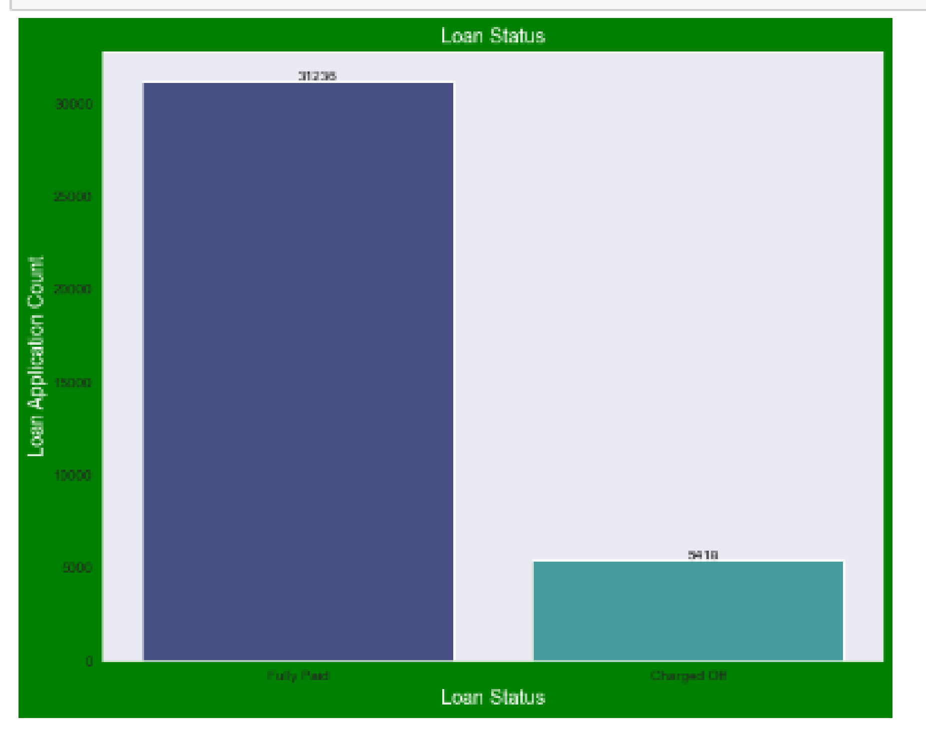


- Observation: Annual income of the borrowers are mostly in the range of 40000c-80000

Conclusion
Analysis

Univariate Analysis

4. Count plot of loan status.

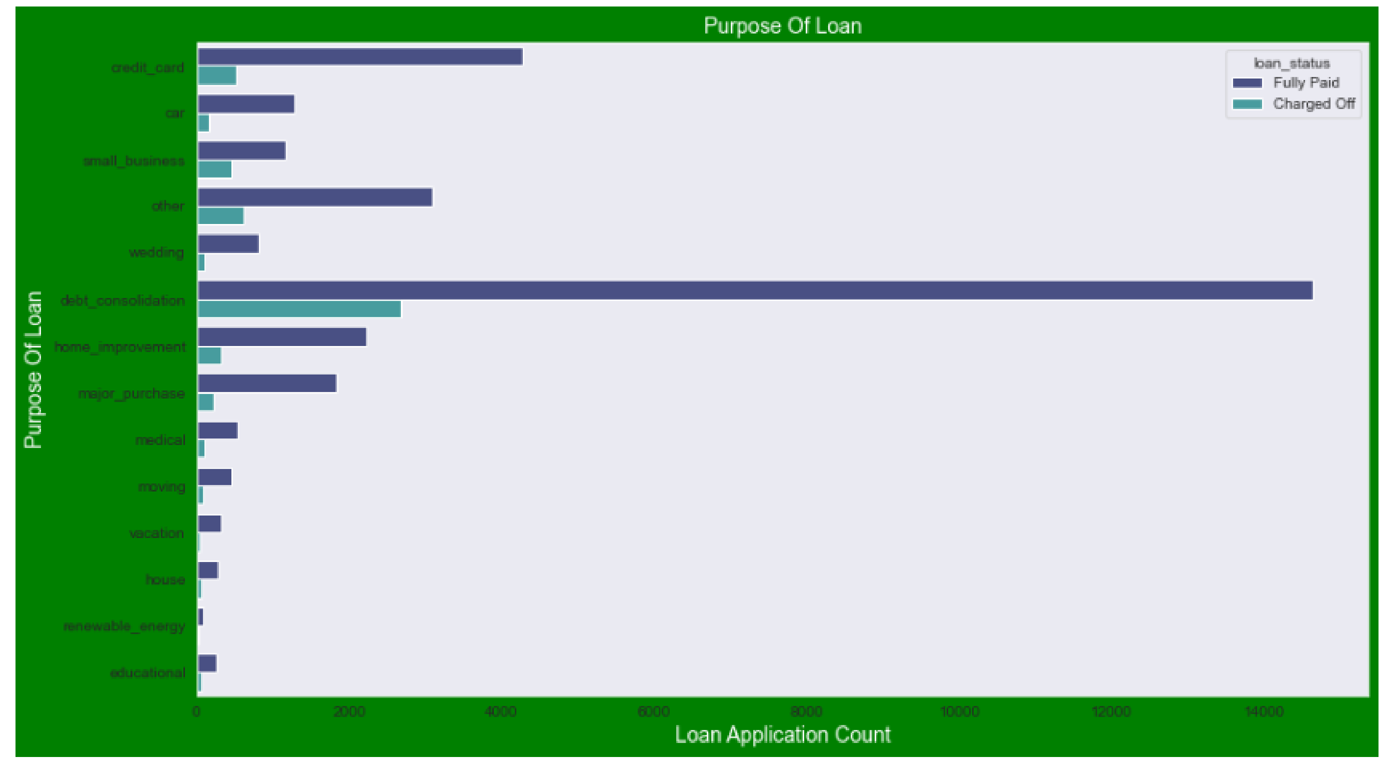


- Observation: Close to 15% of loans were charged off out of total loans issued.

Analysis
Data
Problem
Team

Univariate Analysis

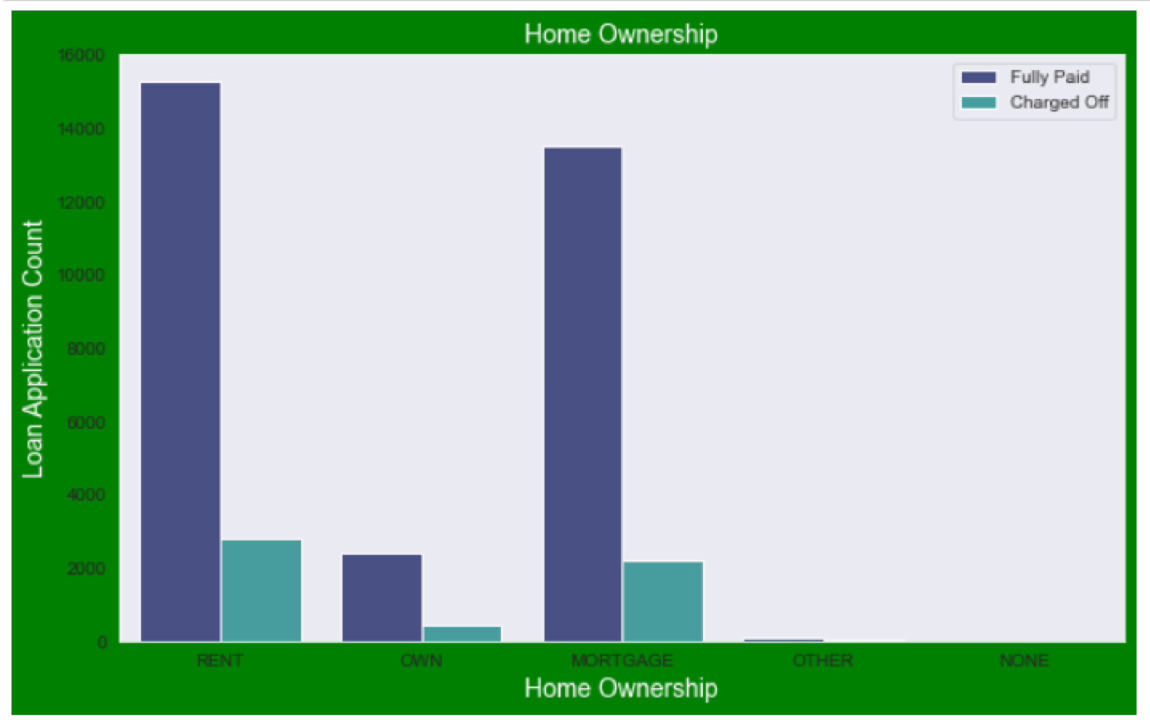
5. Count plot of loan purpose.



- Observation: Most of the loans were taken for debt consolidation and credit bill payments. Also, the charged-off loans are high for these types of loans.

Univariate Analysis

5. Count plot of home ownership.



- Observation: Most of the loans were taken from people who are living on rent or those who have mortgaged their home.

Univariate Analysis

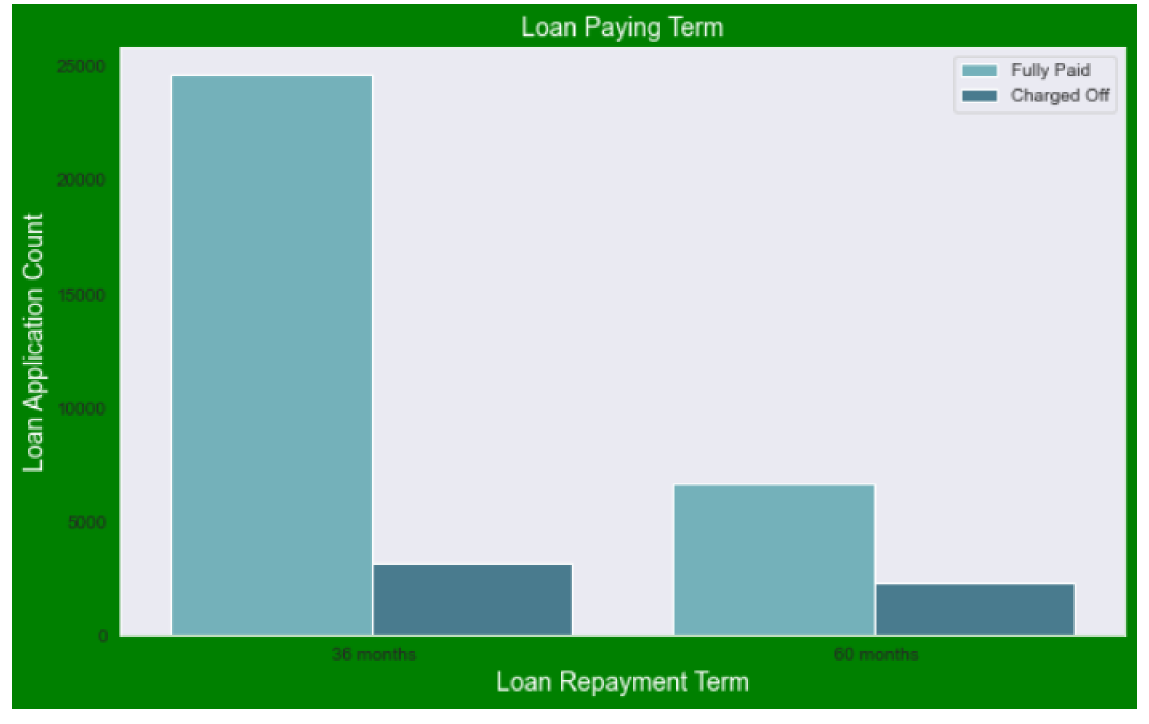
6. Bar plot of year and month.

- Observations: Number of loan applications are increasing year on year.
- Increasing in number of loans add to number of charged loans.
- Number of loans reduced in 2008 (may be due to great recession)

Conclusion
Analysis
Analysis
Data
Problem
Team

Univariate Analysis

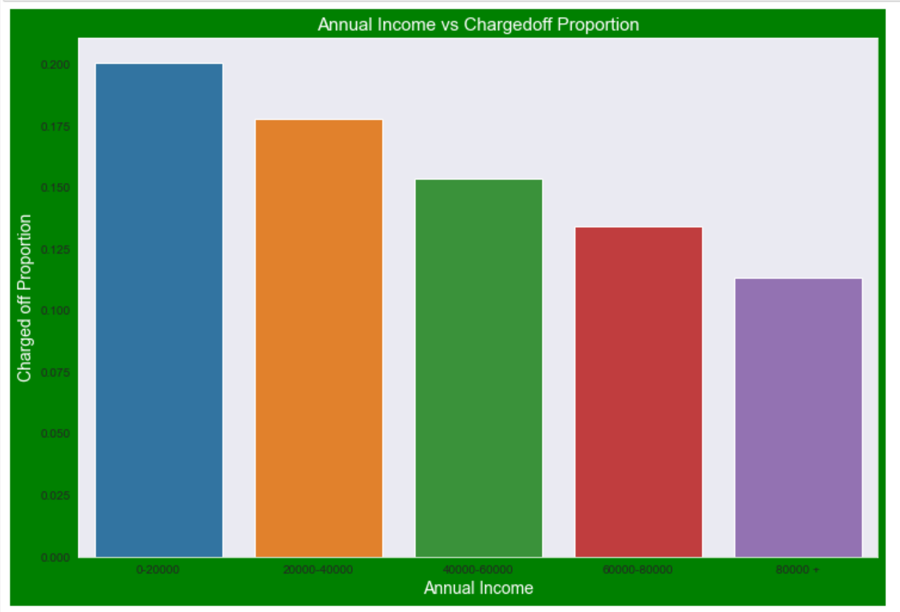
7. Count plot of loan paying term.



- Observations: Applicants getting charged off are more in number who have taken 60 month tenure than 36.

Bivariate Analysis

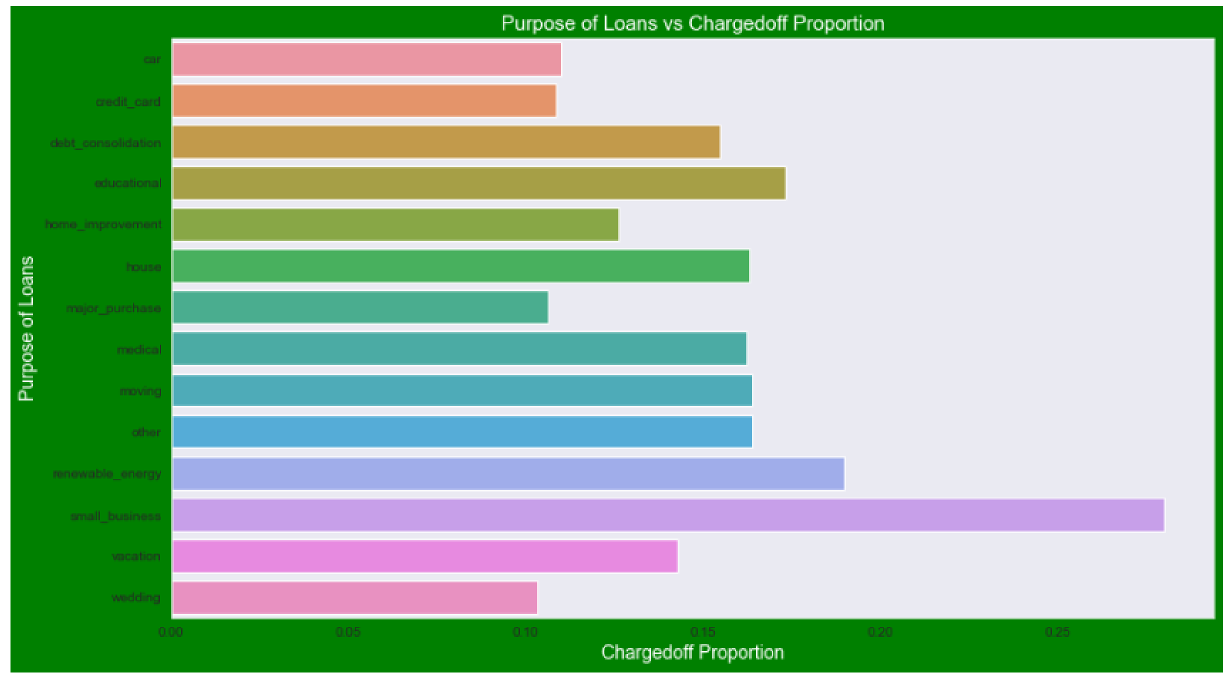
1. Annual income vs charged off proportion



- Observations: Income range of 80000+ are less likely to default.
- Income range of 0-20000 are more likely to default.
- Increase in annual income results in low charged off proportion. High income applicants are less likely to default.

Bivariate Analysis

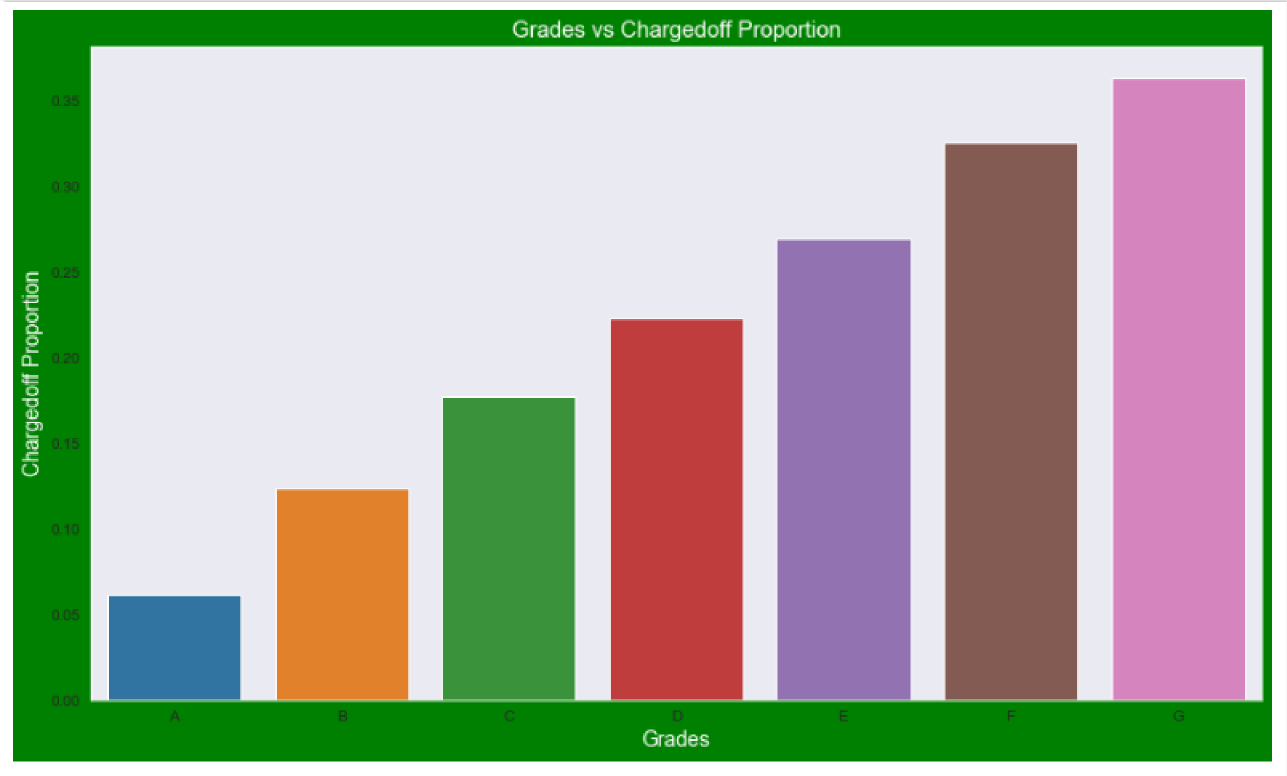
2. Purpose of loan vs charged off proportion



- Observation: Small business applicants are more likely to default.

Bivariate Analysis

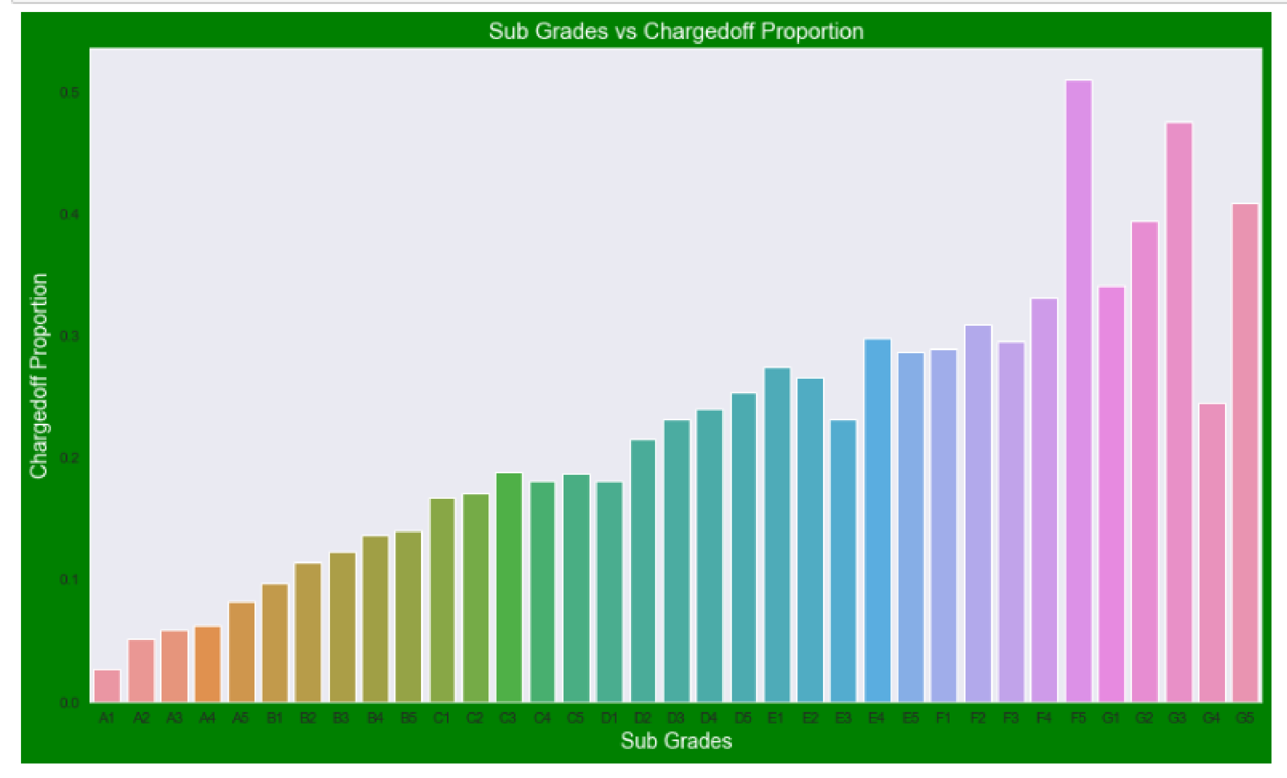
3. Grade vs charged off proportion



- Observation: Grade ‘A’ are very low on defaulter proportion
- Grade ‘F’ and ‘G’ are high on defaulter proportion
- Defaulter proportion is increasing with increasing grades.

Bivariate Analysis

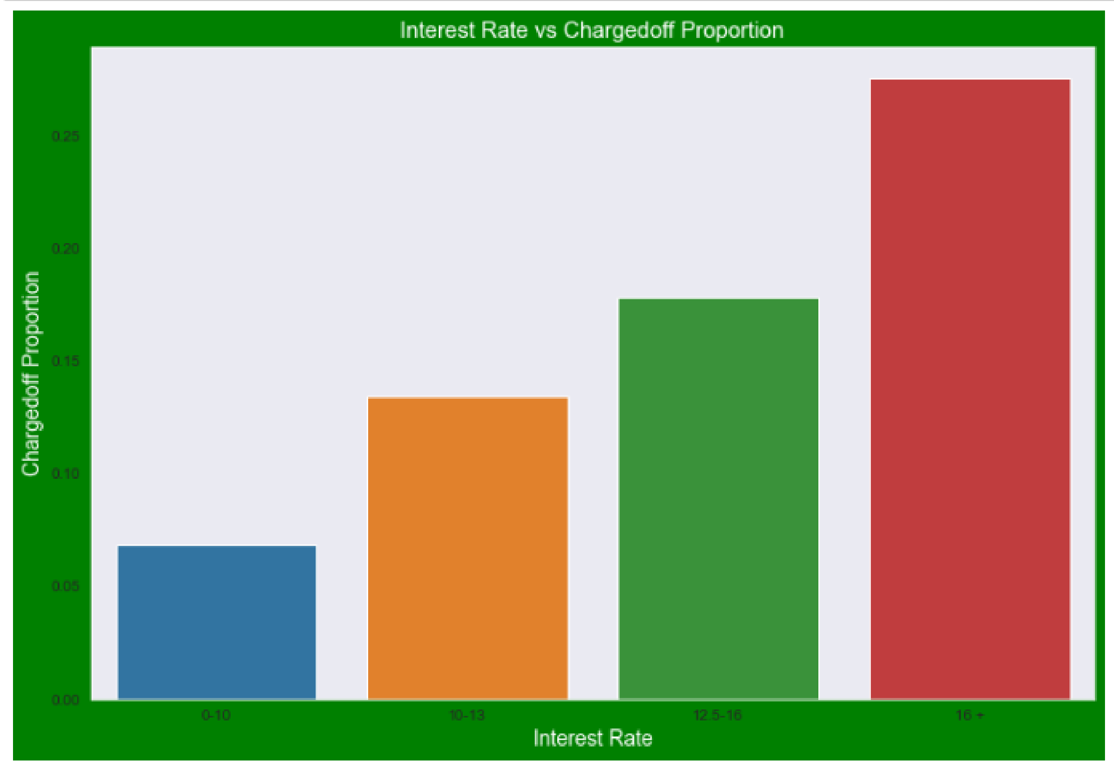
4. Sub-Grade vs charged off proportion



- Observation: Sub Grades of "A" are very less likely to charged off.
- Sub Grades of "F" and "G" are more likely to charged off.
- Defaulter proportion is increasing with sub grades moving from sub grades of "A" towards sub grades of "G"

Bivariate Analysis

5. Interest rate vs charged off proportion



- Observation: Interest rates of more than 16% are more likely to default
- Interest rates less than 10% are less likely to default
- The defaulter proportion is increasing with higher interest rates

Conclusion

Analysis

Analysis

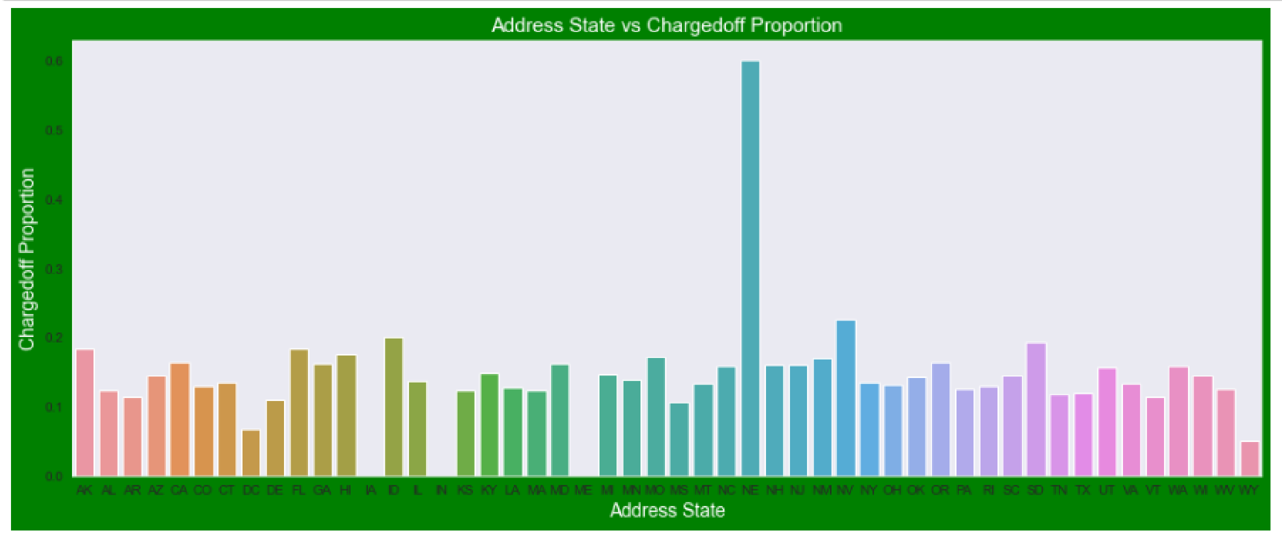
Data

Problem

Team

Bivariate Analysis

6. Address state vs charged off proportion



- Observation: State 'NE' is showing high chances of default

Conclusion

Analysis

Analysis

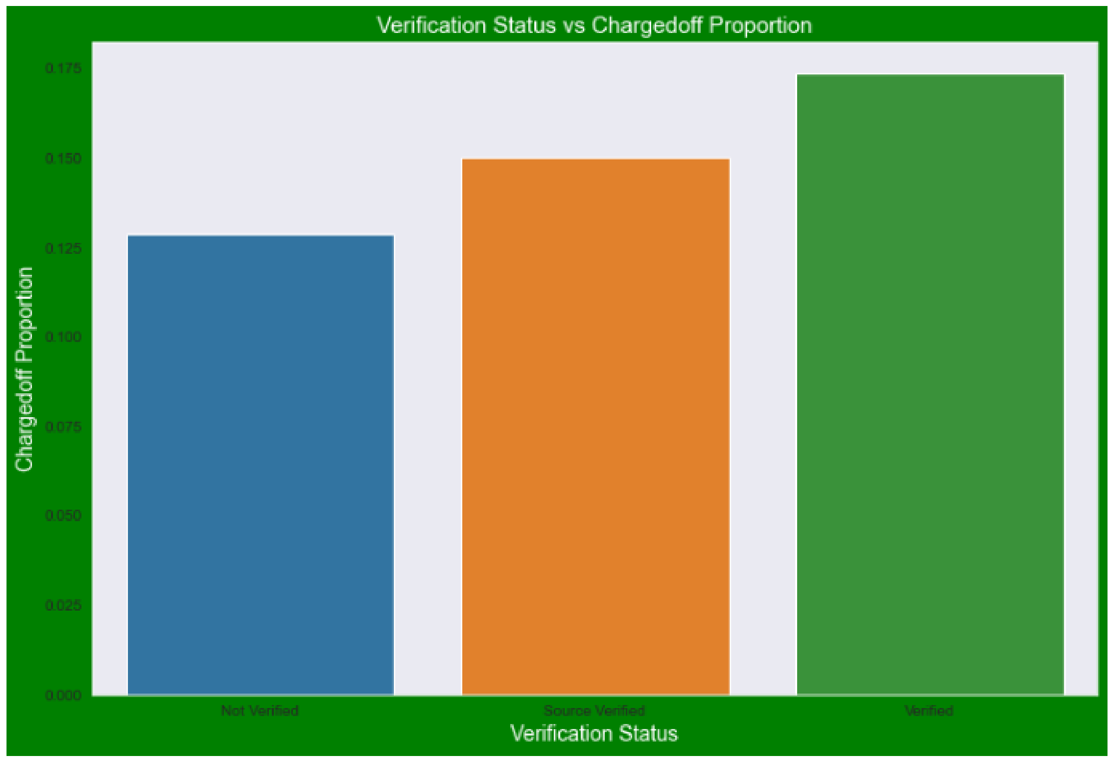
Data

Problem

Team

Bivariate Analysis

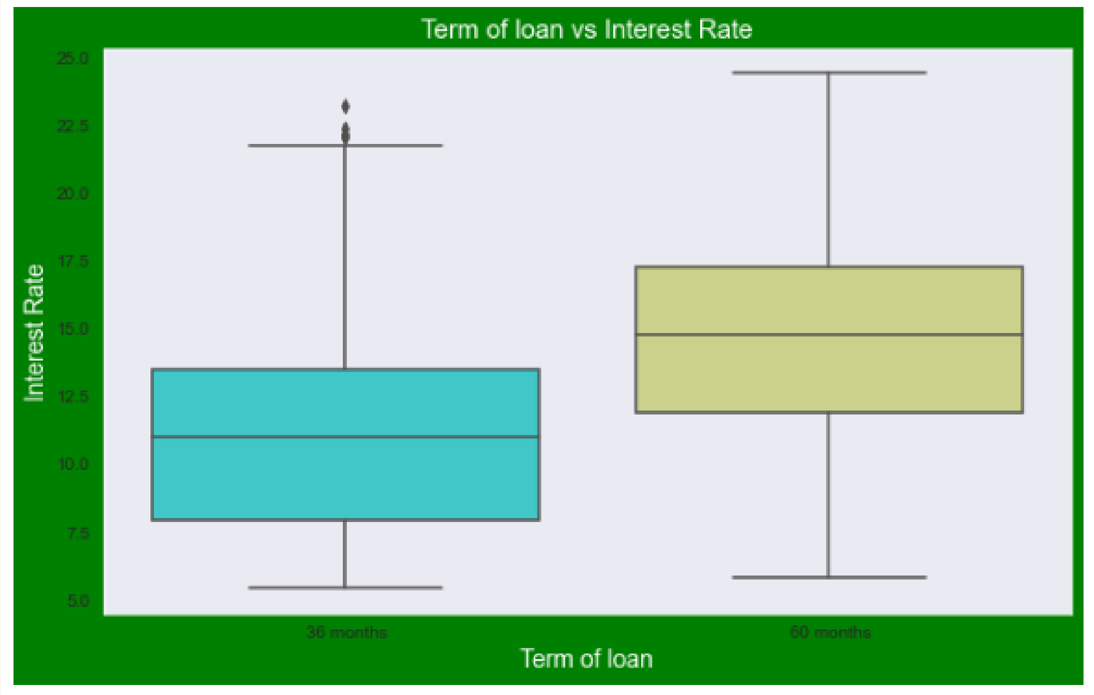
7. Verification status vs charged off proportion



- Observation: The difference in charged off proportion is very less
- This variable doesn't provide a significant difference in making a decision

Bivariate Analysis

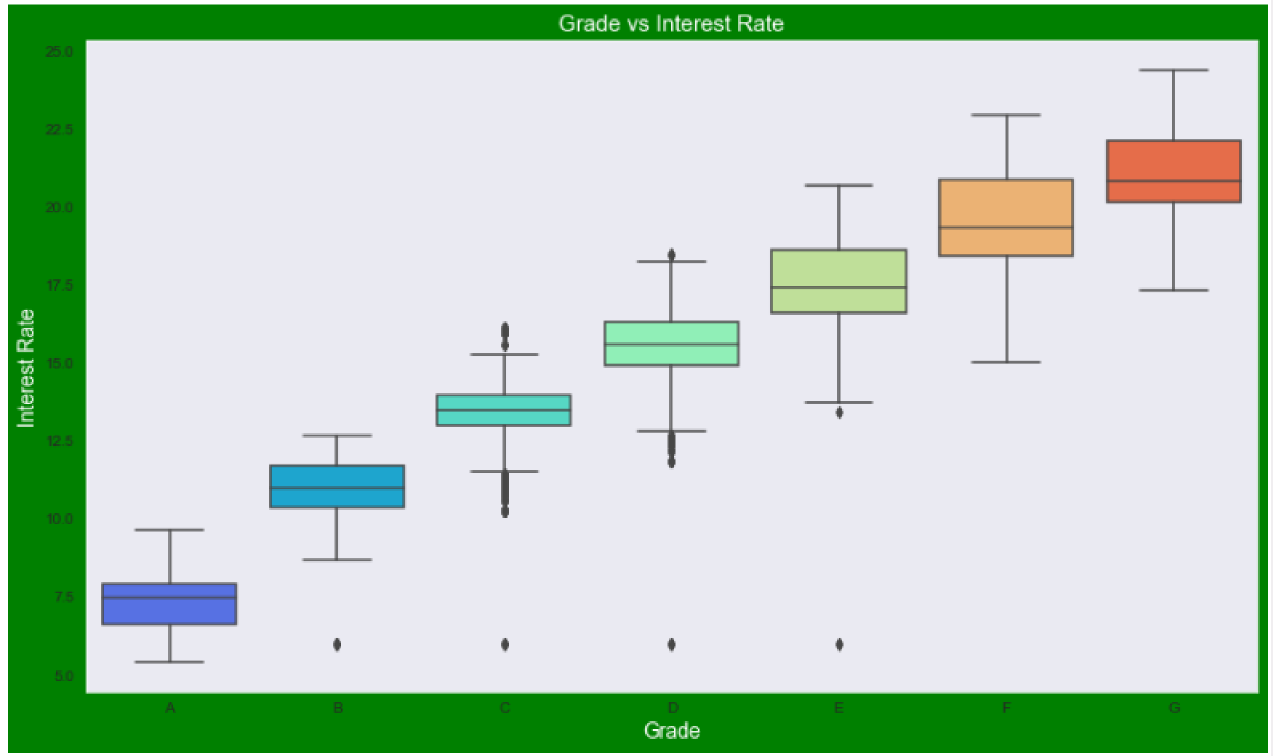
8. Verification status vs Interest rate



- Observation: Average interest rate is higher for 60 months
- Most of the loans issued for the longer term had higher interest rates for repayment.

Bivariate Analysis

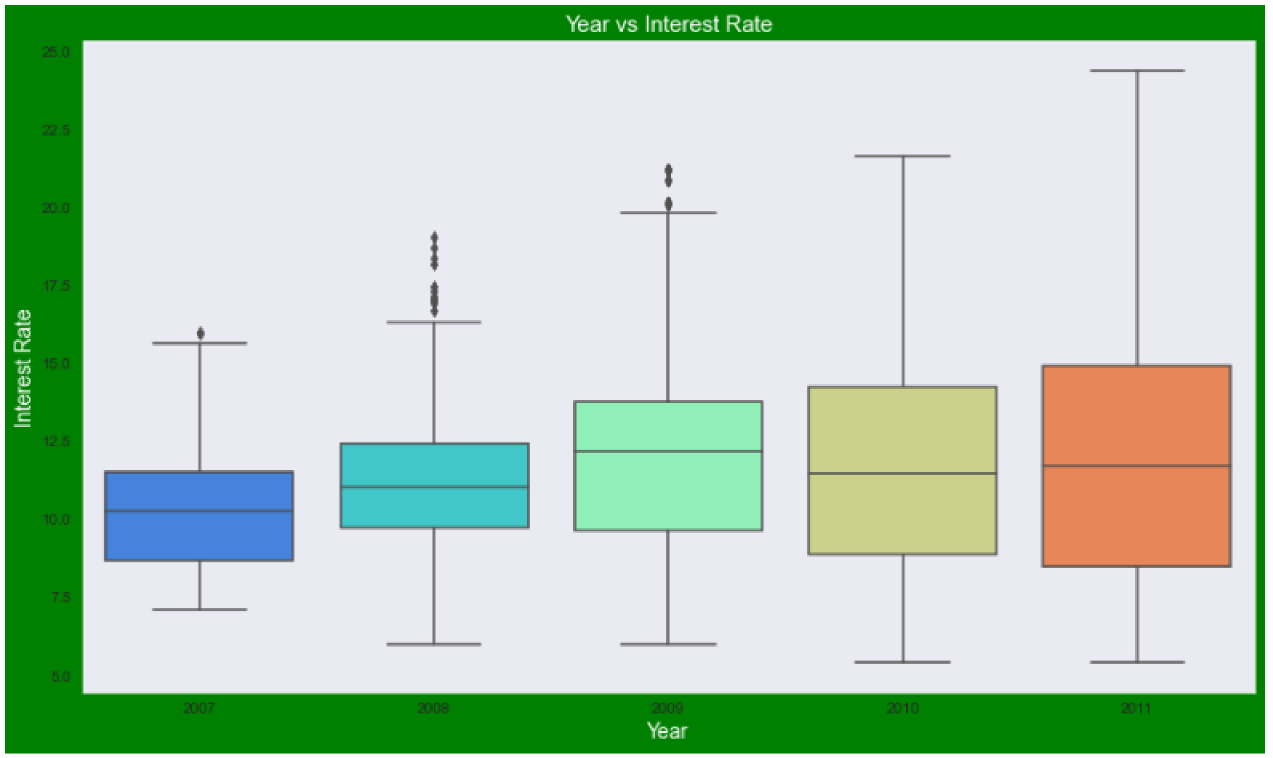
9. Grade vs Interest rate



- Observation: Higher the grade lower the interest rate
- Interest rate is increasing with grades

Bivariate Analysis

10. Year vs Interest rate



Observation: Interest rate is increasing slowly with increase in year.

1. Giving loans to applicants with low income are likely to default.
2. Applicants who are living on rent or mortgaged the home are likely to default
3. Number of loans given should be balanced over the years
4. Small business applicants are likely to default
5. Grade ‘F’ and ‘G’ are high on defaulter proportion
6. Interest rates more than 16% are more likely to default
7. Applicants who are not working or having less than one year of experience are likely to default.
8. State ‘NE’ applicants are likely to deault.