Lending Club Case study

Agenda

- Problem Statement
- Objective
- Python Modules used
- Univariate Analysis
- Bivariate Analysis
- Conclusion

Problem Statement

- We work for a finance company whose expertise is to lend different types of loans to urban customers
- When a loan application is received, a decision needs to be taken based on applicant's profile
  - Denying a loan to an applicant who can repay will result in revenue loss
  - Approving a loan to a defaulter can result in financial loss

Objective

- The objective is to find patterns which indicates the person is likely to default by
  - Analysing data of past loan applicant’s
  - Cleaning the data
  - Dropping unnecessary/redundant columns
  - Performing univariate analysis
  - Performing bivariate analysis
  - Find out the key features that are indicators of becoming ‘Default’.

Python modules used

- Numpy
- Pandas
- Seaborn
- Matplotlib.pyplot
- Plotly.express
- datetime

Data exploration

- Understanding the following
  - Size and shape of the data – There are 39717 rows and 111 columns
  - Data types
  - Check null values
  - Check the percentage of unique values for each column
  - Understand the median and spread of the data.

Data cleansing

- Drop columns which
  - have all null values
  - Have unique values like id, member_id
  - Has a single value like ‘initial_status_list’, ‘pymnt_plan’
  - Has description like ‘title’, ‘desc’
  - do not affect loan_status ‘Default’ like ‘out_prncp’, ‘next_pymt_d’
- After dropping columns as mentioned above, the column list reduces to 39 from 111.
- Fill na with default values

Datatypes

- Converted Object to corresponding data types

Convert data types

- Object to Date
- String to Float

```
loan['issue_d'] = loan.issue_d.apply(lambda d: datetime.datetime.strptime(d,'%b-%y'))
loan['issue_d'].head()
```

```
0    2011-12-01
1    2011-12-01
2    2011-12-01
3    2011-12-01
4    2011-12-01
Name: issue_d, dtype: datetime64[ns]
```

```
loan['earliest_cr_line_d'] = loan.earliest_cr_line.apply(lambda d: datetime.datetime.strptime(d,'%b-%y'))
```

```
#Fill nan with Jan 99
loan.last_credit_pull_d=loan.last_credit_pull_d.fillna('Jan-99')

# Convert last_credit pull date to date
loan['last_credit_pull_d']=loan.last_credit_pull_d.apply(lambda d: datetime.datetime.strptime((str(d)[:6]),'%b-%y') )
```

```
#Fill nan with Jan 99
loan['last_pymnt_d']=loan['last_pymnt_d'].fillna('Jan-99')

# Convert last payment date to date
loan['last_pymnt_d']=loan.last_pymnt_d.apply(lambda d: datetime.datetime.strptime((str(d)[:6]),'%b-%y') )
```

```
#Convert revol_util and int rate to number
loan.revol_util=loan.revol_util.apply(lambda x: float(str(x).replace('%', '')))
loan.int_rate=loan.int_rate.apply(lambda x: float(x.replace('%','')))
```

Derived columns

- Created few derived columns extracting month and year for date columns.

Create derived columns

4]:
```
#Create year and month columns for analysis
loan['earliest_cr_line_year']=pd.DatetimeIndex(loan['earliest_cr_line_d']).year
loan['earliest_cr_line_month']=pd.DatetimeIndex(loan['earliest_cr_line_d']).month

loan['last_credit_pull_year']=pd.DatetimeIndex(loan['last_credit_pull_d']).year
loan['last_credit_pull_month']=pd.DatetimeIndex(loan['last_credit_pull_d']).month



loan['issue_year']=pd.DatetimeIndex(loan['issue_d']).year
loan['issue_month']=pd.DatetimeIndex(loan['issue_d']).month


loan['last_pymt_year']=pd.DatetimeIndex(loan['last_pymnt_d']).year
loan['last_pymt_month']=pd.DatetimeIndex(loan['last_pymnt_d']).month
```

8]:
```
loan['diff_in_days_credit_and_last_pymt']= ((loan['last_credit_pull_d'] -loan['last_pymnt_d']))
```

3]:
```
#create default column and set to True if Charged Off
loan['default'] = loan.loan_status.apply(lambda x: True if x == 'Charged Off' else False)
```

Univariate Analysis

Issue Date

Most of the loans are taken in last quarter of the year

```
#Most of the loans got issued in last 2 quarters of 2011
loan.issue_d.value_counts(normalize=True).sort_values(ascending=False)
```

```
2011-12-01    0.056903
2011-11-01    0.055971
2011-10-01    0.053227
2011-09-01    0.051942
2011-08-01    0.048543
2011-07-01    0.047083
2011-06-01    0.046000
2011-05-01    0.042526
2011-04-01    0.039328
2011-03-01    0.036332
2011-01-01    0.034746
2011-02-01    0.032656
2010-12-01    0.031901
2010-10-01    0.028502
2010-11-01    0.028225
2010-07-01    0.028174
2010-09-01    0.027343
2010-08-01    0.027142
2010-06-01    0.025908
```

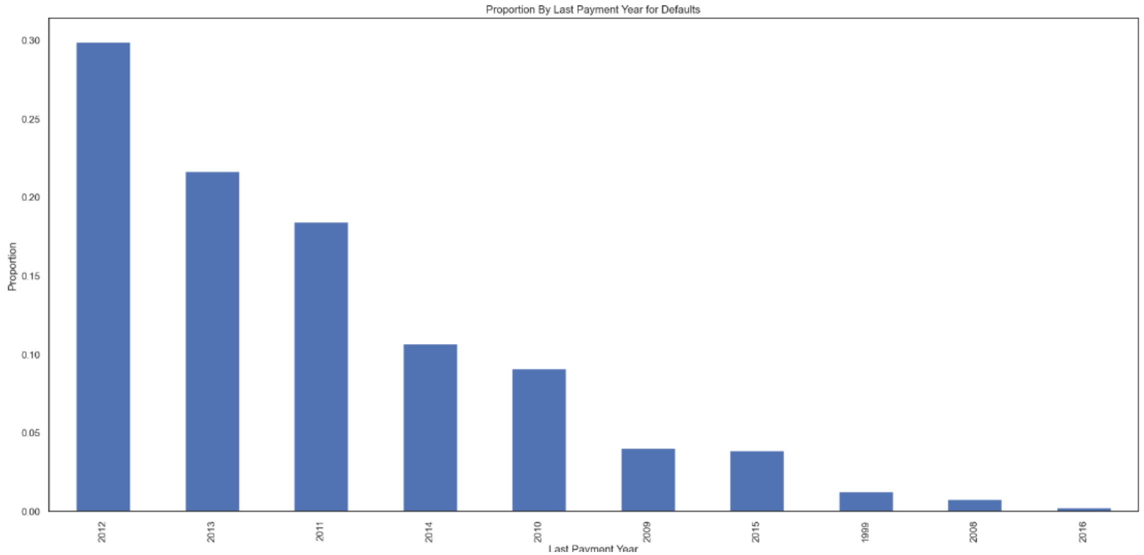
```
# 50% of the loans got issued in 2011
loan.issue_year.value_counts(normalize=True)
```

```
2011    0.545258
2010    0.290354
2009    0.118740
2008    0.039328
2007    0.006320
Name: issue_year, dtype: float64
```

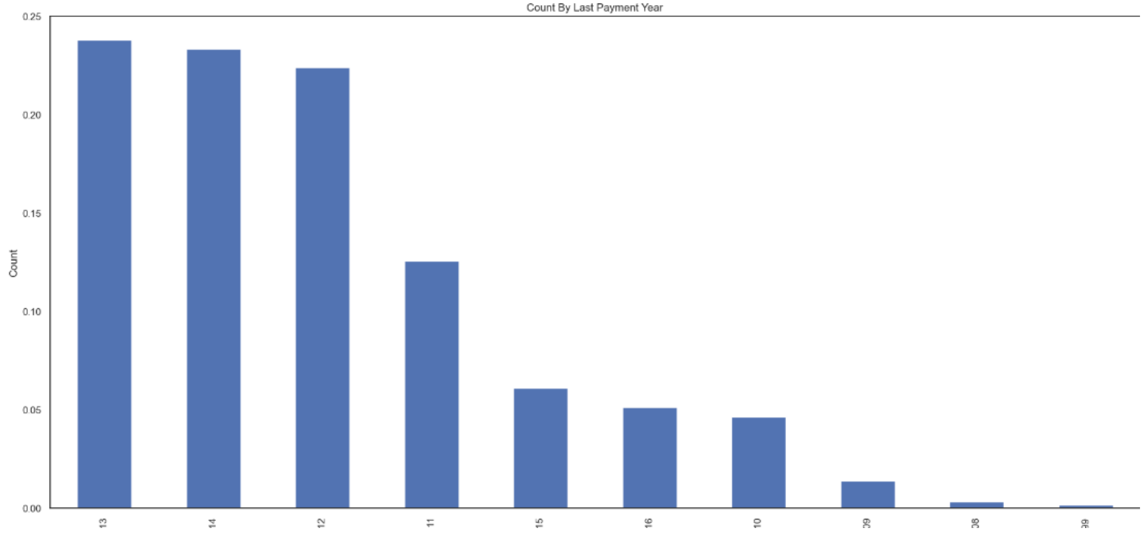
Last Payment Date

- Most of them have last payment date in between 2013 and 2015
- Most of the defaulters have last payment date in 2012

```
# Most Last payment dates for defaulters is 2012
plt=value_count_graph('last_pymt_year', 'Proportion By Last Payment Year for Defaults', 'Last Payment Year', 'Proportion', True,
plt.show()
```

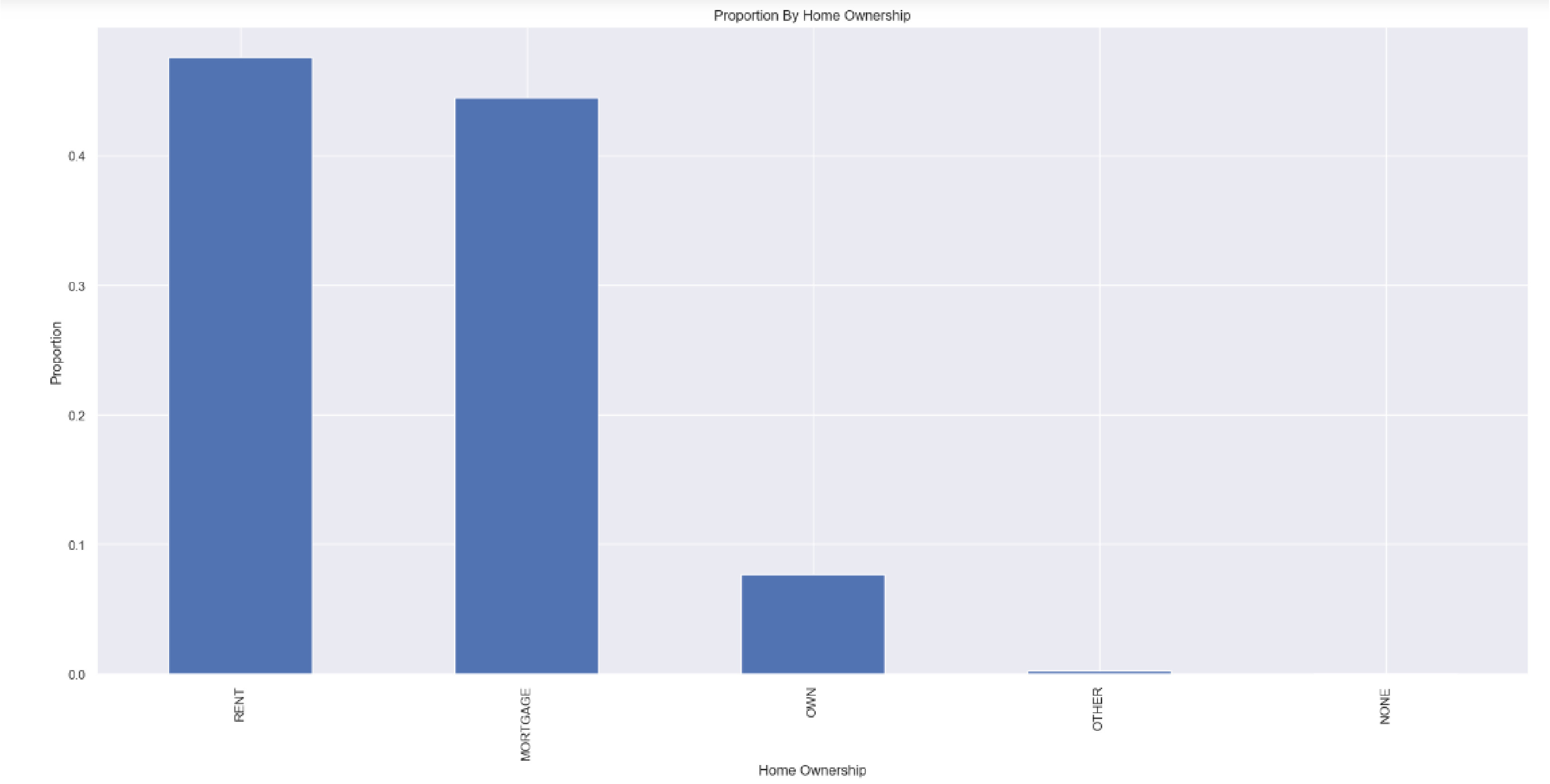



```
# Most Last payment dates is in between 2013 and 2015
plt=value_count_graph('last_pymt_year', 'Count By Last Payment Year', 'Last Payment Year', 'Count', True)
plt.show()
```



Home ownership

- People living on rent tend to go for more loans



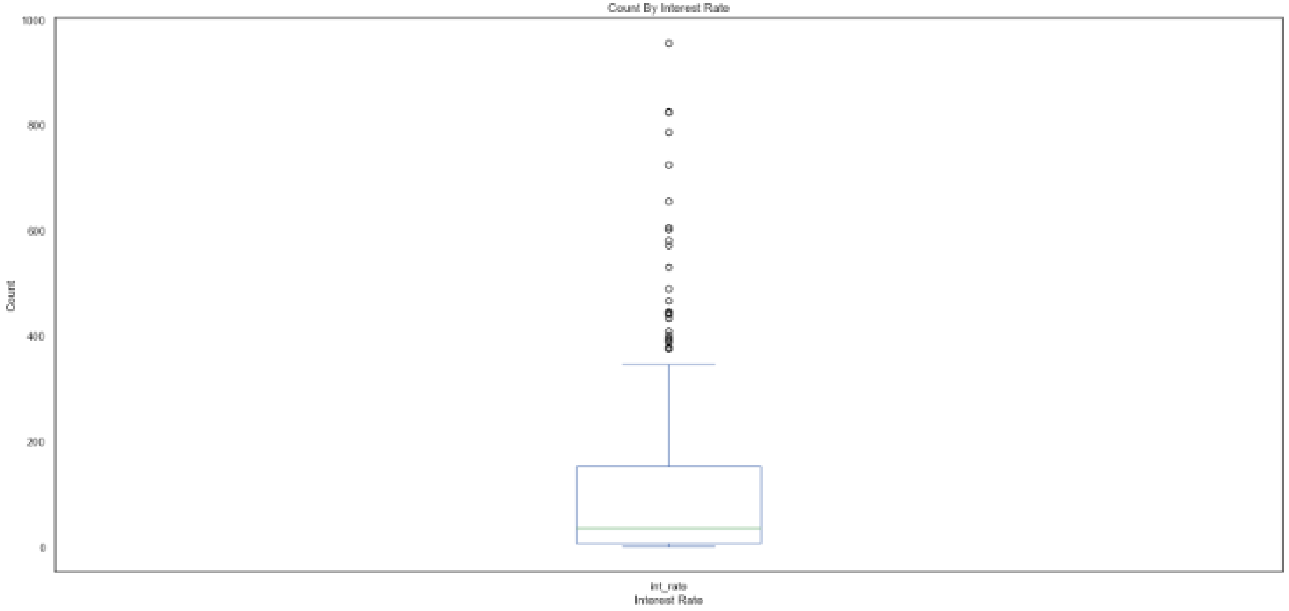
Interest Rate

- 50% of interest rates are between 9.25 and 14.59

```
loan['int_rate'].describe()
```

```
count    39717.000000
mean        12.021177
std          3.724825
min          5.420000
25%          9.250000
50%         11.860000
75%         14.590000
max         24.590000
Name: int_rate, dtype: float64
```

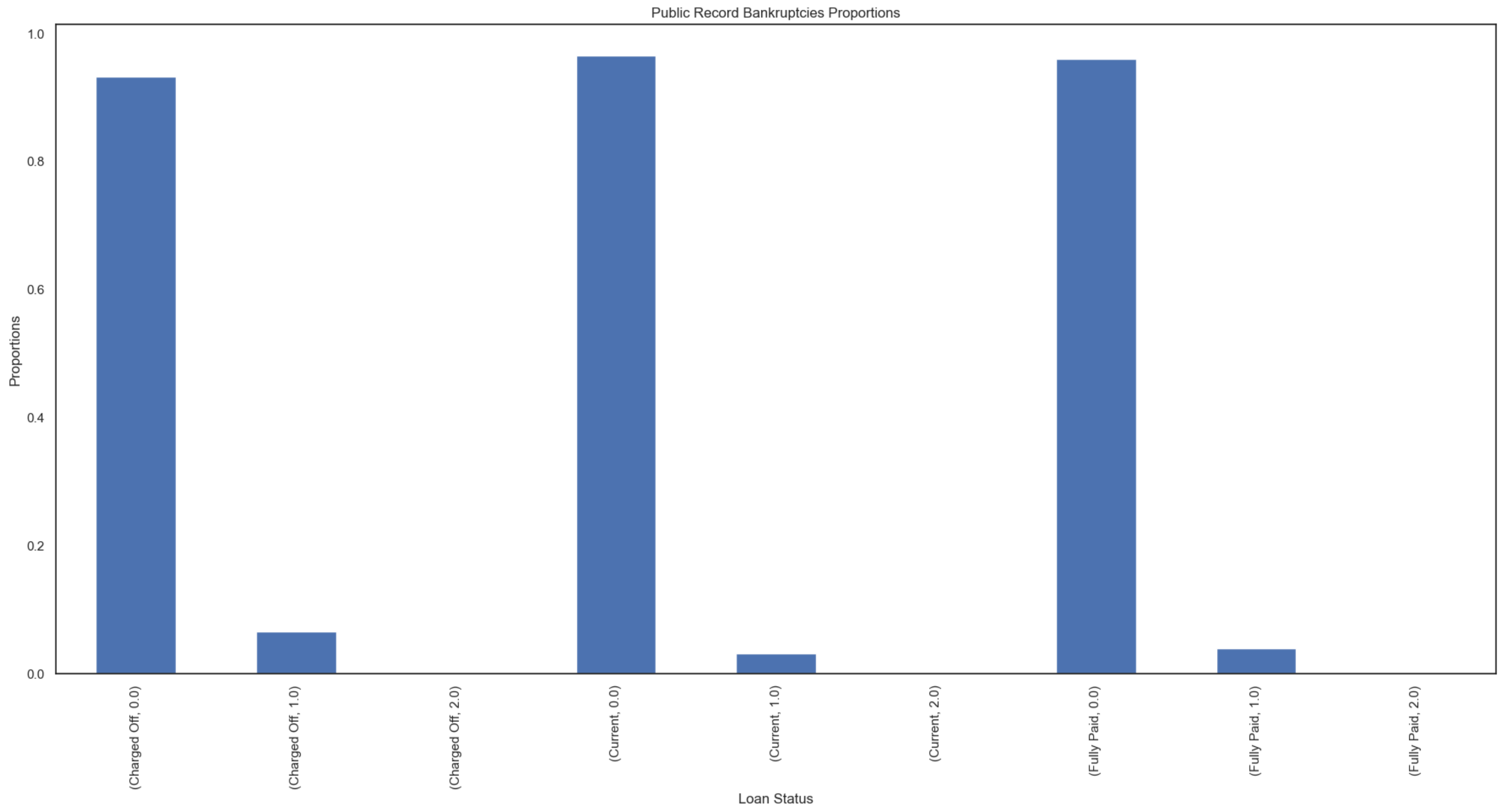
```
plt=value_count_graph('int_rate', 'Count By Interest Rate', 'Interest Rate', 'Count', False, loan,  'box')
plt.show()
```



Bivariate Analysis

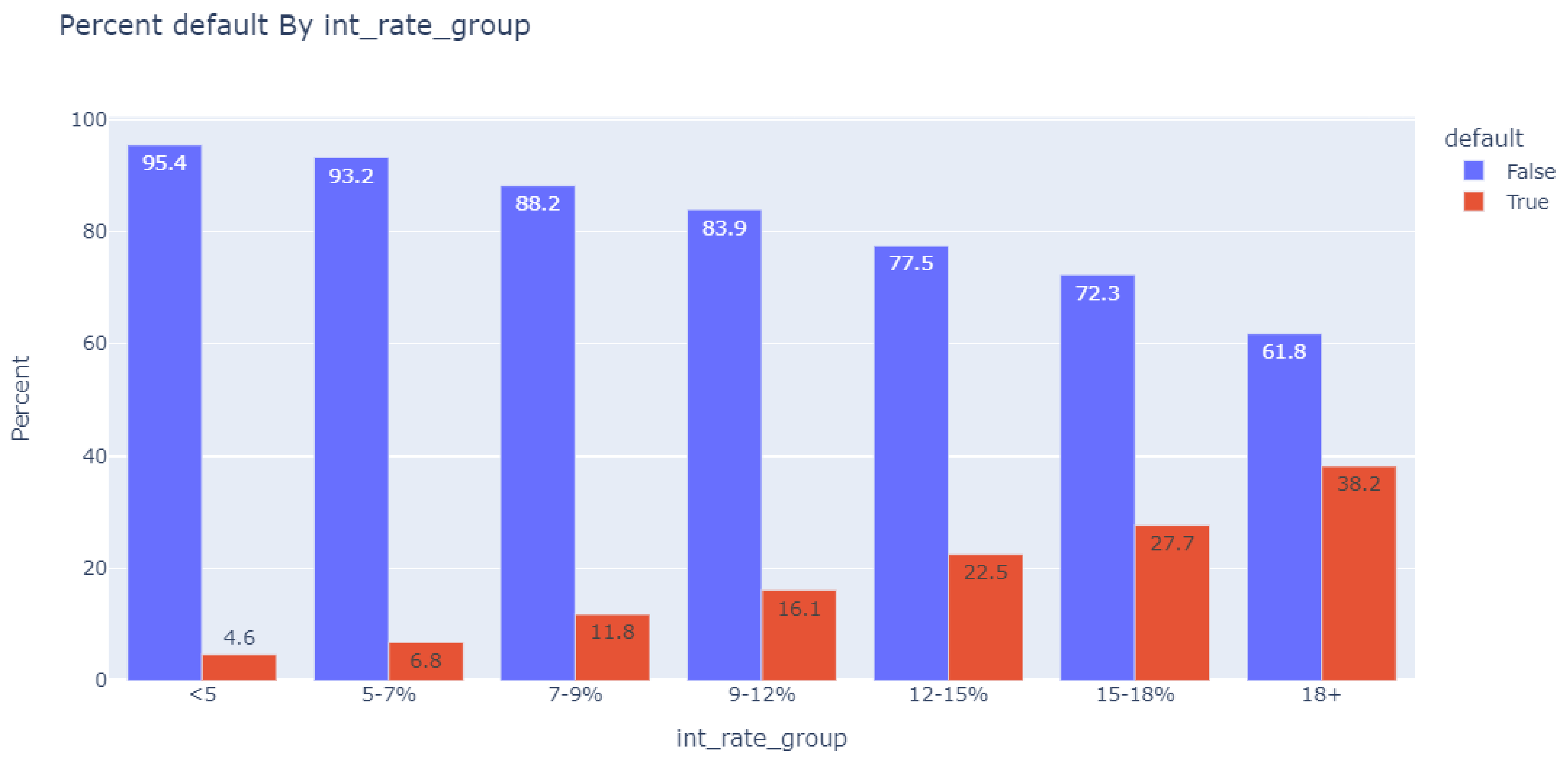
Public Record Bankruptcies

- loan.groupby('loan_status')['pub_rec_bankruptcies'].value_counts(normalize=True).plot.bar()
- The distribution of bankruptcies remain consistent across all loan status. This is not influencing loan Status 'Charged Off'



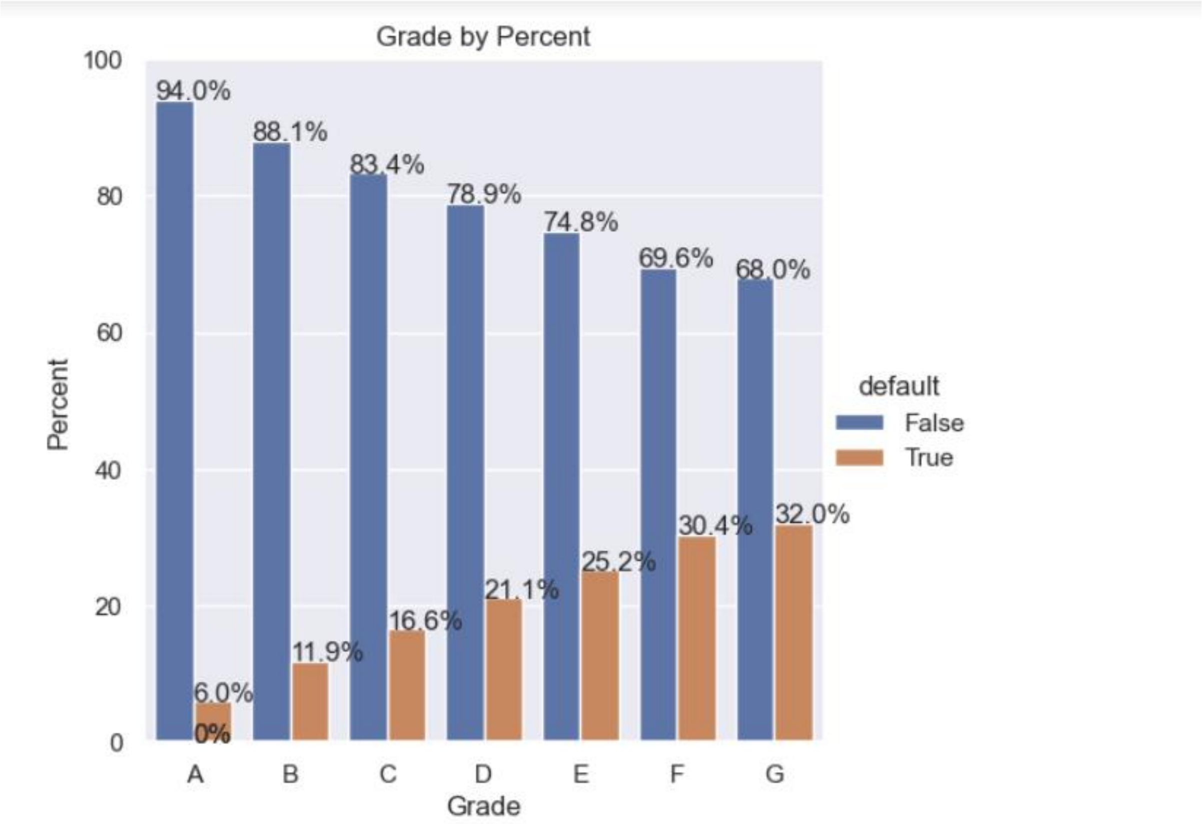
Interest Rate Group

- Interest rates are binned based on the
  [5, 7, 9, 12, 15, 18, 21, 24]
- Applicants with Higher interest rate tend to default more.



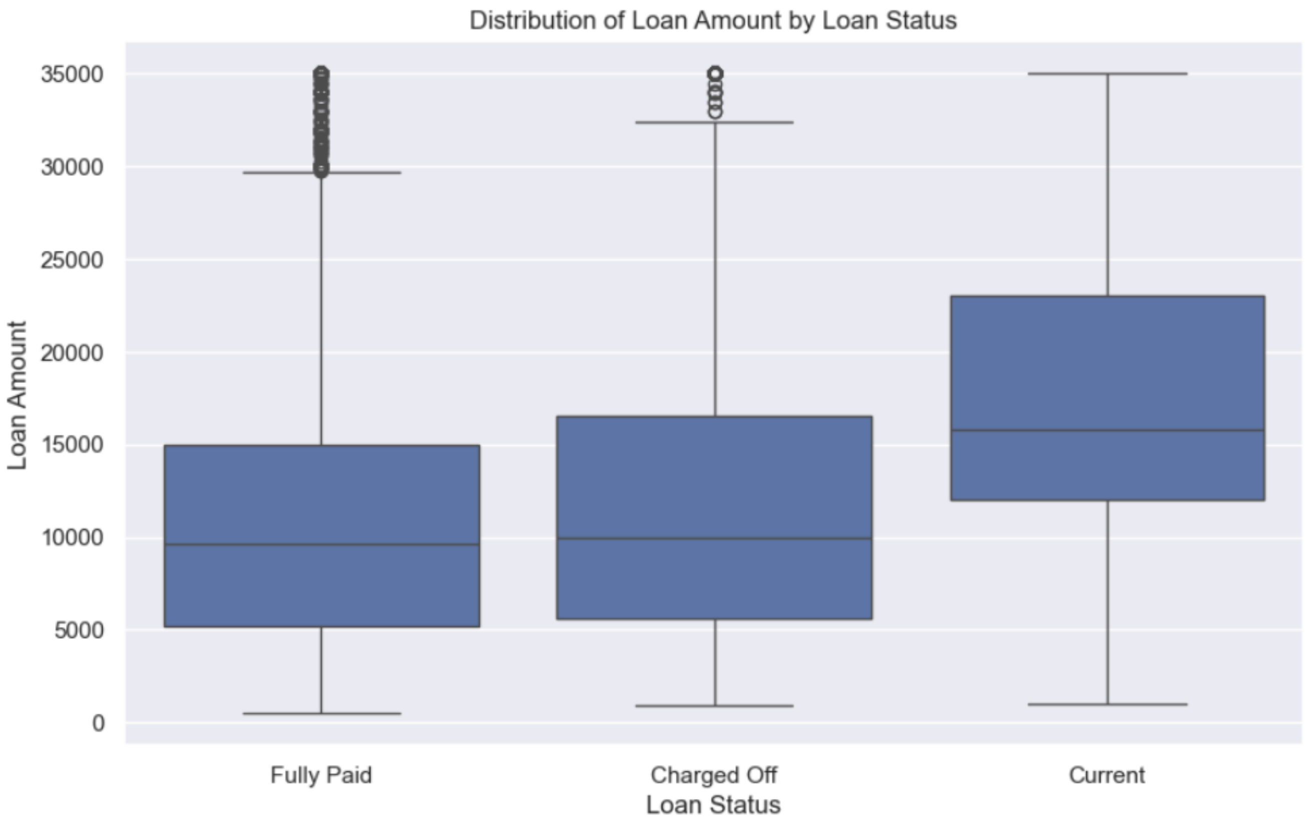
Grade

- Applications with higher grades tend to default more.



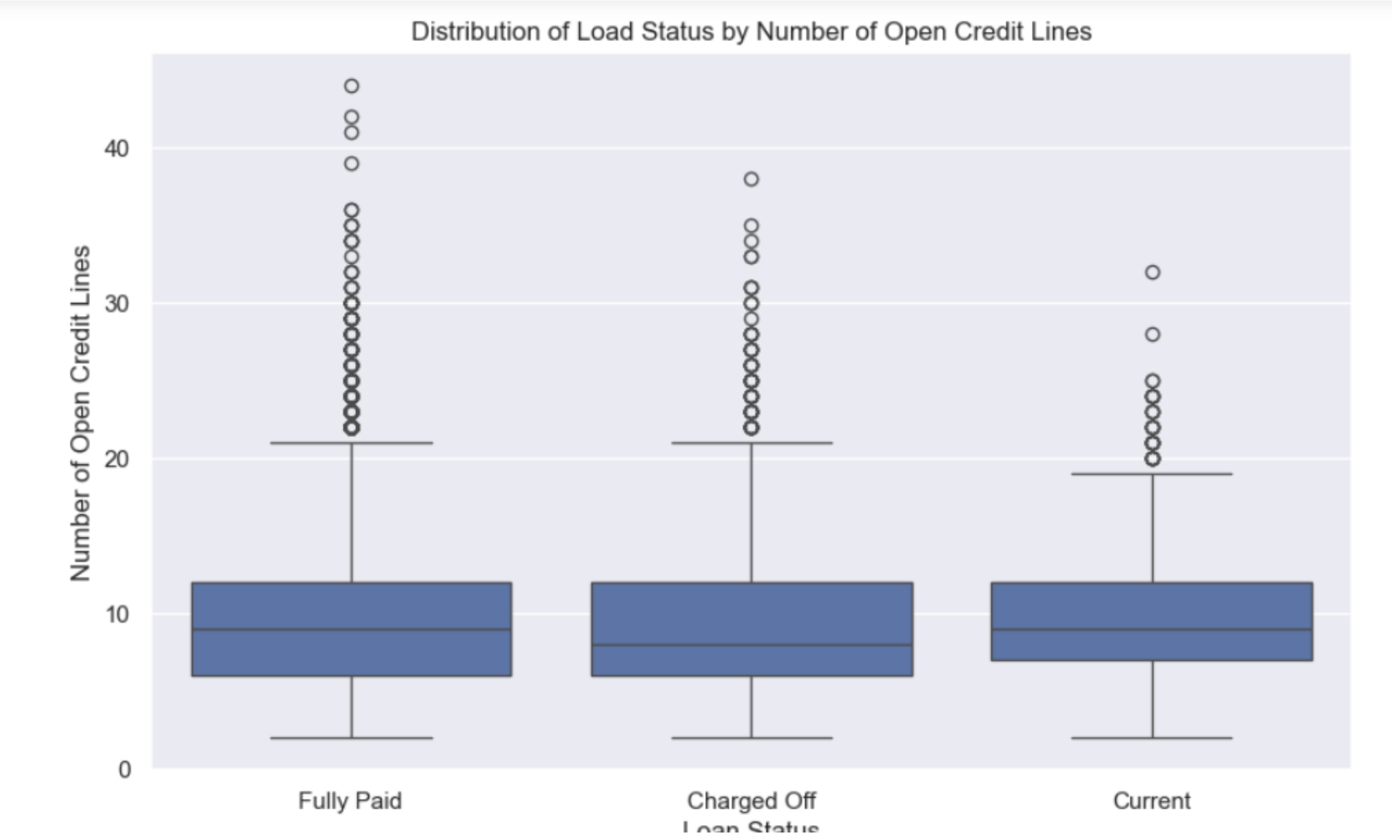
Loan Amount

- Loan amount does not impact the Loan Status



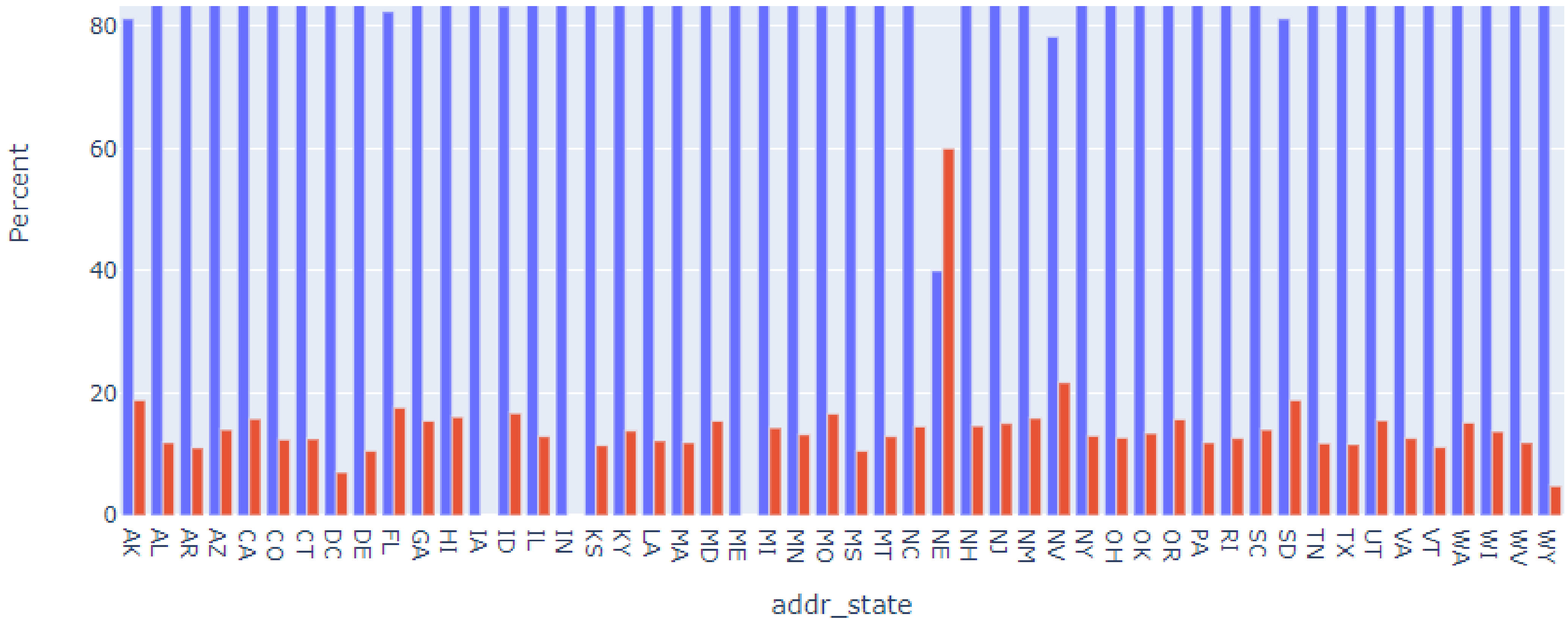
Open credit Lines

- Open Credit Lines doesn't seem to impact Loan status



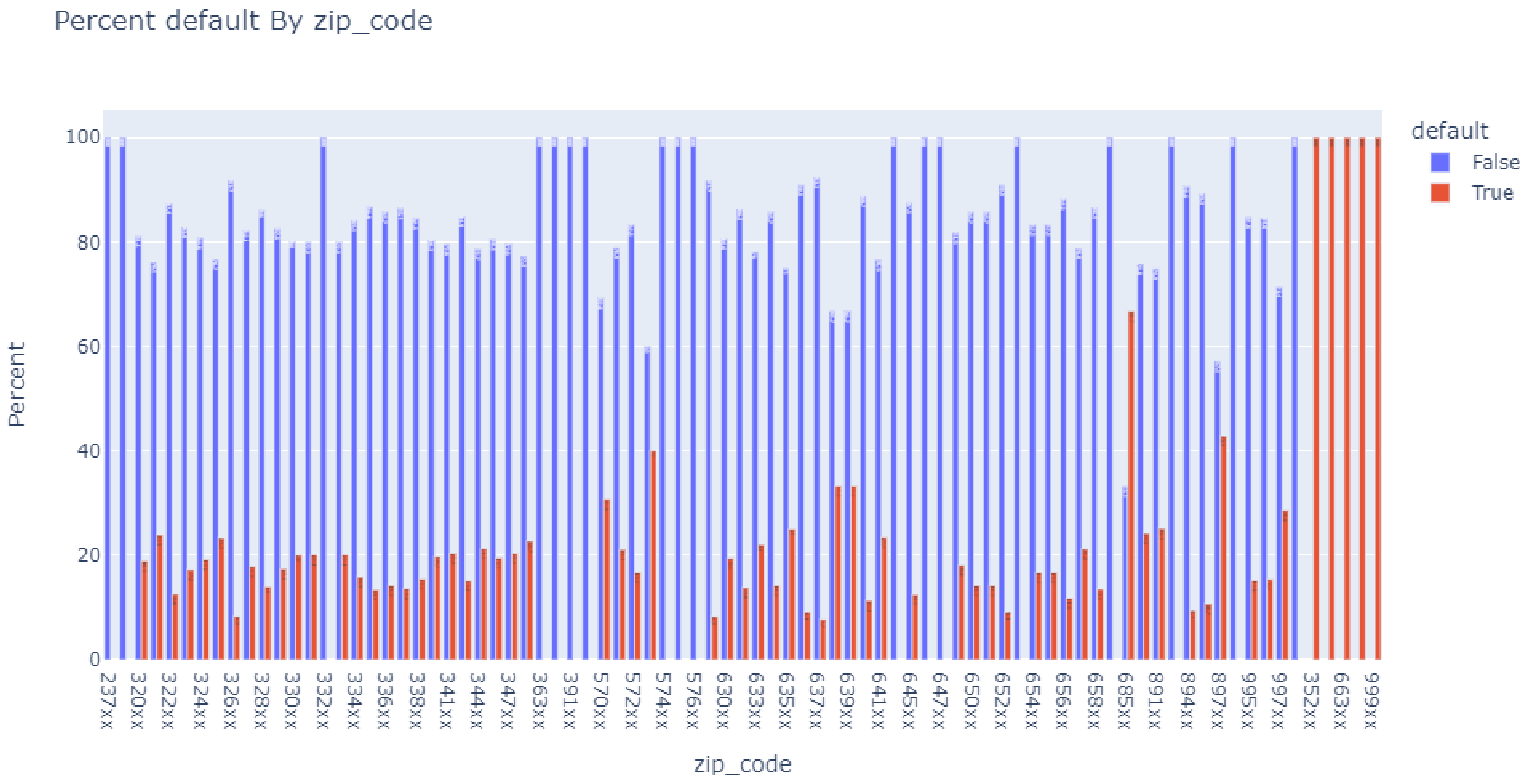
Addr_state

- Applicants from ‘NE’ tend to default more.



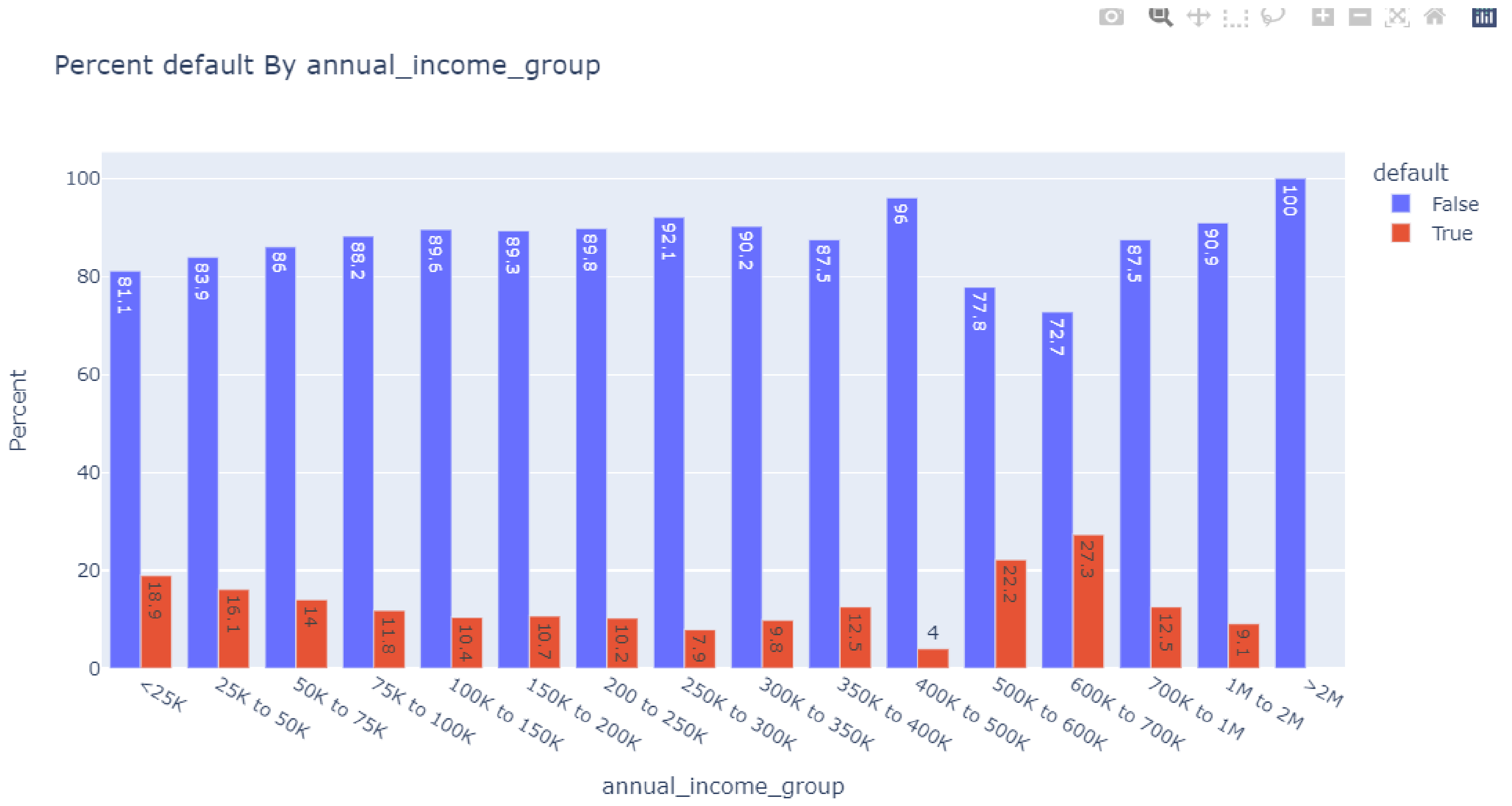
Zipcode

- Zipcode starting with 999,663,352 are more likely to default



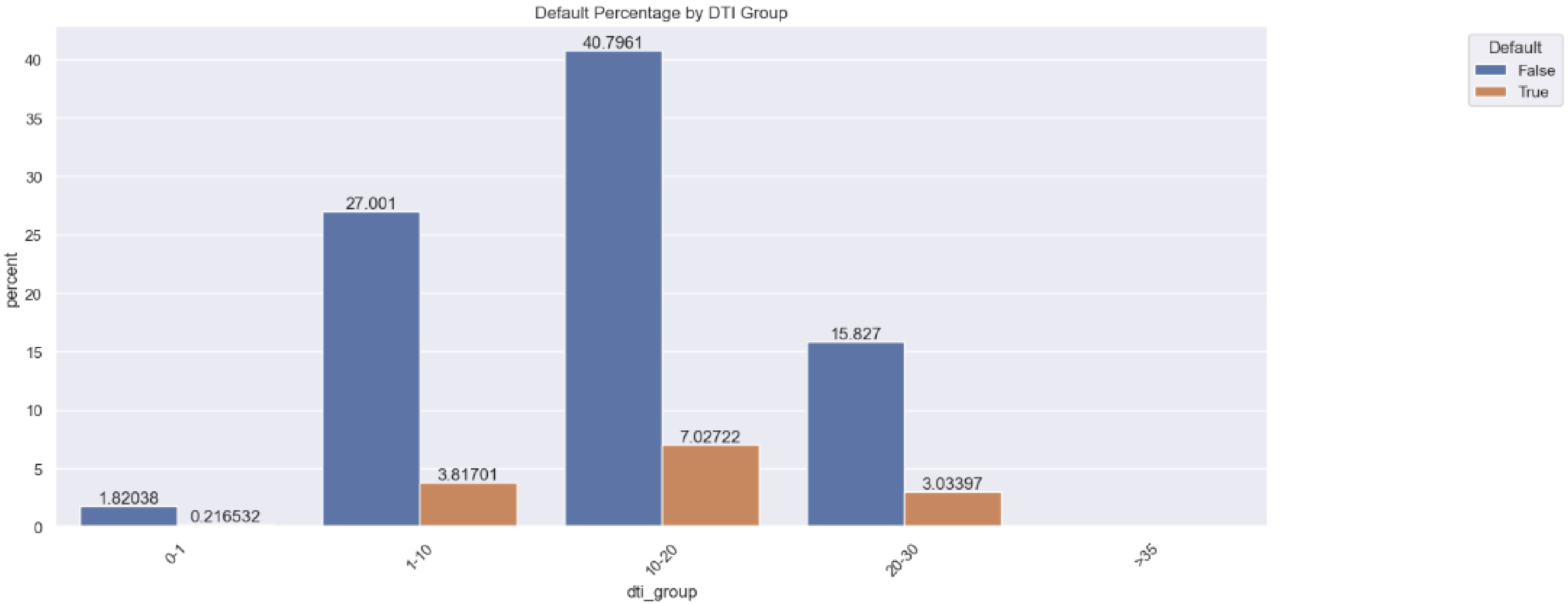
Annual Income

- Applicants between 500-700K tend to default more. Also, applicants less than 25K



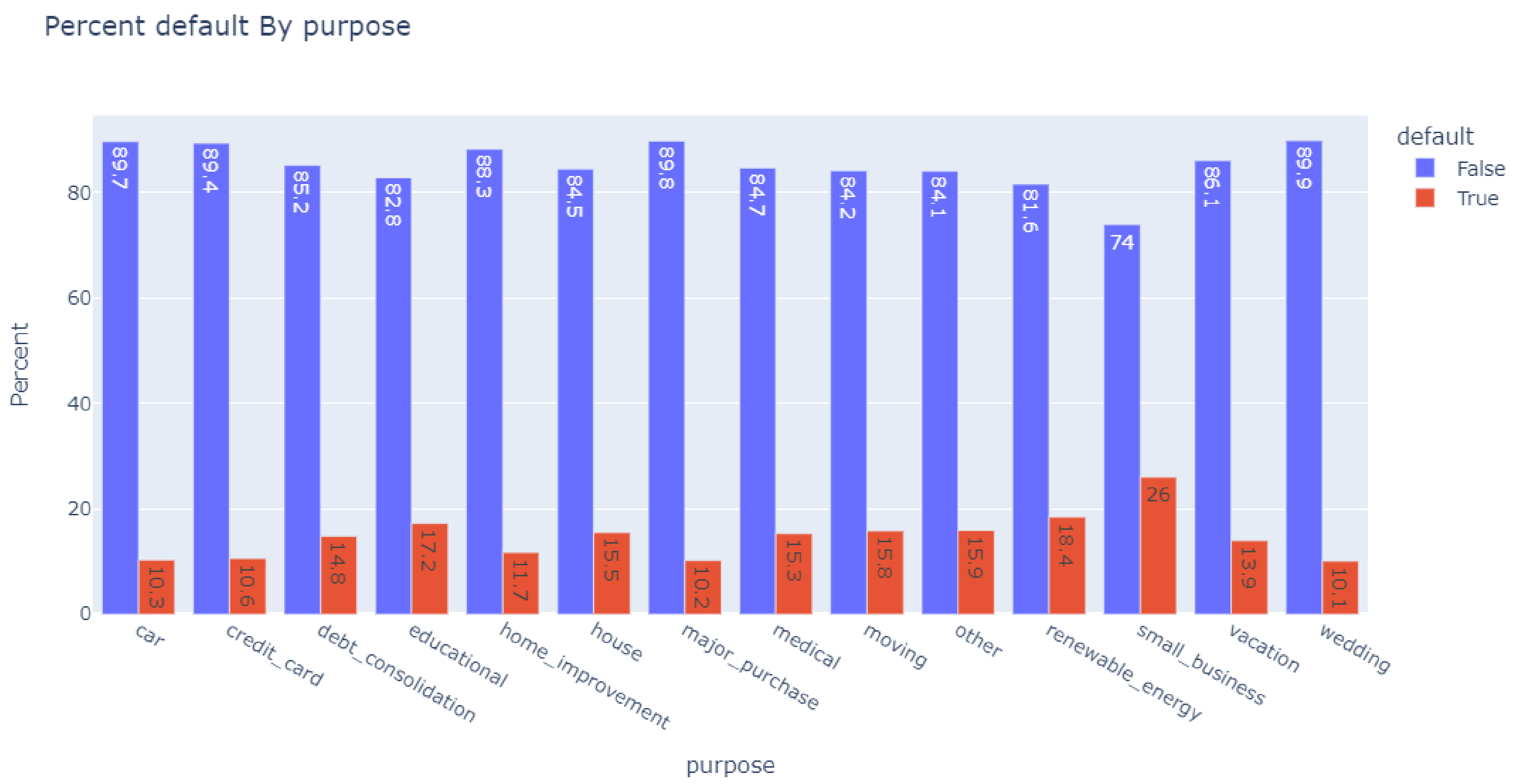
DTI

- More DTI ratio tends to have more defaults



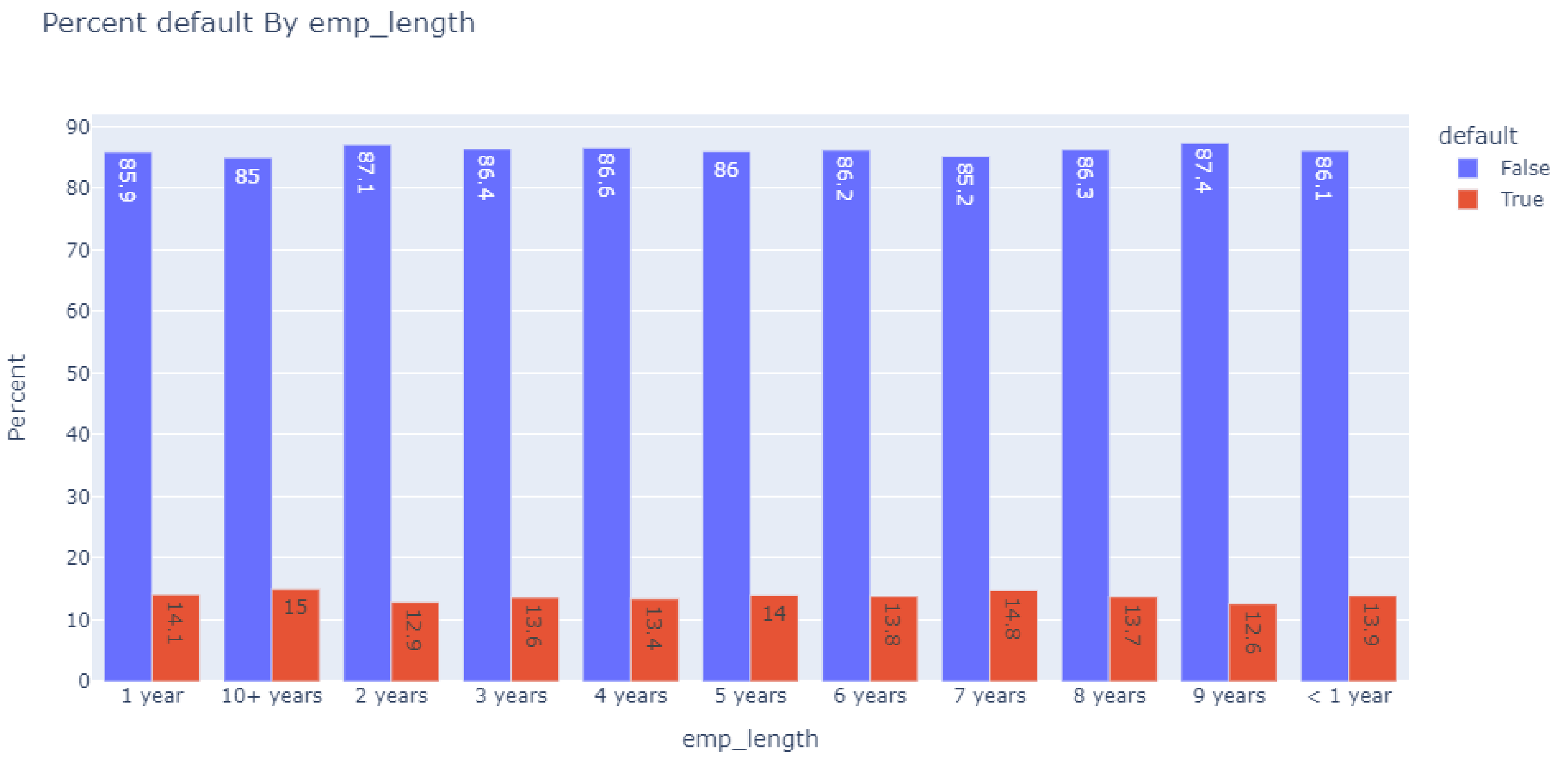
Purpose

- Loan Purpose with 'Small business' and 'Renewable Energy has more percent of defaulters



Emp Length

- Emp length does not seem to impact Loan status



Observations

- Applicants are likely to default
  - Coming from few states ‘NE’ and zipcode starting with 999,663,352
  - High DTI
  - ‘Renewable energy’ and ‘Small business’ Purposes
  - Higher grades
  - Annual Income with 500-700K tend to default more. Also, applicants less than 25K
  - High Interest rate
- We can consider them with high interest rates for the following
  - ‘Renewable energy’ and ‘Small business’ Purposes
  - Higher grades
  - Annual Income with 500-700K tend to default more. Also, applicants less than 25K
- Defaulters are not impacted by
  - Employee length
  - Revolving balance
  - Loan amount