Exploratory Data Analysis (EDA) – Credit Card Fraud Detection Case Study

KARTIK SONI

Overview

Lots of financial losses are caused every year due to credit card fraud transactions, the financial industry has switched from a posterior investigation approach to an a priori predictive approach with the design of fraud detection algorithms to warn and help fraud investigators.

This case study is focused to give you an idea of applying Exploratory Data Analysis (EDA) in a real business scenario. In this case study, apart from applying the various Exploratory Data Analysis (EDA) techniques, you will also develop a basic understanding of risk analytics and understand how data can be utilized in order to minimise the risk of losing money while lending to customers.

Flow of the case study:

1. Business understanding
2. Data loading and clearing
3. Data Analysis
4. Derived Metrics
5. Univariate Analysis
6. Bi-Variate Analysis
7. Conclusion

Business Problem Understanding

The loan providing companies find it hard to give loans to people due to their inadequate or missing credit history. Some consumers use this to their advantage by becoming a defaulter. Let us consider your work for a consumer finance company that specialises in lending various types of loans to customers. You must use **Exploratory Data Analysis (EDA)** to analyse the patterns present in the data which will make sure that the loans are not rejected for the applicants capable of repaying.

When the company receives a loan application, the company has to rights for loan approval based on the applicant's profile. Two types of risks are associated with the bank's or company's decision:

- If the aspirant is likely to repay the loan, then not approving the loan tends in a business loss to the company
- If the aspirant is not likely to repay the loan, i.e. he/she is likely to default/fraud, then approving the loan may lead to a financial loss for the company.

The data contains information about the loan application.

When a client applies for a loan, there are four types of decisions that could be taken by the bank/company:

1. **Approved**

---

2. **Cancelled**
3. **Refused**
4. **Unused offer:** The loan has been cancelled by the applicant but at different stages of the process.

This case study aims to identify patterns that indicate if an applicant will repay their instalments which may be used for taking further actions such as denying the loan, reducing the amount of loan, lending at a higher interest rate, etc. This will make sure that the applicants capable of repaying the loan are not rejected. Recognition of such aspirants using Exploratory Data Analysis (EDA) techniques is the main focus of this case study.

Data Loading and inspection

```
[268] from google.colab import drive
      drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[269] #import the warnings.
      import warnings
      warnings.filterwarnings("ignore")
```

```
[270] #import the useful libraries.
      import pandas as pd, numpy as np
      import matplotlib.pyplot as plt, seaborn as sns
      %matplotlib inline
```

```
[271] #read the data set of in inp0.
      inp0= pd.read_csv("/content/drive/MyDrive/Upgrad Colab/EDA Case Study/loan.csv")
      inp0.head()
```

Handling the null values

```
[275] #converting into percentage:
      inp0.isnull().sum()/len(inp0.index)*100

      id                            0.000000
      member_id                     0.000000
      loan_amnt                     0.000000
      funded_amnt                   0.000000
      funded_amnt_inv               0.000000
                                      ...
      tax_liens                     0.098195
      tot_hi_cred_lim             100.000000
      total_bal_ex_mort           100.000000
      total_bc_limit              100.000000
      total_il_high_credit_limit  100.000000
      Length: 111, dtype: float64
```

We can see that a lot of columns have 100 percent missing values so removing them first

```
[276] missing_col = inp0.columns[100*(inp0.isnull().sum()/len(inp0.index)) > 90]
      print(missing_col)

      Index(['mths_since_last_record', 'next_pymnt_d', 'mths_since_last_major_derog',
             'annual_inc_joint', 'dti_joint', 'verification_status_joint',
             'tot_coll_amt', 'tot_cur_bal', 'open_acc_6m', 'open_il_6m',
             'open_il_12m', 'open_il_24m', 'mths_since_rcnt_il', 'total_bal_il',
             'il_util', 'open_rv_12m', 'open_rv_24m', 'max_bal_bc', 'all_util',
             'total_rev_hi_lim', 'inq_fi', 'total_cu_tl', 'inq_last_12m',
             'acc_open_past_24mths', 'avg_cur_bal', 'bc_open_to_buy', 'bc_util',
             'mo_sin_old_il_acct', 'mo_sin_old_rev_tl_op', 'mo_sin_rcnt_rev_tl_op',
             'mo_sin_rcnt_tl', 'mort_acc', 'mths_since_recent_bc',
             'mths_since_recent_bc_dlq', 'mths_since_recent_inq',
             'mths_since_recent_revol_delinq', 'num_accts_ever_120_pd',
             'num_actv_bc_tl', 'num_actv_rev_tl', 'num_bc_sats', 'num_bc_tl',
             'num_il_tl', 'num_op_rev_tl', 'num_rev_accts', 'num_rev_tl_bal_gt_0',
             'num_sats', 'num_tl_120dpd_2m', 'num_tl_30dpd', 'num_tl_90g_dpd_24m',
             'num_tl_op_past_12m', 'pct_tl_nvr_dlq', 'percent_bc_gt_75',
             'tot_hi_cred_lim', 'total_bal_ex_mort', 'total_bc_limit',
             'total_il_high_credit_limit'],
            dtype='object')
```

```
[277] inp0 = inp0.drop(missing_col, axis=1)
      print(inp0.shape)

      (39717, 55)
```

```
[278] inp0.loc[:, ['desc', 'mths_since_last_delinq']].head()
```

	desc	mths_since_last_delinq
0	Borrower added on 12/22/11 > I need to upgra...	NaN
1	Borrower added on 12/22/11 > I plan to use t...	NaN
2	NaN	NaN
3	Borrower added on 12/21/11 > to pay for prop...	35.0
4	Borrower added on 12/21/11 > I plan on combi...	38.0

```
[279] #droping the columns with more than 30% missing values.
      inp0 = inp0.drop(['desc', 'mths_since_last_delinq'], axis=1)
```

```
[280] #sumarizing the columns
      100*(inp0.isnull().sum()/len(inp0.index))

      id                  0.000000
      member_id           0.000000
      loan_amnt           0.000000
      funded_amnt         0.000000
      funded_amnt_inv     0.000000
      term                0.000000
```

Theoretically, 25 to 30% is the maximum missing values are allowed, beyond which we might want to drop the variable from analysis. But practically we get variables with ~50% of missing values but still, the customer insists to have it for analyzing. In those cases, we have to treat them accordingly. Here, we will remove columns with null values of more than 35% after observing those columns.

Univariate Analysis

First, let's look at the overall default rate. The overall default rate is about 14%.

Let's first visualise the average default rates across categorical variables:

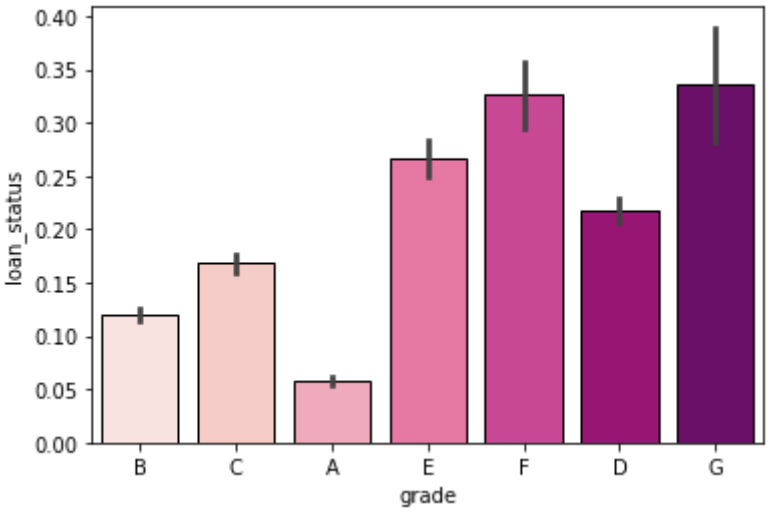
We can see that plotting default rates across grade of the loan gives us this graph from here we can infer that how loan_status varies with grade.

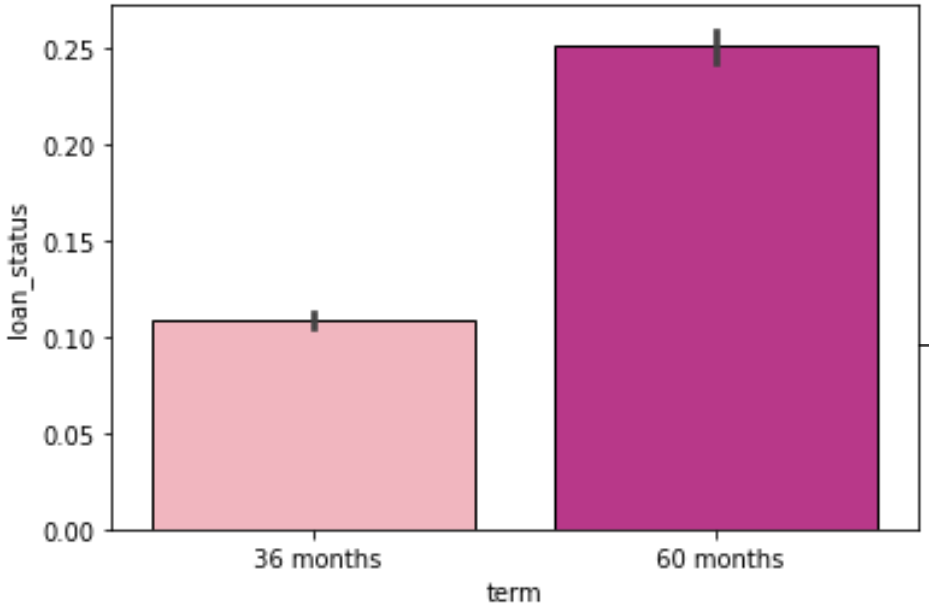
Clearly, as the grade of loan goes from A to G, the default rate increases. This is expected because the grade is decided by Lending Club based on the riskiness of the loan.

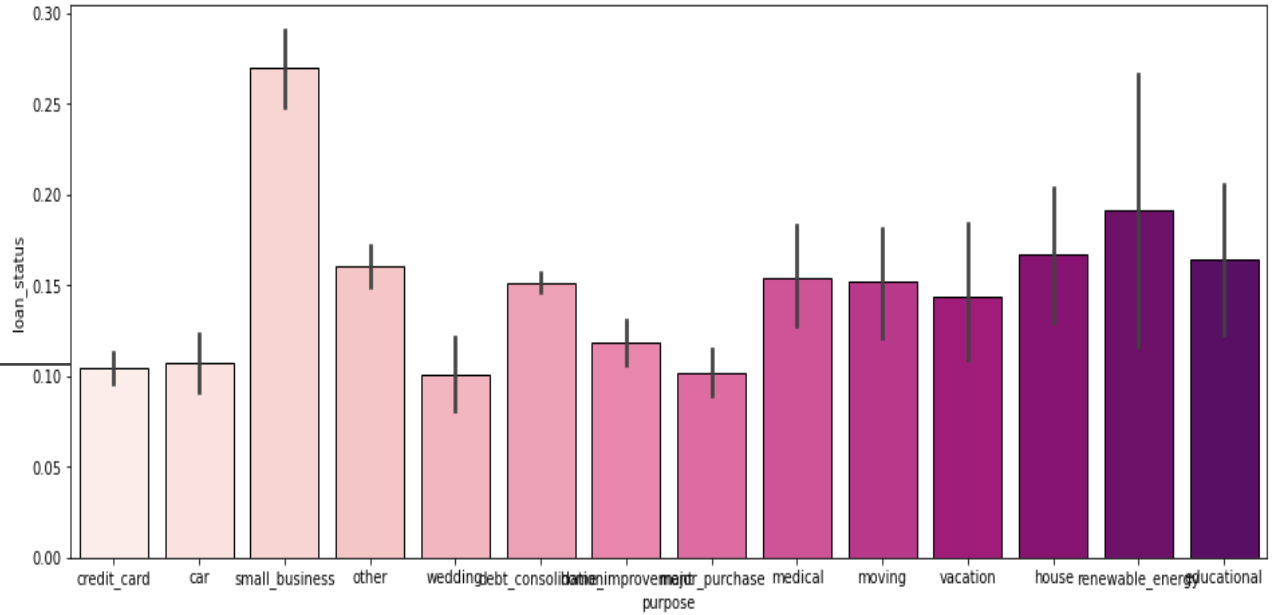
First, let's look at the overall default rate.

```
[ ]  # default rate
     round(np.mean(df['loan_status']), 2)
```

```
0.14000000000000001
```







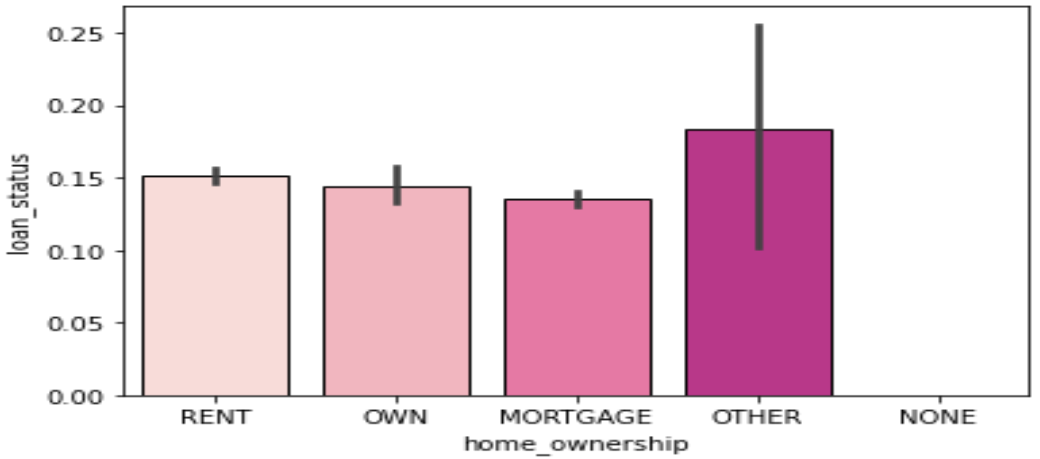
Term: 60 months loans default more than 36 months loans

From these plots we can clearly see that the longer the term period the more chances are of default.

Home ownership: not a great discriminator ->

We can see no differentiation could be made on the basis of type of home, all are equally likely to default.

Small business loans default the most, then renewable energy and education.



Variation of defaults across time period:

```
# let's also observe the distribution of loans across years
# first lets convert the year column into datetime and then extract year and month from it
df['issue_d'].head()
```

```
0    Dec-11
1    Dec-11
2    Dec-11
3    Dec-11
5    Dec-11
Name: issue_d, dtype: object
```

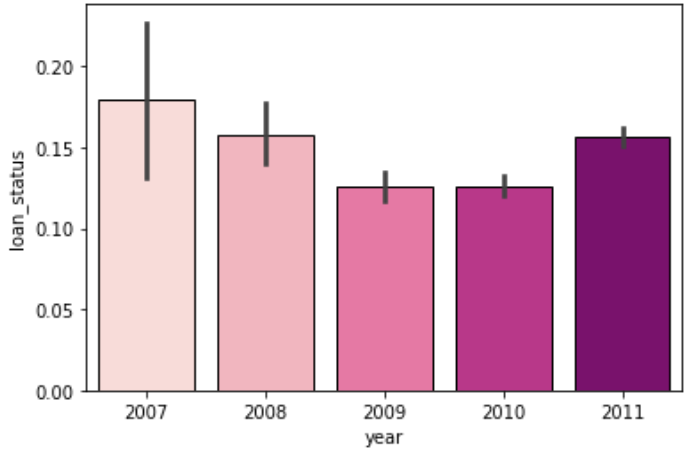
```
[ ] from datetime import datetime
    df['issue_d'] = df['issue_d'].apply(lambda x: datetime.strptime(x, '%b-%y'))
```

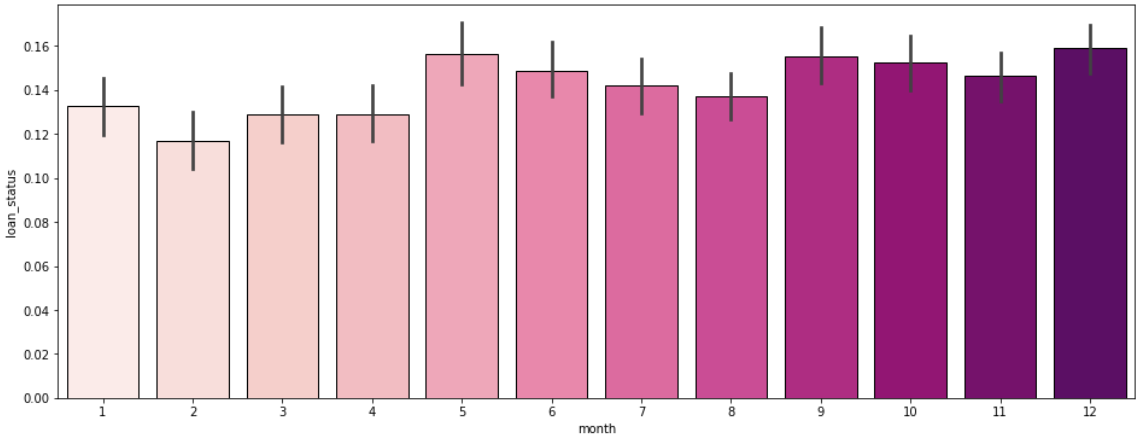
```
[ ] # extracting month and year from issue_date
    df['month'] = df['issue_d'].apply(lambda x: x.month)
    df['year'] = df['issue_d'].apply(lambda x: x.year)
```

```
[ ] # let's first observe the number of loans granted across years
    df.groupby('year').year.count()
```

```
year
2007      251
2008     1562
2009     4716
2010    11214
2011    19801
Name: year, dtype: int64
```

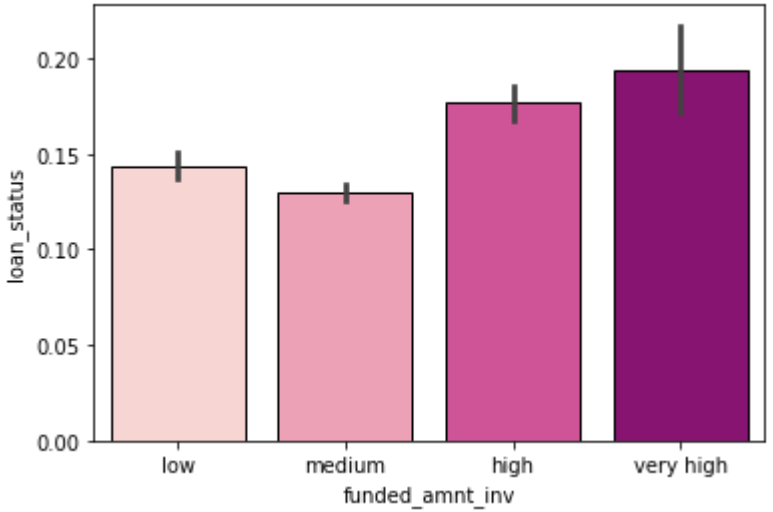
You can see that the number of loans has increased steadily across years.

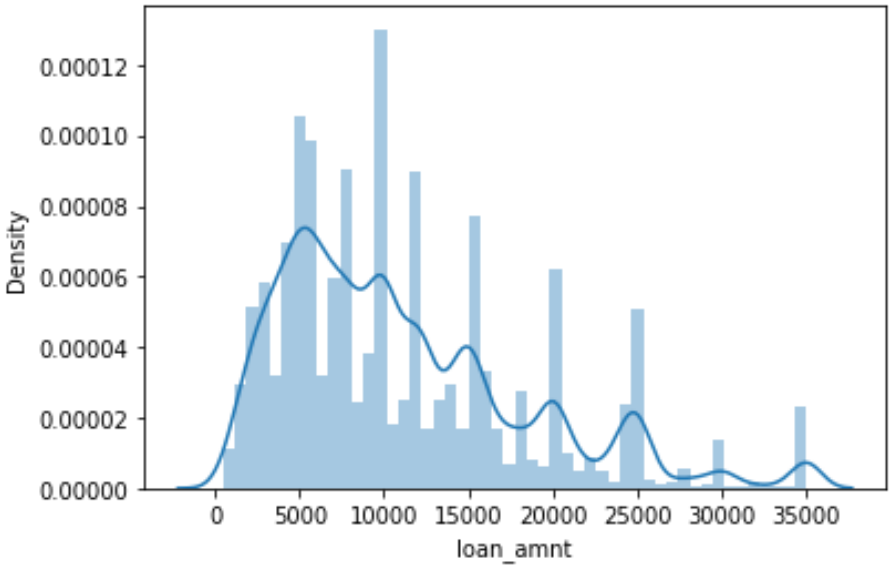




Analyzing how the default rate varies across continuous variables:

- Loan amount: the median loan amount is around 10,000.
- Plot of funded amount invested vs loan status



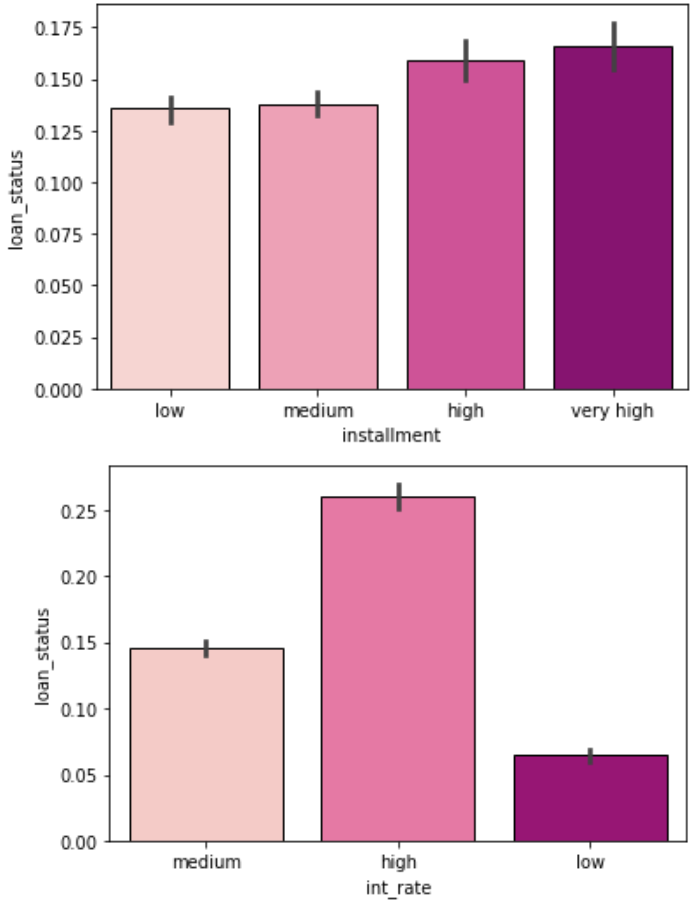


- We can infer from above graphs that higher the loan amount, higher the default rate.

Comparing default rates across installment and interest rates:

- The higher the installment amount, the higher the default rate.

- High interest rates translates into higher default rates, as expected

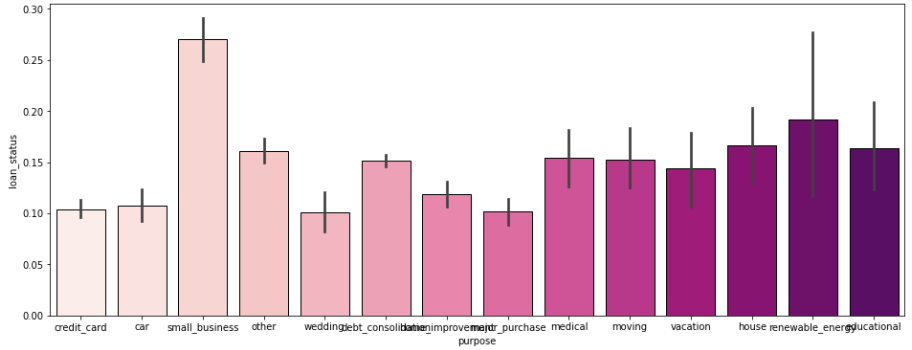


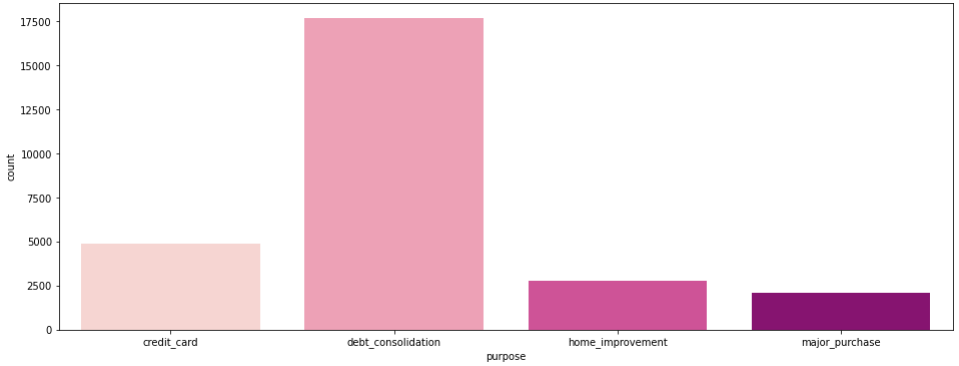
Segmented Univariate Analysis

We have now compared the default rates across various variables, and some of the important predictors are purpose of the loan, interest rate, annual income, grade etc.

In the credit industry, one of the most important factors affecting default is the purpose of the loan - home loans perform differently than credit cards, credit cards are very different from debt consolidation loans etc.

Small business loans default the most, then renewable energy and education.



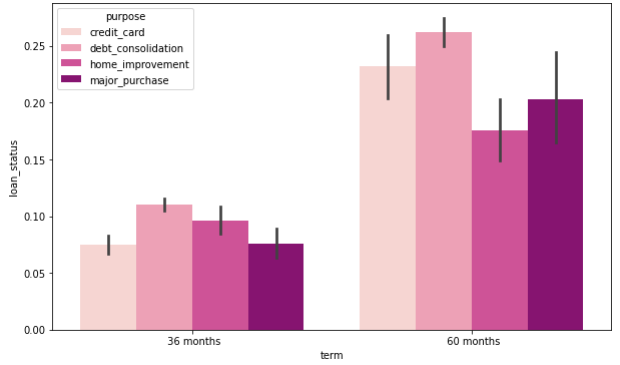


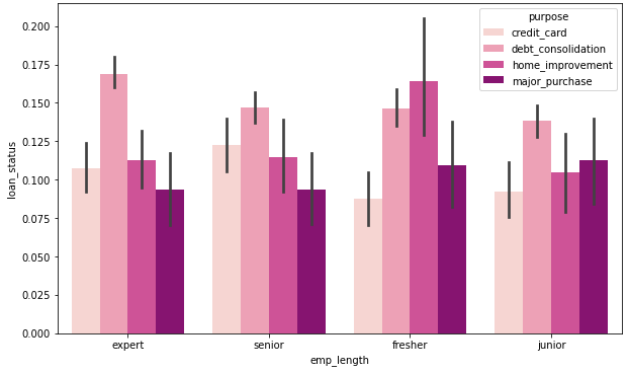
Comparing multicategory variables that changes:

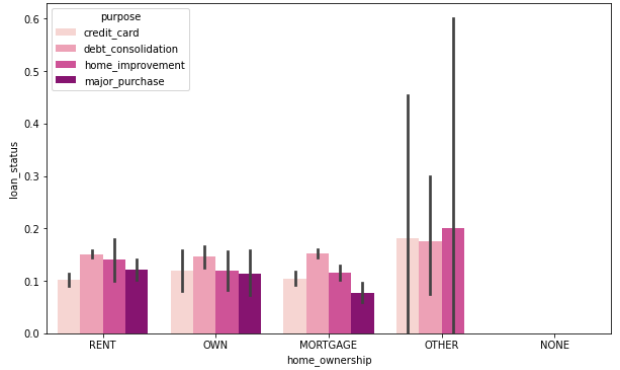
Let's now compare the default rates across two types of categorical variables.

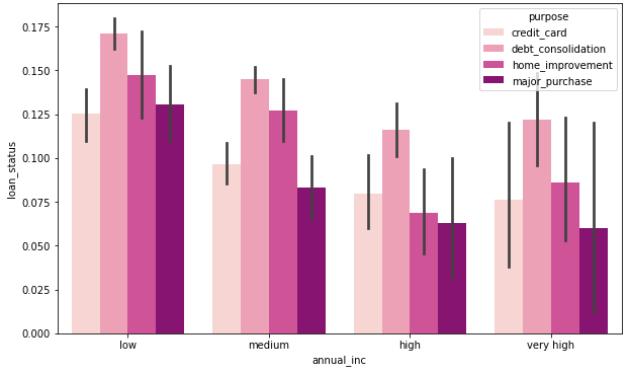
Purpose of loan (constant) and another categorical variable (which changes).

Thus, there is a 6% increase in default rate as you go from high to low annual income. We can compute this difference for all the variables and roughly identify the ones that affect default rate the most.









Conclusion:
Target Variable -> LOAN STATUS

Top-5 Major variables to consider for loan prediction:

- Purpose of Loan
- Employment Length
- Grade
- Interest Rate
- Term

EDA for the banking data set revealed that :

- The proportion of defaulters is 8.7%
- The bank lends more to females
- More Cash Loans go into default: bank should give more Revolving Loans.
- Proportion of Working defaults more / State Servants less
- People with Higher Education; Older People default less
- Single People Default more : giving loans to married safer
- Higher Amount loans; Higher Income — less defaults
- Longer employment history, longer registration days less default
- Loans Previously Refused or Cancelled - higher default rate

THANKS