Lending_Club_Case_Study_Project_Ajit_Siddhant

Project done by :-

Ajit John

Siddhant Srivastava

Lending Club :EDA case study

- There are four major parts that are needed to be done for this case study:
  - 1. Data understanding
  - 2. Data cleaning (cleaning missing values, removing redundant columns etc.)
  - 3. Univariate analysis
  - 4. Bivariate analysis
  - 5.Observations and Results

Data Cleaning

1. Firstly, we are going to check the percentage of missing values
2. Secondly, we will remove all those with very high missing percentage
3. Next ,for columns with less missing percentage we will be performing Imputations
4. Then, we will just identify the correct metric to impute the column.
5. Finally, we will be dropping rows where the missing percentage is quite high

Data Analysis

i. The objective is to identify predictors of default so that at the time of loan application, we can use those variables for approval/rejection of the loan.

    i. There are broadly three types of variables –
    ii. those which are related to the applicant (demographic variables such as age, occupation, employment details etc.),
    iii. Loan characteristics (amount of loan, interest rate, purpose of loan etc.) and
    iv. Customer behavior variables (those which are generated after the loan is approved such as delinquent 2 years, revolving balance, next payment date etc.).

ii. Now, the customer behavior variables are not available at the time of loan application, and thus they cannot be used as predictors for credit approval.

iii. The ones marked 'current' are neither fully paid not defaulted, so get rid of the current loans. Also, tag the other two values as 0 or 1 to make your analysis simple and clean.

Data Analysis

- **Data Analysis: Univariate Analysis**

- For univariate analysis, we will check the default rate across various categorical features.
- For continuous features, we will perform binning and then you may perform univariate analysis.

- **Data Analysis: Bivariate Analysis**

- Here we are planning to choose two or more features to understand the Default variable

Variables those can be dropped

id , member_id, emp_title, url, desc, zip_code, addr_state, funded_amnt (as funded_amnt_inv is present), title (as purpose is present),

In [19]:
```
# Dropping columns mentioned above
loan_data.drop(columns=["id" , "member_id", "emp_title", "url", "desc", "zip_code", "addr_state", "funded_amnt", "title"], inplace=True)
print(loan_data.shape)
loan_data.columns
```

Out[19]:
```
(38575, 33)
Index(['loan_amnt', 'funded_amnt_inv', 'int_rate', 'installment', 'grade',
       'sub_grade', 'emp_length', 'home_ownership', 'annual_inc', 'purpose',
       'dti', 'delinq_2yrs', 'inq_last_6mths', 'open_acc', 'revol_bal',
       'revol_util', 'total_pymnt', 'total_pymnt_inv', 'total_rec_prncp',
       'total_rec_int', 'total_rec_late_fee', 'recoveries',
       'collection_recovery_fee', 'last_pymnt_d', 'last_pymnt_amnt',
       'last_credit_pull_d', 'verified', 'term_36_not60', 'loan_paid',
       'issue_d_year', 'issue_d_month', 'credit_age', 'closed_acc'],
      dtype='object')
```

Removing skewed data

In [24]:

```
def getBoxPlot(df: pd.DataFrame, percentile: float) -> None:
    # for attr in ("loan_amnt", "int_rate", "installment", "annual_inc", "dti", "open_acc", "closed_acc"):
    for attr in ("installment", "annual_inc",):
        plt.figure()
        (df.loc[df[attr] < df[attr].quantile(percentile)]).boxplot(attr)
        print(attr, df[attr].quantile(percentile))

# for x in (1.0, 0.99, 0.98, 0.97):
for x in (0.94, 0.93, 0.9):
    print(f"percentile: {x}")
    getBoxPlot(loan_data, x)
```

```
percentile: 0.94
installment 709.8367999999999
annual_inc 134000.0
percentile: 0.93
installment 682.74
annual_inc 127500.0
percentile: 0.9
installment 620.9
```

Removing sub-categories with very less data

In [29]:

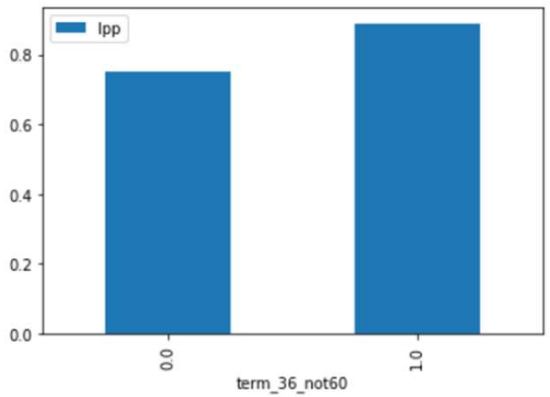
```
'''
Too less data for G under grade
Too less data for NONE under home_ownership
Too less data for OTHER under home_ownership
Too less data for renewable_energy under purpose
Too less data for 4 under inq_last_6mths
Too less data for 5 under inq_last_6mths
Too less data for 6 under inq_last_6mths
Too less data for 7 under inq_last_6mths
Too less data for 8 under inq_last_6mths
'''
loan.drop(index=loan.index[loan["home_ownership"] == 'NONE'], inplace=True)
loan.drop(index=loan.index[loan["home_ownership"] == 'OTHER'], inplace=True)
loan.drop(index=loan.index[loan["inq_last_6mths"] >= 4], inplace=True)
```

Univariate & Segmented Univariate Analysis

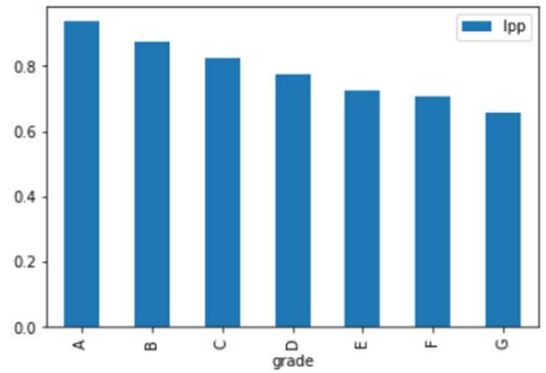
```
def getOrderedList(allVals: list) -> list:
    months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
              "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
    if type(allVals[0]) == str and (allVals[0].split('_')[-1]).isdigit():
        sortedAllVals = sorted(allVals, key=lambda d: int(d.split('_')[-1]))
    else:
        if allVals[0] in months:
            sortedAllVals = months
        else:
            sortedAllVals = sorted(allVals)
    return sortedAllVals
```

```
for attr in ("funded_amnt_inv_groups",
             "grade",
             "sub_grade",
             "emp_length",
             "home_ownership",
             "purpose",
             "inq_last_6mths",
             "term_36_not60",
             "loan_paid",
             "issue_d_year",
             "issue_d_month",
             "loan_amnt_groups",
             "int_rate_groups",
             "installment_groups",
             "annual_inc_groups",
             "dti_groups",
             "credit_age_groups",
             "closed_acc_groups",
             "open_acc_groups",
             "revol_util_groups",
    ):
    d = dict()
    sortedAllVals = getOrderedList(loan[attr].value_counts().index.values)
    for val in sortedAllVals:
        if len(loan.loc[loan[attr] == val]) == 0:
            print(f"No data for {val} under {attr}")
            continue
        elif len(loan.loc[loan[attr] == val]) < len(loan)/(10*len(loan[attr].value_counts().index.values)):
            print(f"Too less data for {val} under {attr}")
        d[val] = len(loan.loc[(loan[attr] == val) & (loan["loan_paid"] == 1)])/len(loan.loc[loan[attr] == val])
    plt.figure()
    (pd.DataFrame({attr:d.keys(), "lpp"  : d.values()})).plot.bar(x=attr, y="lpp")
    plt.figure(figsize=(15, 5))
    sns.countplot(x=attr, hue="loan_paid", data=loan, order=d.keys())
```

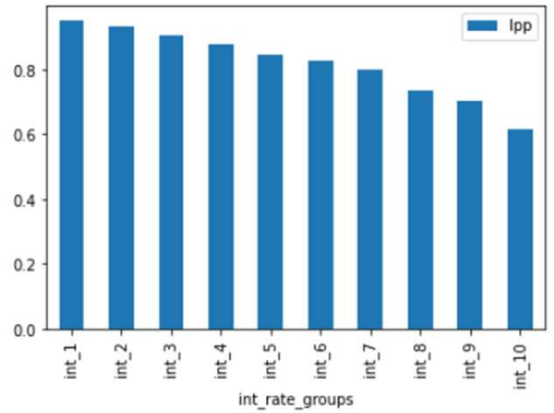
<Figure size 432x288 with 0 Axes>



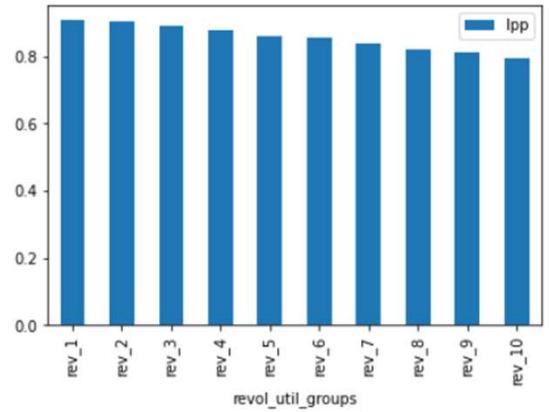
<Figure size 432x288 with 0 Axes>



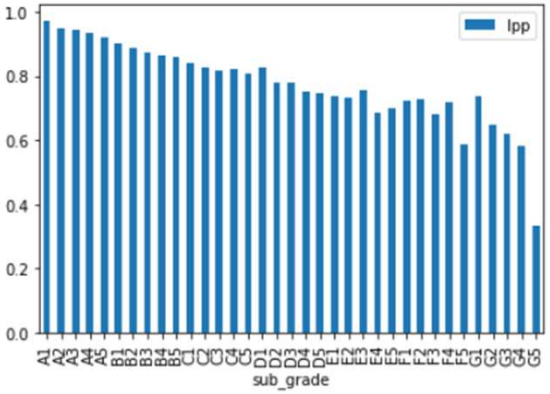
<Figure size 432x288 with 0 Axes>



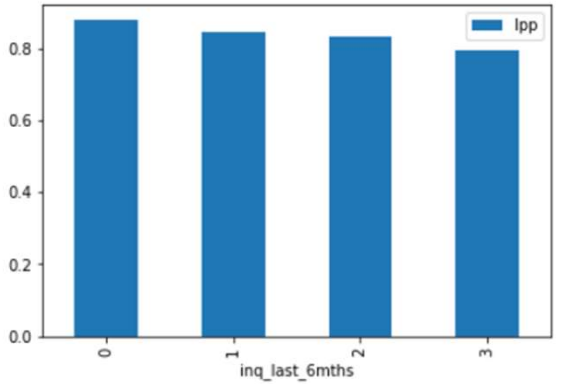
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



Observations (Univariate Analysis)

Indicators for loan payment:

Higher Grade or Sub-Grade

Lower interest-rate

Lower revolving utility percentage

Less enquired in last six months (inq_last_6mths).

Term is 36 months.

If purpose is "wedding", "major purchage", credit-card" & "car".

Indicators for loan default:

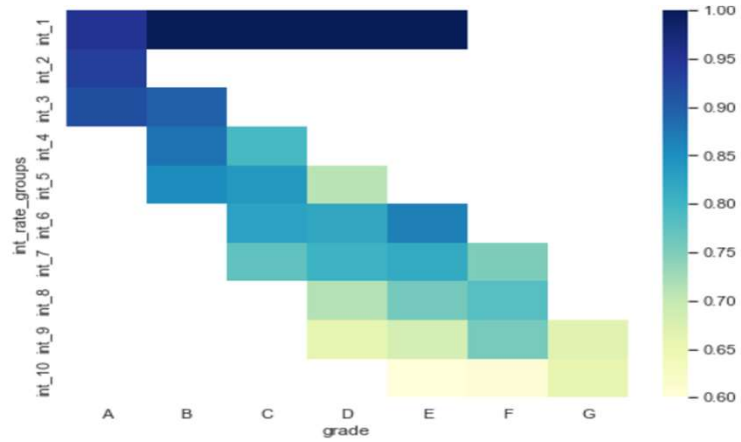
purpose is marked as 'other'

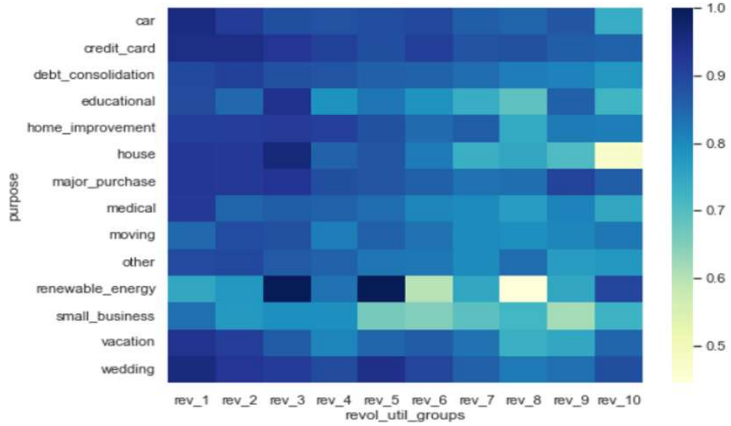
Bivariate Analysis

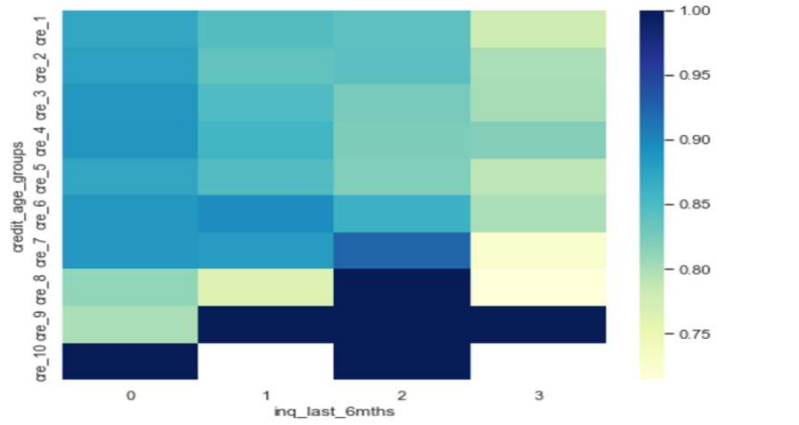
Now that we have analyzed each of the variables and its impact on the loan-status, let us take group of variables together and and analyze their combined effect on the loan-status. These categories are based on our business understanding. The original distribution column shows the average trend in all the data and we compare that with the data after applying our conditions.

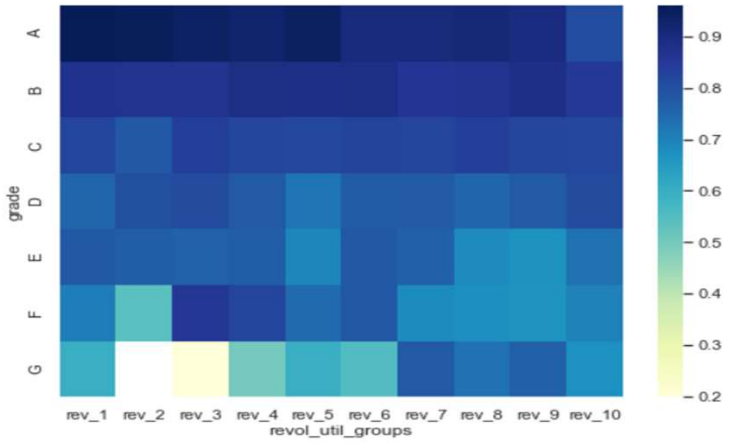
```
cat_var = ["grade", "sub_grade", "emp_length", "home_ownership", "purpose", "inq_last_6mths",
           "verified", "term_36_not60", "issue_d_year", "issue_d_month", "loan_amnt_groups", "int_rate_groups",
           "installment_groups", "annual_inc_groups", "dti_groups", "credit_age_groups", "funded_amnt_inv_groups",
           "revol_util_groups"]
```

```
colors = ["#a9fea9","#fca9a9"]
sns.set(style='whitegrid', palette=sns.color_palette(colors))
by = 'loan_status'
order = ['Fully Paid','Charged Off']
def bivariateCategoricalAna (data: pd.DataFrame, x_axis, y_axis, title=None, figsize=(8,6)) -> None:
    """
        Parameters:
            data : defaulted at the 'loan' variable, but can be changed externally if needed
            x, y : columns of data in the corresponding axis, both categorical
            figsize : a default have is given, but can be overridden
        Returns:
            None, does a heatmap plot
    """
    fig, ax = plt.subplots(figsize=figsize)
    pt = pd.pivot_table(data=data, values='loan_paid', index=y_axis, columns=x_axis)
    sns.heatmap(pt, ax=ax, cmap='YlGnBu')
    if title is not None:
        plt.title(title)
    plt.show()
```


int_rate_groups
int_1
int_2
int_3
int_4
int_5
int_6
int_7
int_8
int_9
int_10
A
B
C
D
E
F
G
grade
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60


purpose
car
credit_card
debt_consolidation
educational
home_improvement
house
major_purchase
medical
moving
other
renewable_energy
small_business
vacation
wedding
rev_1
rev_2
rev_3
rev_4
rev_5
rev_6
rev_7
rev_8
rev_9
rev_10
revol_util_groups
1.0
0.9
0.8
0.7
0.6
0.5


credit_age_groups
cre_1
cre_2
cre_3
cre_4
cre_5
cre_6
cre_7
cre_8
cre_9
cre_10
0
1
2
3
inq_last_6mths
1.00
0.95
0.90
0.85
0.80
0.75


grade
A
B
C
D
E
F
G
rev_1
rev_2
rev_3
rev_4
rev_5
rev_6
rev_7
rev_8
rev_9
rev_10
revol_util_groups
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

Observations (BiVariate Analysis)

- *Interest-rate & grade is highly corelated : higher grade indicates lower interest rate.*

- *Grade is a strong indicator. Higher grade (A) will have high probability to pay loan.*

- *Revolving utility percentage (lower utility leads to higher loan pay probability) has low impact on "wedding", "major purchage", credit-card" & "car" purpose category.*

In higher credit-age group (554 month +), trend for inq_last_6mths acts opposite relative to univariate analysis outcome.

If term is 36 months, loan-payment probability is higher as indicated by univariate analysis.

Higher income group(102k+) have high probability of paying loan.