# Hackathon1

March 17, 2024

[51]:
```python
import warnings
warnings.filterwarnings("ignore")

import math
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from imblearn.over_sampling import SMOTE
from xgboost import XGBClassifier
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.impute import SimpleImputer, KNNImputer
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    confusion_matrix,
    classification_report,
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    make_scorer
)
from sklearn.ensemble import (
    BaggingClassifier,
    RandomForestClassifier,
    GradientBoostingClassifier,
    AdaBoostClassifier,
    StackingClassifier
)
from sklearn.model_selection import (
    train_test_split,
    StratifiedKFold,
    cross_val_score,
```

```
    GridSearchCV,
    RandomizedSearchCV
)

# to suppress scientific notations
pd.set_option('display.float_format', lambda x: '%.3f' % x)
%matplotlib inline
sns.set()
```

[52]:
```
data = pd.read_csv('train_loan_data (1).csv')
df = data.copy()
```

[53]:
```
df.head()
```

[53]:
```
  addr_state  annual_inc earliest_cr_line emp_length  \
0         CO   85000.000           Jul-97  10+ years
1         CA   40000.000           Apr-87  10+ years
2         FL   60000.000           Aug-07  10+ years
3         IL  100742.000           Sep-80  10+ years
4         MD   80000.000           Jul-99  10+ years

                        emp_title  fico_range_high  fico_range_low grade  \
0                          Deputy              744             740     E
1  Department of Veterans Affairs              724             720     B
2                 Marble polishing              679             675     B
3                          printer              664             660     B
4                   Southern Mgmt              669             665     F

  home_ownership application_type  … pub_rec_bankruptcies  \
0       MORTGAGE       Individual  …                0.000
1           RENT       Individual  …                0.000
2       MORTGAGE       Individual  …                0.000
3       MORTGAGE       Individual  …                0.000
4           RENT       Individual  …                0.000

              purpose  revol_bal  revol_util  sub_grade        term  \
0  debt_consolidation       5338      93.600         E1   60 months
1  debt_consolidation      19944      60.300         B1   36 months
2  debt_consolidation      23199      88.500         B5   36 months
3  debt_consolidation      18425      69.000         B2   36 months
4  debt_consolidation      34370      90.000         F5   60 months

                title  total_acc  verification_status loan_status
0  Debt consolidation          8      Source Verified   Defaulted
1         Credit Loan         12             Verified        Paid
2  Debt consolidation         16      Source Verified        Paid
3  Debt consolidation         19      Source Verified        Paid
```

```
4  Debt Connsolidation         59             Verified        Paid

[5 rows x 28 columns]
```

[54]:
```python
df.shape
```

[54]:
```
(80000, 28)
```

[55]:
```python
df.replace({'loan_status':{'Paid': 0, 'Defaulted':1}}, inplace=True)
```

[56]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 80000 entries, 0 to 79999
Data columns (total 28 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   addr_state            80000 non-null  object
 1   annual_inc            80000 non-null  float64
 2   earliest_cr_line      80000 non-null  object
 3   emp_length            75412 non-null  object
 4   emp_title             74982 non-null  object
 5   fico_range_high       80000 non-null  int64
 6   fico_range_low        80000 non-null  int64
 7   grade                 80000 non-null  object
 8   home_ownership        80000 non-null  object
 9   application_type      80000 non-null  object
 10  initial_list_status   80000 non-null  object
 11  int_rate              80000 non-null  float64
 12  loan_amnt             80000 non-null  int64
 13  num_actv_bc_tl        76052 non-null  float64
 14  mort_acc              77229 non-null  float64
 15  tot_cur_bal           76052 non-null  float64
 16  open_acc              80000 non-null  int64
 17  pub_rec               80000 non-null  int64
 18  pub_rec_bankruptcies  79969 non-null  float64
 19  purpose               80000 non-null  object
 20  revol_bal             80000 non-null  int64
 21  revol_util            79947 non-null  float64
 22  sub_grade             80000 non-null  object
 23  term                  80000 non-null  object
 24  title                 79030 non-null  object
 25  total_acc             80000 non-null  int64
 26  verification_status   80000 non-null  object
 27  loan_status           80000 non-null  int64
dtypes: float64(7), int64(8), object(13)
memory usage: 17.1+ MB
```

[57]:
```
df.head()
```

[57]:
```
  addr_state  annual_inc earliest_cr_line emp_length  \
0         CO   85000.000           Jul-97  10+ years
1         CA   40000.000           Apr-87  10+ years
2         FL   60000.000           Aug-07  10+ years
3         IL  100742.000           Sep-80  10+ years
4         MD   80000.000           Jul-99  10+ years

                        emp_title  fico_range_high  fico_range_low grade  \
0                          Deputy              744             740     E
1  Department of Veterans Affairs              724             720     B
2                Marble polishing              679             675     B
3                         printer              664             660     B
4                   Southern Mgmt              669             665     F

  home_ownership application_type  … pub_rec_bankruptcies  \
0       MORTGAGE       Individual  …                0.000
1           RENT       Individual  …                0.000
2       MORTGAGE       Individual  …                0.000
3       MORTGAGE       Individual  …                0.000
4           RENT       Individual  …                0.000

              purpose  revol_bal  revol_util  sub_grade        term  \
0  debt_consolidation       5338      93.600         E1   60 months
1  debt_consolidation      19944      60.300         B1   36 months
2  debt_consolidation      23199      88.500         B5   36 months
3  debt_consolidation      18425      69.000         B2   36 months
4  debt_consolidation      34370      90.000         F5   60 months

                  title  total_acc  verification_status loan_status
0    Debt consolidation          8      Source Verified           1
1           Credit Loan         12             Verified           0
2    Debt consolidation         16      Source Verified           0
3    Debt consolidation         19      Source Verified           0
4   Debt Connsolidation         59             Verified           0

[5 rows x 28 columns]
```

[58]:
```
df.isnull().sum().sort_values(ascending=False)
```

[58]:
```
emp_title               5018
emp_length              4588
num_actv_bc_tl          3948
tot_cur_bal             3948
mort_acc                2771
title                    970
```

```
revol_util               53
pub_rec_bankruptcies     31
open_acc                  0
verification_status       0
total_acc                 0
term                      0
sub_grade                 0
revol_bal                 0
purpose                   0
pub_rec                   0
addr_state                0
annual_inc                0
loan_amnt                 0
int_rate                  0
initial_list_status       0
application_type          0
home_ownership            0
grade                     0
fico_range_low            0
fico_range_high           0
earliest_cr_line          0
loan_status               0
dtype: int64
```

[59]:
```
df.isnull().sum()
```

[59]:
```
addr_state                0
annual_inc                0
earliest_cr_line          0
emp_length             4588
emp_title              5018
fico_range_high           0
fico_range_low            0
grade                     0
home_ownership            0
application_type          0
initial_list_status       0
int_rate                  0
loan_amnt                 0
num_actv_bc_tl         3948
mort_acc               2771
tot_cur_bal            3948
open_acc                  0
pub_rec                   0
pub_rec_bankruptcies     31
purpose                   0
revol_bal                 0
```

```
revol_util                 53
sub_grade                   0
term                        0
title                     970
total_acc                   0
verification_status         0
loan_status                 0
dtype: int64
```

[60]:
```
df.duplicated().sum()
```

[60]:
```
0
```

[61]:
```
cat_cols = df.select_dtypes(include='object').columns
df[cat_cols] = df[cat_cols].astype('category')
df.select_dtypes(include='category').columns
```

[61]:
```
Index(['addr_state', 'earliest_cr_line', 'emp_length', 'emp_title', 'grade',
       'home_ownership', 'application_type', 'initial_list_status', 'purpose',
       'sub_grade', 'term', 'title', 'verification_status'],
      dtype='object')
```

[62]:
```
for i in df.select_dtypes(include=['category']).columns:
    print('Unique values in', i, 'are :')
    print(df[i].value_counts(dropna=False))
    print('*'*50)
```

```
Unique values in addr_state are :
addr_state
CA    11744
TX     6493
NY     6461
FL     5618
IL     3098
NJ     2853
PA     2676
OH     2575
GA     2530
NC     2291
VA     2249
MI     2091
AZ     1993
MA     1862
MD     1802
CO     1790
WA     1736
MN     1414
IN     1329
```

```
MO     1298
NV     1224
TN     1207
CT     1143
WI     1043
OR     1025
SC     1007
AL      986
LA      928
KY      836
OK      725
KS      649
AR      590
UT      554
NM      440
HI      404
MS      373
NH      373
RI      356
WV      268
NE      240
MT      229
DE      219
AK      215
DC      201
SD      192
WY      187
VT      181
ME      110
ID      106
ND       85
IA        1
Name: count, dtype: int64
**************************************************
Unique values in earliest_cr_line are :
earliest_cr_line
Sep-03    547
Aug-03    545
Aug-01    544
Oct-01    541
Sep-02    539
         …
Jul-65      1
Sep-59      1
Sep-65      1
Jul-64      1
Nov-66      1
Name: count, Length: 640, dtype: int64
```

```
**************************************************
Unique values in emp_length are :
emp_length
10+ years    26278
2 years       7319
3 years       6474
< 1 year      6297
1 year        5294
5 years       5094
4 years       4763
NaN           4588
6 years       3691
7 years       3597
8 years       3583
9 years       3022
Name: count, dtype: int64
**************************************************
Unique values in emp_title are :
emp_title
NaN                                 5018
Teacher                             1278
Manager                             1194
Owner                                592
RN                                   526
                                    …
Hotel Desk Coordinator                 1
Hotel & Travel Credit Union            1
Hot oiler                              1
Hostler                                1
MyBuys                                 1
Name: count, Length: 36662, dtype: int64
**************************************************
Unique values in grade are :
grade
B    23502
C    22525
A    13996
D    11936
E     5620
F     1885
G      536
Name: count, dtype: int64
**************************************************
Unique values in home_ownership are :
home_ownership
MORTGAGE    39628
RENT        31688
OWN          8654
```

```
ANY            19
OTHER           7
NONE            4
Name: count, dtype: int64
**************************************************
Unique values in application_type are :
application_type
Individual    78446
Joint App      1554
Name: count, dtype: int64
**************************************************
Unique values in initial_list_status are :
initial_list_status
w    46745
f    33255
Name: count, dtype: int64
**************************************************
Unique values in purpose are :
purpose
debt_consolidation    46418
credit_card           17506
home_improvement       5268
other                  4683
major_purchase         1746
small_business          950
medical                 902
car                     868
moving                  548
vacation                518
house                   413
wedding                 110
renewable_energy         54
educational              16
Name: count, dtype: int64
**************************************************
Unique values in sub_grade are :
sub_grade
C1    4982
B4    4973
B5    4950
B3    4866
C2    4698
B2    4477
C3    4440
C4    4425
B1    4236
C5    3980
A5    3743
```

```
A4    3189
D1    3024
A1    2639
D2    2626
D3    2364
A3    2278
A2    2147
D4    2128
D5    1794
E1    1431
E2    1278
E3    1107
E4     911
E5     893
F1     566
F2     431
F3     354
F4     292
F5     242
G1     178
G2     151
G3      82
G4      78
G5      47
Name: count, dtype: int64
**************************************************
Unique values in term are :
term
 36 months    60750
 60 months    19250
Name: count, dtype: int64
**************************************************
Unique values in title are :
title
Debt consolidation                           39396
Credit card refinancing                      14802
Home improvement                              4542
Other                                         4035
Major purchase                                1422
                                             …
Get on the right track                           1
Get me out of debt with lower interest!          1
Get it right                                     1
Get it done                                      1
Mama to Be                                       1
Name: count, Length: 5349, dtype: int64
**************************************************
Unique values in verification_status are :
```

```
verification_status
Source Verified    30855
Verified           24876
Not Verified       24269
Name: count, dtype: int64
**************************************************
```

[63]:

```
def histogram_boxplot(feature, figsize=(15, 7), bins=None):
    """
    Boxplot and histogram combined
    feature: 1-d feature array
    figsize: size of fig (default (15,10))
    bins: number of bins (default None / auto)
    """
    f2, (ax_box2, ax_hist2) = plt.subplots(nrows = 2, # Number of rows of the␣
↪subplot grid= 2
                                           sharex = True, # x-axis will be␣
↪shared among all subplots
                                           gridspec_kw = {"height_ratios": (.
↪25, .75)},
                                           figsize = figsize
                                           ) # creating the 2 subplots

    sns.boxplot(feature, ax=ax_box2, showmeans=True, color='yellow') # boxplot␣
↪will be created and a star will indicate the mean value of the column
    sns.distplot(feature, kde=True, ax=ax_hist2, bins=bins) if bins else sns.
↪distplot(feature, kde=True, ax=ax_hist2) # For histogram
    ax_hist2.axvline(np.mean(feature), color='green', linestyle='--') # Add␣
↪mean to the histogram
    ax_hist2.axvline(np.median(feature), color='blue', linestyle='-');# Add␣
↪median to the histogram
```

[64]:

```
def perc_on_bar(feature):
    '''
    plot
    feature: categorical feature
    the function won't work if a column is passed in hue parameter
    '''
    #Creating a countplot for the feature
    sns.set(rc={'figure.figsize':(15,7)})
    ax=sns.countplot(x=feature, data=data, palette='mako')

    total = len(feature) # length of the column
    for p in ax.patches:
        # percentage of each class of the category
        percentage = 100 * p.get_height()/total
        percentage_label = f"{percentage:.1f}%"
```

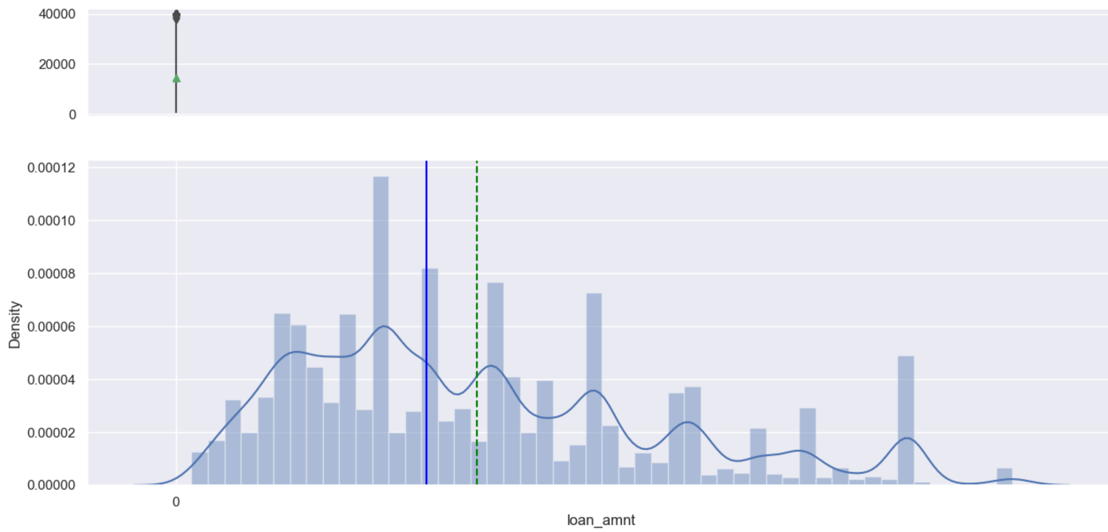
```
        x = p.get_x() + p.get_width() / 2 - 0.05 # width of the plot
        y = p.get_y() + p.get_height()           # hieght of the plot
        ax.annotate(percentage_label, (x, y), size = 12) # annotate the␣
↪percantage

    plt.show() # show the plot
```

[65]:
```
### Function to plot distributions and Boxplots of customers
def target_plot(x, target='loan_status'):
    '''
    plot
    feature: categorical feature
    the function won't work if a column is passed in hue parameter
    '''
    fig,axs = plt.subplots(2, 2, figsize=(12,10))
    axs[0, 0].set_title('Distribution of an loan_status')
    sns.distplot(data[(data[target] == 1)][x], ax=axs[0,0], color='teal')
    axs[0, 1].set_title('Distribution of an non-loan_status')
    sns.distplot(data[(data[target] == 0)][x], ax=axs[0,1], color='orange')

    axs[1,0].set_title('Boxplot w.r.t loan_status')
    sns.boxplot(data[target],data[x], ax=axs[1,0],palette='gist_rainbow')
    axs[1,1].set_title('Boxplot w.r.t non-loan_status - Without outliers')
    sns.
↪boxplot(data[target],data[x],ax=axs[1,1],showfliers=False,palette='gist_rainbow')
    plt.tight_layout()
    plt.show()
```

[66]:
```
df.select_dtypes(include='integer').columns
```

[66]:
```
Index(['fico_range_high', 'fico_range_low', 'loan_amnt', 'open_acc', 'pub_rec',
       'revol_bal', 'total_acc', 'loan_status'],
      dtype='object')
```

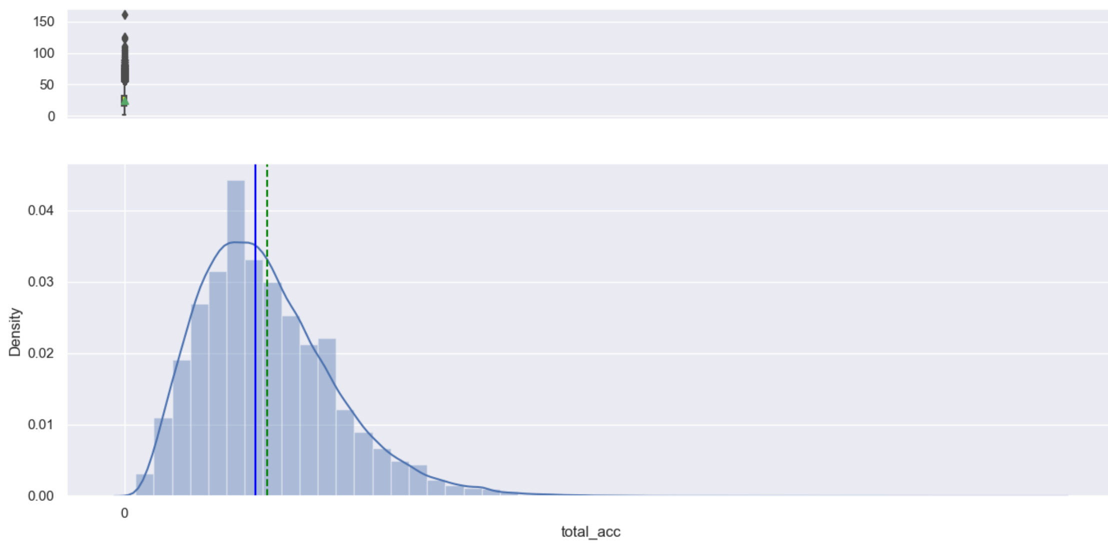
[67]:
```
histogram_boxplot(df.loan_amnt)
```




## 0.1 revolving balance

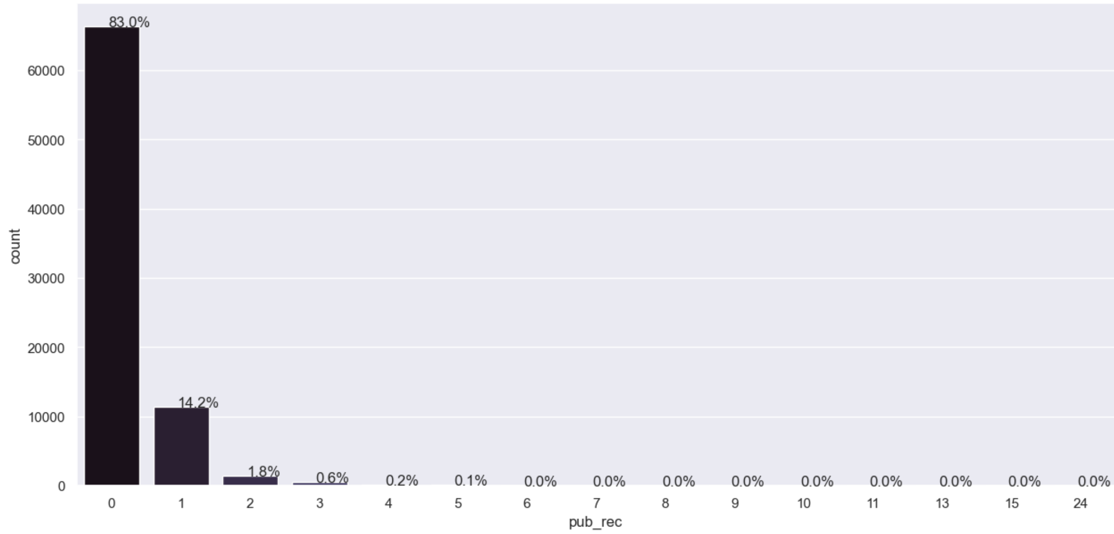
[68]:
```
histogram_boxplot(df.revol_bal)
```


[69]:
```
histogram_boxplot(df.total_acc)
```

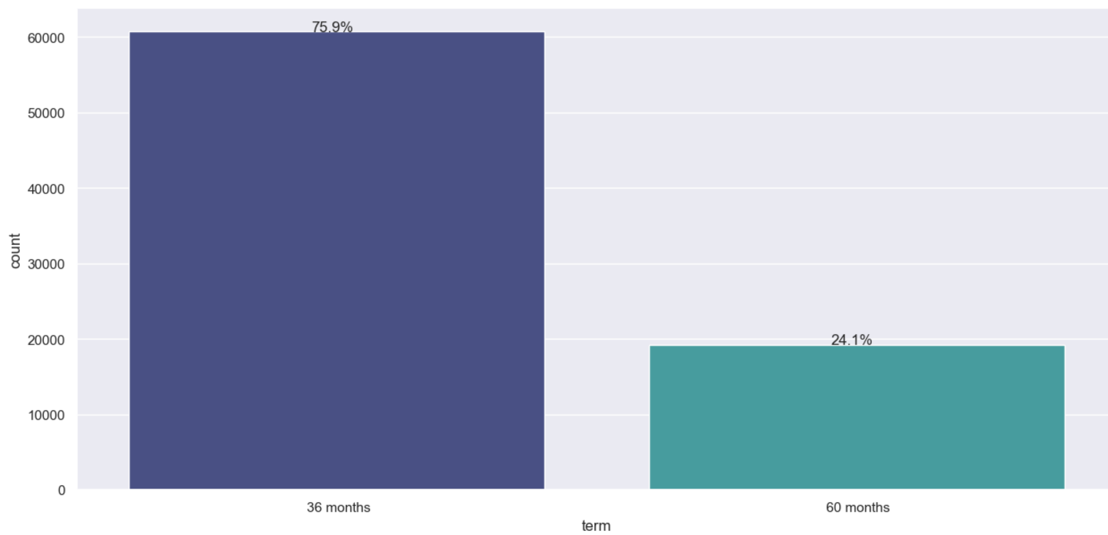



[70]:
```
histogram_boxplot(df.annual_inc)
```

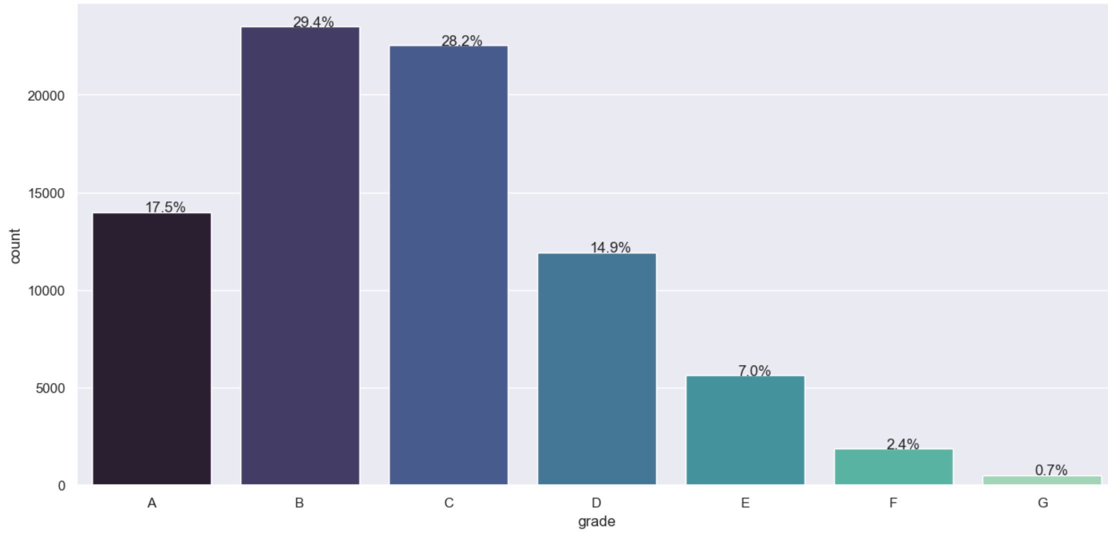

[71]:
```
perc_on_bar(df.pub_rec)
```



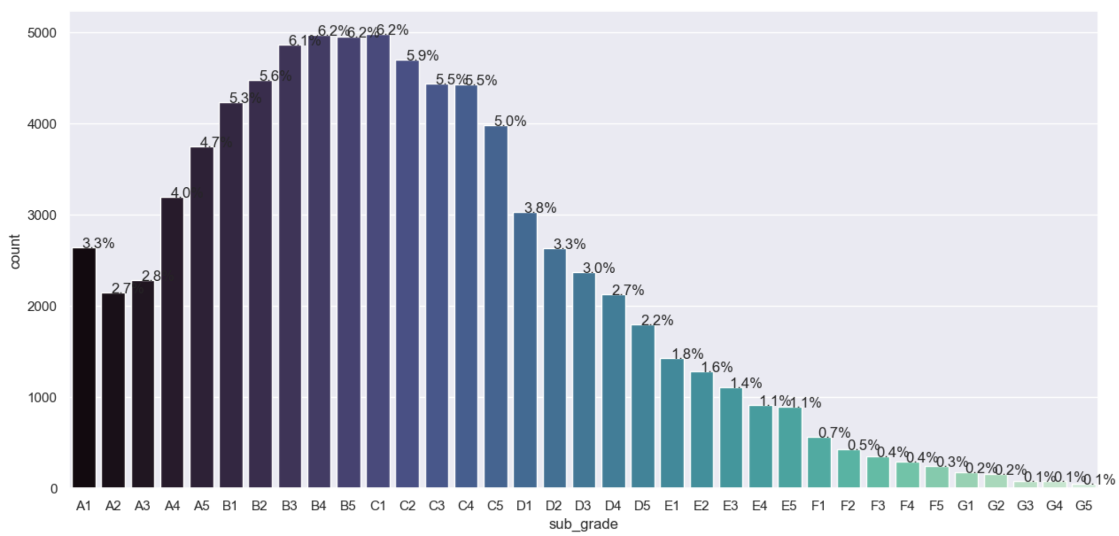

[72]:
```
perc_on_bar(df.term)
```



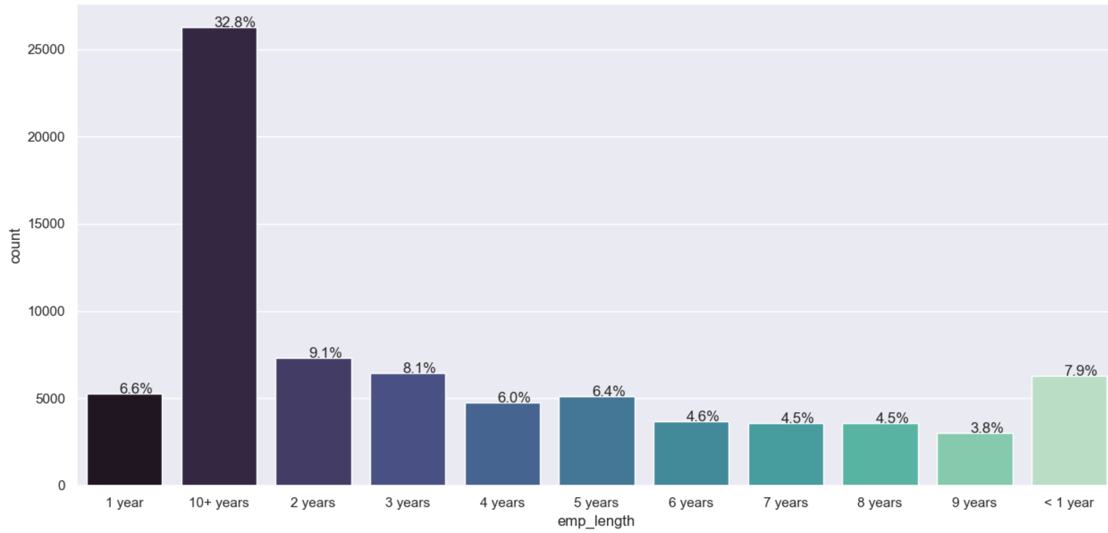

[73]:
```
perc_on_bar(df.grade)
```



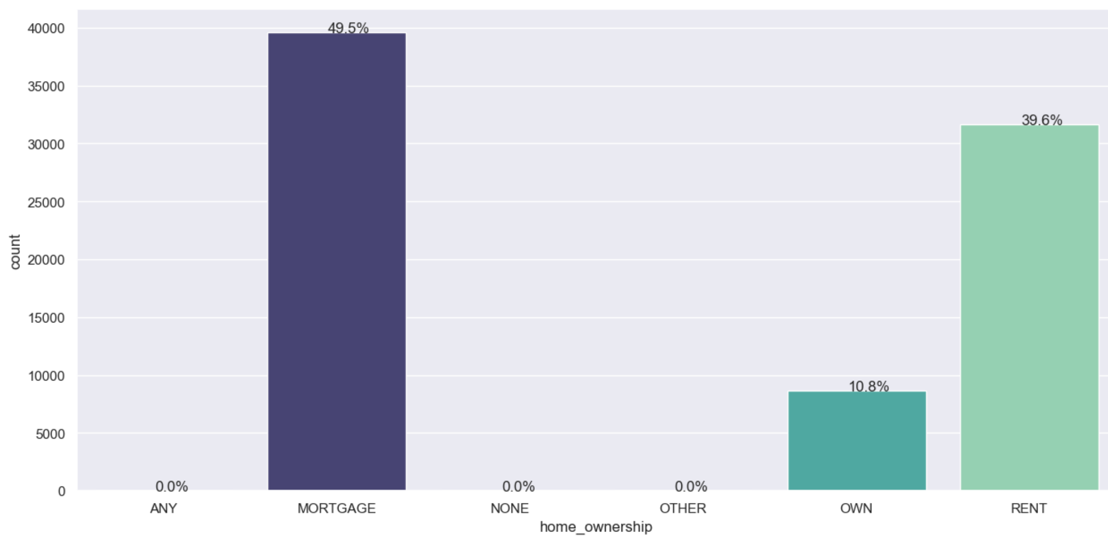

[74]:
```
perc_on_bar(df.sub_grade)
```



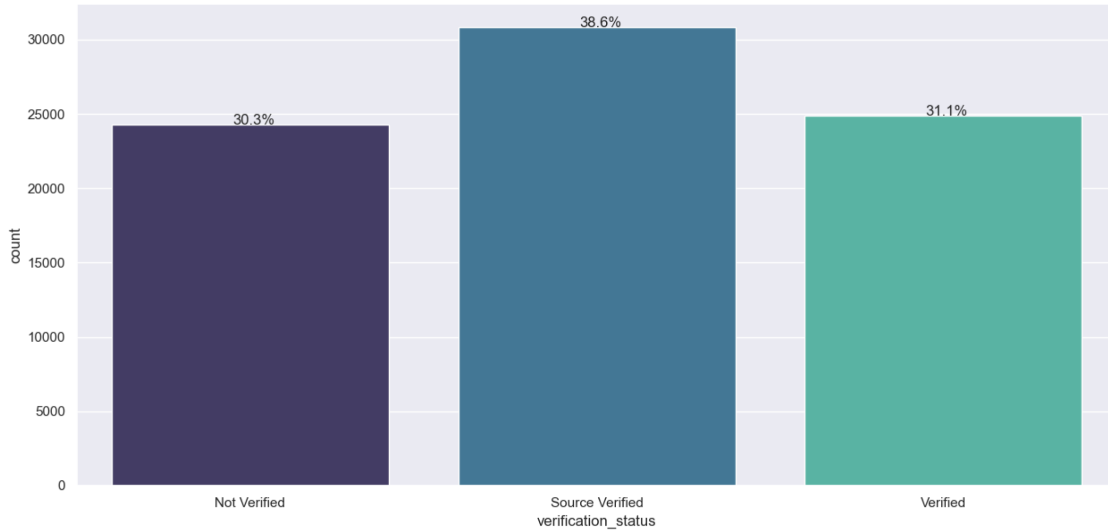

[75]:
```
# employee tenure

perc_on_bar(df.emp_length)
```



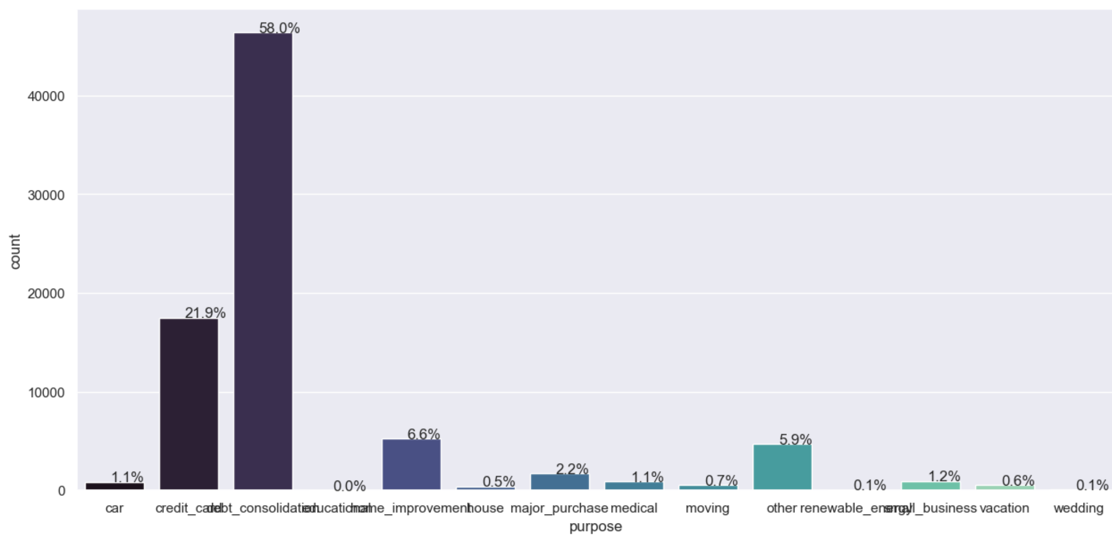

[76]: 
```
perc_on_bar(df.home_ownership)
```




[77]: 
```
perc_on_bar(df.verification_status)
```




[78]:
```
perc_on_bar(df.purpose)
```




# 1 Bivariate Analysis

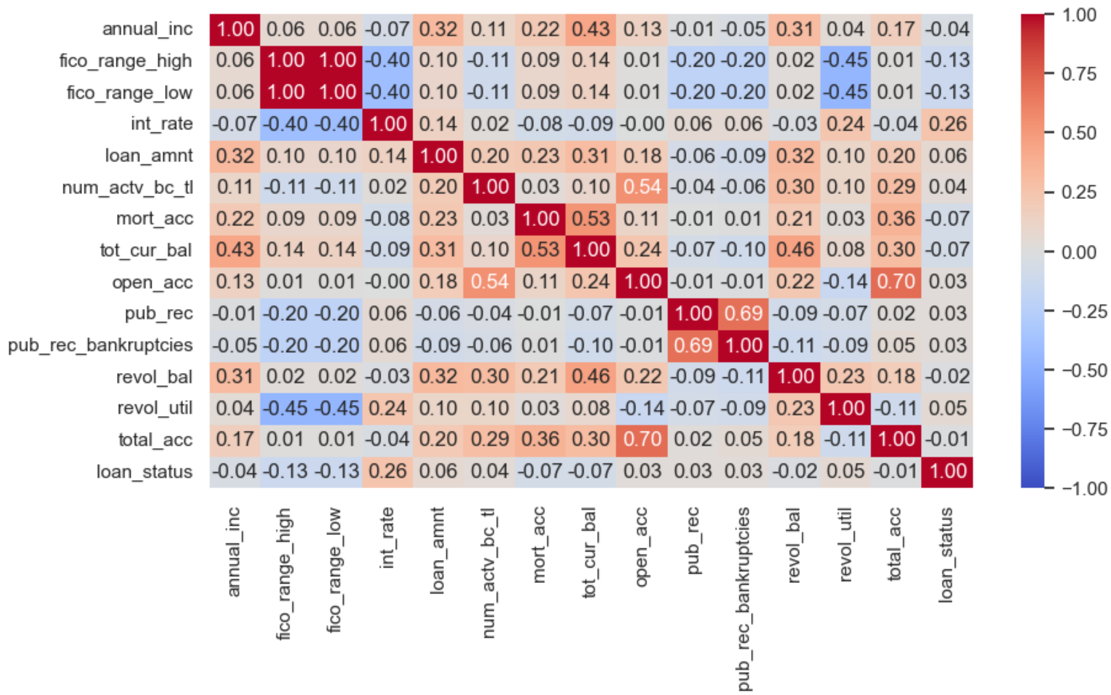
[79]:
```
## Function to plot stacked bar chart
def stacked_plot(x, y, show_df=True):
    """
    Shows stacked plot from x and y pandas data series
    x: pandas data series
    y: pandas data series
```

```
    show_df: flag to show dataframe above plot (loan_status=True)
    """
    if show_df == True:
        info = pd.crosstab(x, y, margins=True)
        info['% - 0'] = round(info[0]/info['All']*100, 2)
        info['% - 1'] = round(info[1]/info['All']*100, 2)
        display(info)

    pd.crosstab(x, y, normalize='index').plot(kind='bar', stacked=True,␣
 ↪figsize=(10,5));
```

[80]:
```
def show_boxplots(cols: list, feature: str, show_fliers=True, data=df): #method␣
 ↪call to show bloxplots
    """
    Shows boxplots from pandas data series
    cols: list of column names
    feature: dataframe categorical feature
    """
    n_rows = math.ceil(len(cols)/3)
    plt.figure(figsize=(15, n_rows*5))
    for i, variable in enumerate(cols):
        plt.subplot(n_rows, 3, i+1)
        if show_fliers:
            sns.boxplot(data[feature], data[variable], palette="mako",␣
 ↪showfliers=True)
        else:
            sns.boxplot(data[feature], data[variable], palette="mako",␣
 ↪showfliers=False)
        plt.tight_layout()
        plt.title(variable, fontsize=12)
    plt.show()
```

[81]:
```
### Function to plot distributions and Boxplots of customers
def plot_target(x, target='loan_status'):
    fig,axs = plt.subplots(2,2,figsize=(12,10))
    axs[0, 0].set_title('Distribution of loan_status')
    sns.distplot(data[(data[target] == 1)][x], ax=axs[0,0], color='teal')
    axs[0, 1].set_title('Distribution of NON-loan_status')
    sns.distplot(data[(data[target] == 0)][x],ax=axs[0,1], color='orange')
    axs[1,0].set_title('Boxplot w.r.t loan_status-flag')
    sns.boxplot(data[target],data[x],ax=axs[1,0], palette='mako')
    axs[1,1].set_title('Boxplot w.r.t loan_status-flag - Without outliers')
    sns.boxplot(data[target],data[x], ax=axs[1,1], showfliers=False,␣
 ↪palette='mako')
    plt.tight_layout()
    plt.show()
```

```
[82]: # Filter out non-numeric columns
numeric_df = df.select_dtypes(include=['float64', 'int64'])

# Plot correlation matrix
plt.figure(figsize=(10, 5))
sns.heatmap(numeric_df.corr(), annot=True, vmin=-1, vmax=1, fmt='.2f',
    cmap='coolwarm')
plt.show()
```

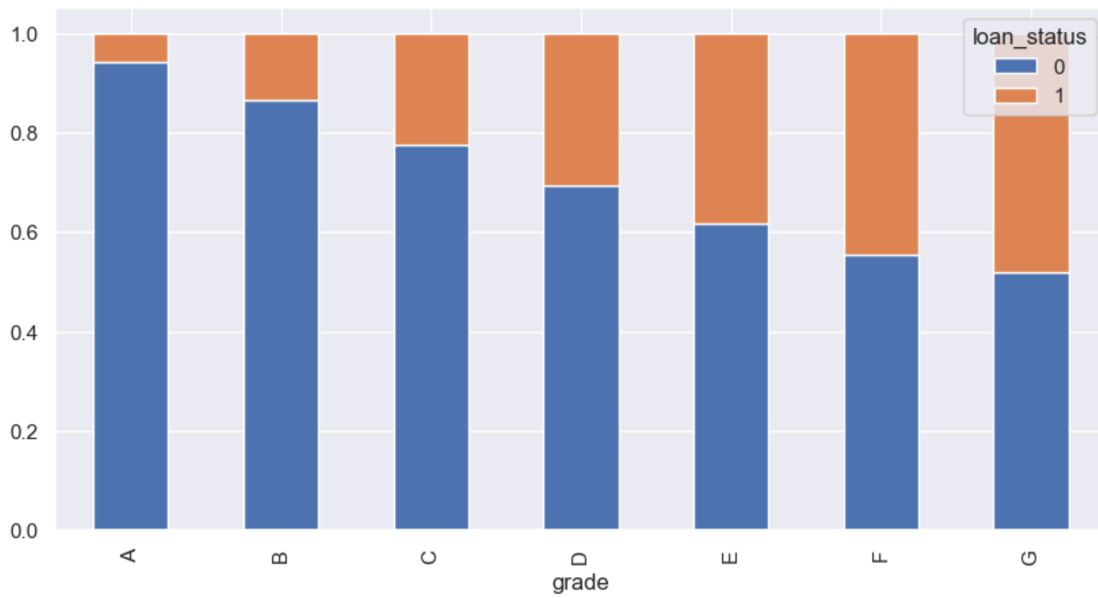



### 1.0.1 default vs grade

```
[ ]:
```

```
[83]: stacked_plot(df.grade, df.loan_status)
```

```
loan_status      0      1    All  % - 0  % - 1
grade
A            13177    819  13996 94.150  5.850
B            20328   3174  23502 86.490 13.510
C            17448   5077  22525 77.460 22.540
D             8288   3648  11936 69.440 30.560
E             3464   2156   5620 61.640 38.360
F             1046    839   1885 55.490 44.510
G              279    257    536 52.050 47.950
All          64030  15970  80000 80.040 19.960
```

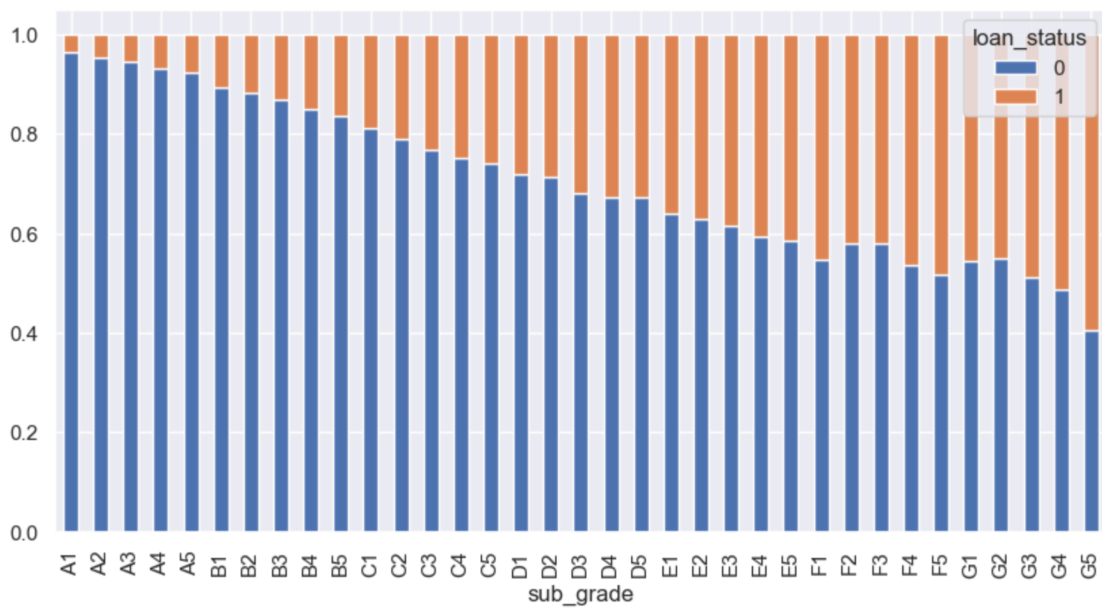



### 1.0.2 default vs loan_subgrade

[84]:
```
stacked_plot(df.sub_grade, df.loan_status)
```

| loan_status | 0 | 1 | All | % - 0 | % - 1 |
|---|---|---|---|---|---|
| sub_grade | | | | | |
| A1 | 2545 | 94 | 2639 | 96.440 | 3.560 |
| A2 | 2047 | 100 | 2147 | 95.340 | 4.660 |
| A3 | 2152 | 126 | 2278 | 94.470 | 5.530 |
| A4 | 2971 | 218 | 3189 | 93.160 | 6.840 |
| A5 | 3462 | 281 | 3743 | 92.490 | 7.510 |
| B1 | 3783 | 453 | 4236 | 89.310 | 10.690 |
| B2 | 3950 | 527 | 4477 | 88.230 | 11.770 |
| B3 | 4235 | 631 | 4866 | 87.030 | 12.970 |
| B4 | 4225 | 748 | 4973 | 84.960 | 15.040 |
| B5 | 4135 | 815 | 4950 | 83.540 | 16.460 |
| C1 | 4045 | 937 | 4982 | 81.190 | 18.810 |
| C2 | 3708 | 990 | 4698 | 78.930 | 21.070 |
| C3 | 3415 | 1025 | 4440 | 76.910 | 23.090 |
| C4 | 3328 | 1097 | 4425 | 75.210 | 24.790 |
| C5 | 2952 | 1028 | 3980 | 74.170 | 25.830 |
| D1 | 2171 | 853 | 3024 | 71.790 | 28.210 |
| D2 | 1872 | 754 | 2626 | 71.290 | 28.710 |
| D3 | 1610 | 754 | 2364 | 68.100 | 31.900 |
| D4 | 1430 | 698 | 2128 | 67.200 | 32.800 |
| D5 | 1205 | 589 | 1794 | 67.170 | 32.830 |
| E1 | 916 | 515 | 1431 | 64.010 | 35.990 |

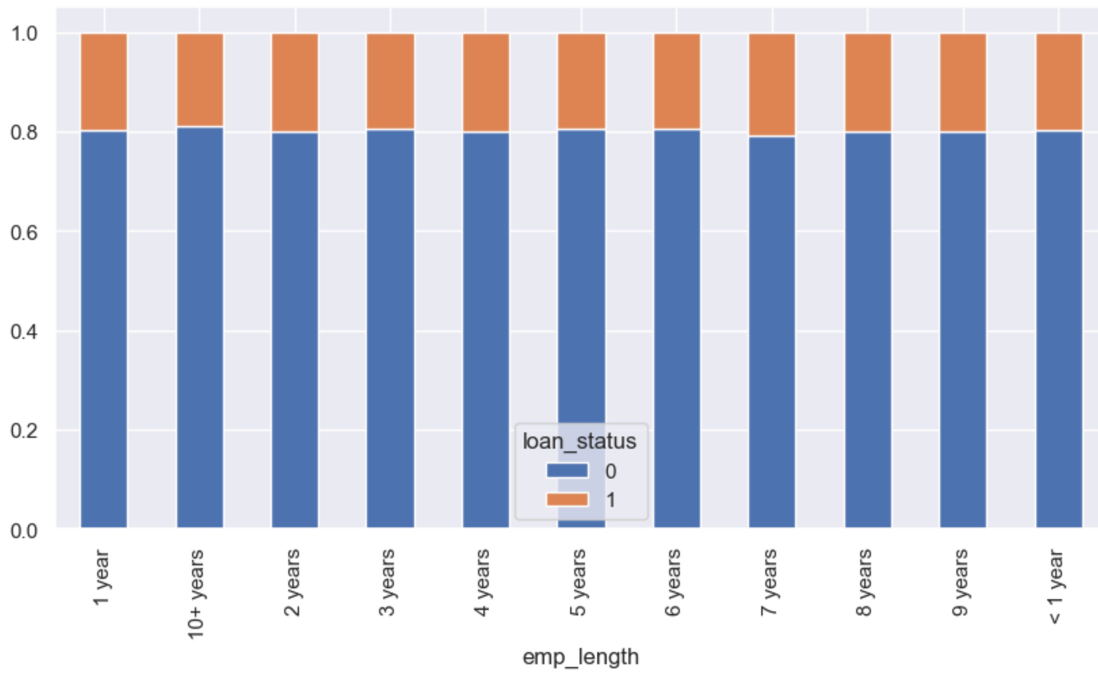
```
E2             803    475    1278 62.830 37.170
E3             681    426    1107 61.520 38.480
E4             542    369     911 59.500 40.500
E5             522    371     893 58.450 41.550
F1             309    257     566 54.590 45.410
F2             250    181     431 58.000 42.000
F3             205    149     354 57.910 42.090
F4             157    135     292 53.770 46.230
F5             125    117     242 51.650 48.350
G1              97     81     178 54.490 45.510
G2              83     68     151 54.970 45.030
G3              42     40      82 51.220 48.780
G4              38     40      78 48.720 51.280
G5              19     28      47 40.430 59.570
All          64030  15970   80000 80.040 19.960
```




[85]:
```
stacked_plot(df.emp_length, df.loan_status)
```

```
loan_status      0      1     All  % - 0  % - 1
emp_length
1 year        4244   1050    5294 80.170 19.830
10+ years    21315   4963   26278 81.110 18.890
2 years       5852   1467    7319 79.960 20.040
3 years       5212   1262    6474 80.510 19.490
4 years       3815    948    4763 80.100 19.900
5 years       4095    999    5094 80.390 19.610
6 years       2969    722    3691 80.440 19.560
```

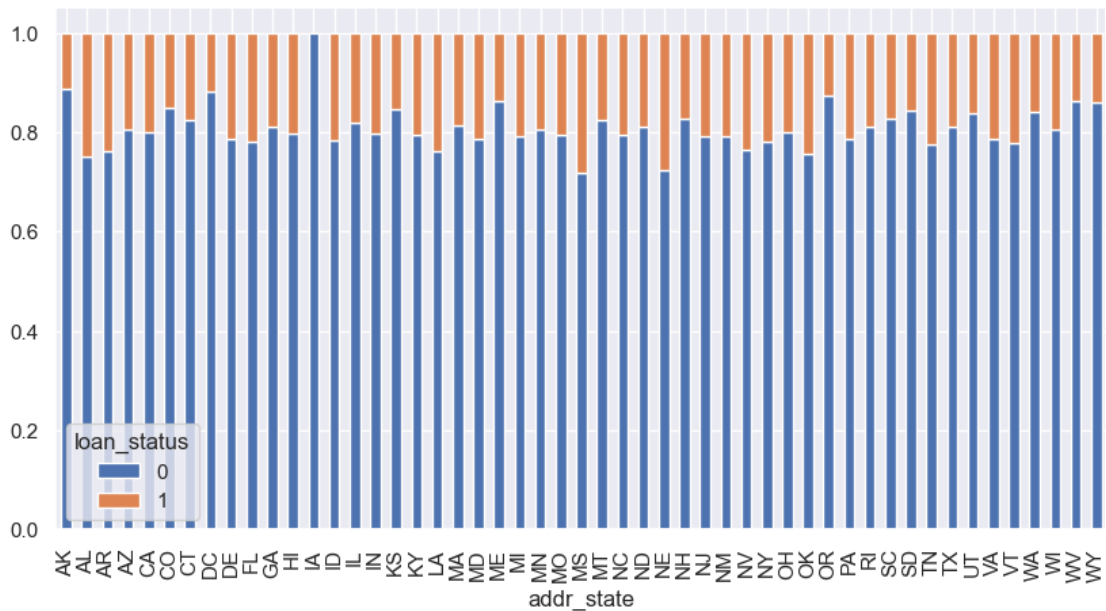
```
7 years        2849    748    3597 79.200 20.800
8 years        2868    715    3583 80.040 19.960
9 years        2416    606    3022 79.950 20.050
< 1 year       5046   1251    6297 80.130 19.870
All           60681  14731   75412 80.470 19.530
```




[86]:
```
stacked_plot(df.addr_state, df.loan_status)
```

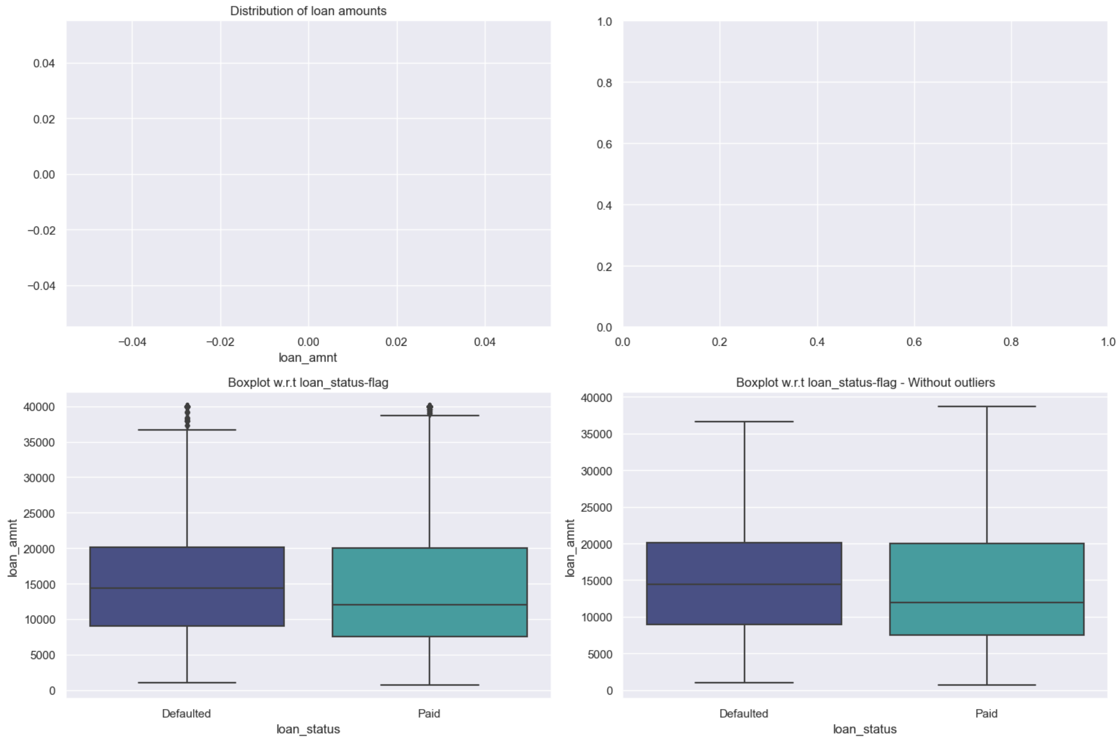
```
loan_status      0      1    All     % - 0  % - 1
addr_state
AK             191     24    215    88.840 11.160
AL             740    246    986    75.050 24.950
AR             450    140    590    76.270 23.730
AZ            1608    385   1993    80.680 19.320
CA            9409   2335  11744    80.120 19.880
CO            1520    270   1790    84.920 15.080
CT             942    201   1143    82.410 17.590
DC             177     24    201    88.060 11.940
DE             172     47    219    78.540 21.460
FL            4393   1225   5618    78.200 21.800
GA            2054    476   2530    81.190 18.810
HI             322     82    404    79.700 20.300
IA               1      0      1   100.000  0.000
ID              83     23    106    78.300 21.700
IL            2540    558   3098    81.990 18.010
IN            1059    270   1329    79.680 20.320
```

| | | | | | |
|---|---|---|---|---|---|
| KS | 550 | 99 | 649 | 84.750 | 15.250 |
| KY | 664 | 172 | 836 | 79.430 | 20.570 |
| LA | 707 | 221 | 928 | 76.190 | 23.810 |
| MA | 1517 | 345 | 1862 | 81.470 | 18.530 |
| MD | 1417 | 385 | 1802 | 78.630 | 21.370 |
| ME | 95 | 15 | 110 | 86.360 | 13.640 |
| MI | 1653 | 438 | 2091 | 79.050 | 20.950 |
| MN | 1140 | 274 | 1414 | 80.620 | 19.380 |
| MO | 1030 | 268 | 1298 | 79.350 | 20.650 |
| MS | 268 | 105 | 373 | 71.850 | 28.150 |
| MT | 189 | 40 | 229 | 82.530 | 17.470 |
| NC | 1823 | 468 | 2291 | 79.570 | 20.430 |
| ND | 69 | 16 | 85 | 81.180 | 18.820 |
| NE | 174 | 66 | 240 | 72.500 | 27.500 |
| NH | 309 | 64 | 373 | 82.840 | 17.160 |
| NJ | 2258 | 595 | 2853 | 79.140 | 20.860 |
| NM | 348 | 92 | 440 | 79.090 | 20.910 |
| NV | 936 | 288 | 1224 | 76.470 | 23.530 |
| NY | 5054 | 1407 | 6461 | 78.220 | 21.780 |
| OH | 2057 | 518 | 2575 | 79.880 | 20.120 |
| OK | 548 | 177 | 725 | 75.590 | 24.410 |
| OR | 896 | 129 | 1025 | 87.410 | 12.590 |
| PA | 2107 | 569 | 2676 | 78.740 | 21.260 |
| RI | 289 | 67 | 356 | 81.180 | 18.820 |
| SC | 833 | 174 | 1007 | 82.720 | 17.280 |
| SD | 162 | 30 | 192 | 84.380 | 15.620 |
| TN | 936 | 271 | 1207 | 77.550 | 22.450 |
| TX | 5272 | 1221 | 6493 | 81.200 | 18.800 |
| UT | 464 | 90 | 554 | 83.750 | 16.250 |
| VA | 1770 | 479 | 2249 | 78.700 | 21.300 |
| VT | 141 | 40 | 181 | 77.900 | 22.100 |
| WA | 1460 | 276 | 1736 | 84.100 | 15.900 |
| WI | 841 | 202 | 1043 | 80.630 | 19.370 |
| WV | 231 | 37 | 268 | 86.190 | 13.810 |
| WY | 161 | 26 | 187 | 86.100 | 13.900 |
| All | 64030 | 15970 | 80000 | 80.040 | 19.960 |



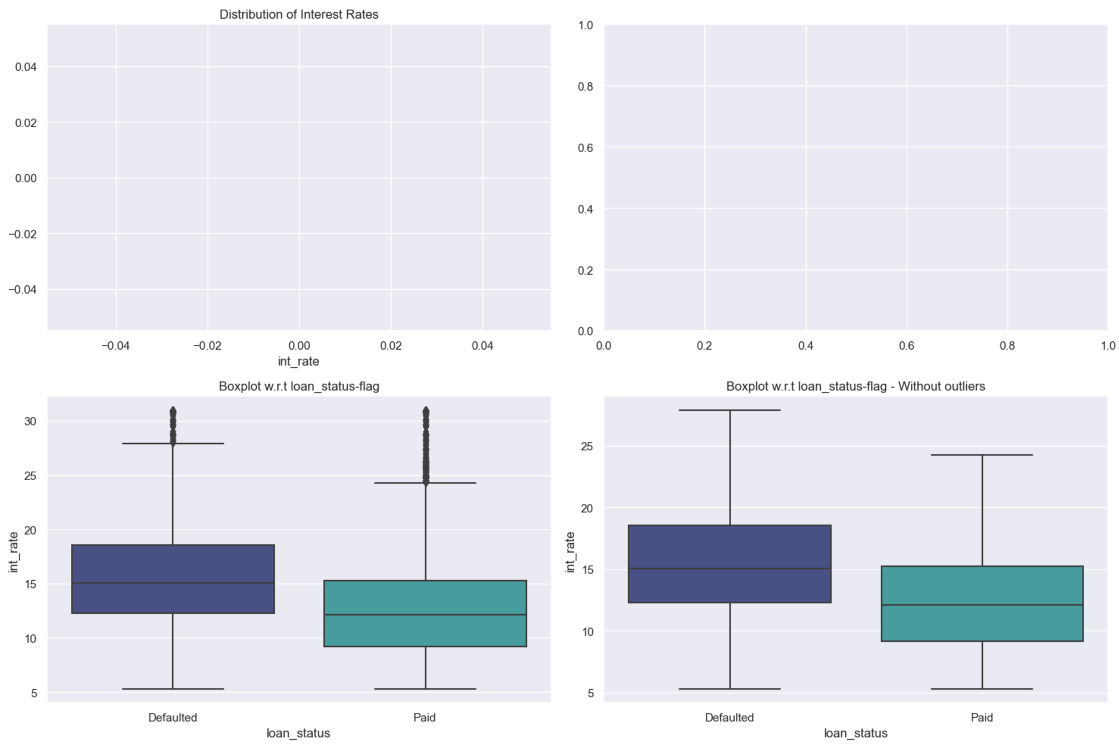

[87]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

def plot_target(x, target, data):
    fig, axs = plt.subplots(2, 2, figsize=(15, 10))

    sns.distplot(data[data[target] == 1][x], ax=axs[0, 0], color='blue', label='Paid')
    sns.distplot(data[data[target] == 0][x], ax=axs[0, 0], color='orange', label='Defaulted')
    axs[0, 0].set_title('Distribution of loan amounts')

    sns.boxplot(x=data[target], y=data[x], ax=axs[1, 0], palette='mako')
    axs[1, 0].set_title('Boxplot w.r.t loan_status-flag')

    sns.boxplot(x=data[target], y=data[x], ax=axs[1, 1], showfliers=False, palette='mako')
    axs[1, 1].set_title('Boxplot w.r.t loan_status-flag - Without outliers')

    plt.tight_layout()
    plt.show()

# Example usage:
# 'loan_amnt' is the feature for which you want to plot boxplots
# 'loan_status' is the target variable
# 'data' is your DataFrame
```

```
plot_target(x='loan_amnt', target='loan_status', data=data)
```




[88]:
```
import seaborn as sns
import matplotlib.pyplot as plt

def plot_int_rate(target, data):
    fig, axs = plt.subplots(2, 2, figsize=(15, 10))

    sns.distplot(data[data[target] == 1]['int_rate'], ax=axs[0, 0],␣
 ↪color='blue', label='Paid')
    sns.distplot(data[data[target] == 0]['int_rate'], ax=axs[0, 0],␣
 ↪color='orange', label='Defaulted')
    axs[0, 0].set_title('Distribution of Interest Rates')

    sns.boxplot(x=data[target], y=data['int_rate'], ax=axs[1, 0],␣
 ↪palette='mako')
    axs[1, 0].set_title('Boxplot w.r.t loan_status-flag')

    sns.boxplot(x=data[target], y=data['int_rate'], ax=axs[1, 1],␣
 ↪showfliers=False, palette='mako')
    axs[1, 1].set_title('Boxplot w.r.t loan_status-flag - Without outliers')
```

```
    plt.tight_layout()
    plt.show()

# Example usage:
# 'loan_status' is the target variable
# 'data' is your DataFrame

plot_int_rate(target='loan_status', data=data)
```




[89]:
```
df.isnull().sum()
```

[89]:
```
addr_state                0
annual_inc                0
earliest_cr_line          0
emp_length             4588
emp_title              5018
fico_range_high           0
fico_range_low            0
grade                     0
home_ownership            0
application_type          0
initial_list_status       0
int_rate                  0
```

```
loan_amnt                  0
num_actv_bc_tl          3948
mort_acc                2771
tot_cur_bal             3948
open_acc                   0
pub_rec                    0
pub_rec_bankruptcies      31
purpose                    0
revol_bal                  0
revol_util                53
sub_grade                  0
term                       0
title                    970
total_acc                  0
verification_status        0
loan_status                0
dtype: int64
```

[90]:

```
for i in df.select_dtypes(include=['category']).columns:
    print('Unique values in', i, 'are :')
    print(df[i].value_counts(dropna=False))
    print('*'*50)
```

```
Unique values in addr_state are :
addr_state
CA    11744
TX     6493
NY     6461
FL     5618
IL     3098
NJ     2853
PA     2676
OH     2575
GA     2530
NC     2291
VA     2249
MI     2091
AZ     1993
MA     1862
MD     1802
CO     1790
WA     1736
MN     1414
IN     1329
MO     1298
NV     1224
TN     1207
CT     1143
```

```
WI     1043
OR     1025
SC     1007
AL      986
LA      928
KY      836
OK      725
KS      649
AR      590
UT      554
NM      440
HI      404
MS      373
NH      373
RI      356
WV      268
NE      240
MT      229
DE      219
AK      215
DC      201
SD      192
WY      187
VT      181
ME      110
ID      106
ND       85
IA        1
Name: count, dtype: int64
**************************************************
Unique values in earliest_cr_line are :
earliest_cr_line
Sep-03    547
Aug-03    545
Aug-01    544
Oct-01    541
Sep-02    539
         …
Jul-65      1
Sep-59      1
Sep-65      1
Jul-64      1
Nov-66      1
Name: count, Length: 640, dtype: int64
**************************************************
Unique values in emp_length are :
emp_length
10+ years    26278
```

```
2 years       7319
3 years       6474
< 1 year      6297
1 year        5294
5 years       5094
4 years       4763
NaN           4588
6 years       3691
7 years       3597
8 years       3583
9 years       3022
Name: count, dtype: int64
**************************************************
Unique values in emp_title are :
emp_title
NaN                                 5018
Teacher                             1278
Manager                             1194
Owner                                592
RN                                   526
                                    …
Hotel Desk Coordinator                 1
Hotel & Travel Credit Union            1
Hot oiler                              1
Hostler                                1
MyBuys                                 1
Name: count, Length: 36662, dtype: int64
**************************************************
Unique values in grade are :
grade
B    23502
C    22525
A    13996
D    11936
E     5620
F     1885
G      536
Name: count, dtype: int64
**************************************************
Unique values in home_ownership are :
home_ownership
MORTGAGE    39628
RENT        31688
OWN          8654
ANY            19
OTHER           7
NONE            4
Name: count, dtype: int64
```

```
**************************************************
Unique values in application_type are :
application_type
Individual    78446
Joint App      1554
Name: count, dtype: int64
**************************************************
Unique values in initial_list_status are :
initial_list_status
w    46745
f    33255
Name: count, dtype: int64
**************************************************
Unique values in purpose are :
purpose
debt_consolidation    46418
credit_card           17506
home_improvement       5268
other                  4683
major_purchase         1746
small_business          950
medical                 902
car                     868
moving                  548
vacation                518
house                   413
wedding                 110
renewable_energy         54
educational              16
Name: count, dtype: int64
**************************************************
Unique values in sub_grade are :
sub_grade
C1    4982
B4    4973
B5    4950
B3    4866
C2    4698
B2    4477
C3    4440
C4    4425
B1    4236
C5    3980
A5    3743
A4    3189
D1    3024
A1    2639
D2    2626
```

```
D3    2364
A3    2278
A2    2147
D4    2128
D5    1794
E1    1431
E2    1278
E3    1107
E4     911
E5     893
F1     566
F2     431
F3     354
F4     292
F5     242
G1     178
G2     151
G3      82
G4      78
G5      47
Name: count, dtype: int64
**************************************************
Unique values in term are :
term
 36 months    60750
 60 months    19250
Name: count, dtype: int64
**************************************************
Unique values in title are :
title
Debt consolidation                           39396
Credit card refinancing                      14802
Home improvement                              4542
Other                                         4035
Major purchase                                1422
                                           …
Get on the right track                           1
Get me out of debt with lower interest!          1
Get it right                                     1
Get it done                                      1
Mama to Be                                       1
Name: count, Length: 5349, dtype: int64
**************************************************
Unique values in verification_status are :
verification_status
Source Verified    30855
Verified           24876
Not Verified       24269
```

```
Name: count, dtype: int64
**************************************************
```

[91]:
```python
df1 = df.copy()
```

[92]:
```python
df1.head()
```

[92]:
```
  addr_state  annual_inc earliest_cr_line emp_length  \
0         CO   85000.000           Jul-97  10+ years
1         CA   40000.000           Apr-87  10+ years
2         FL   60000.000           Aug-07  10+ years
3         IL  100742.000           Sep-80  10+ years
4         MD   80000.000           Jul-99  10+ years

                        emp_title  fico_range_high  fico_range_low grade  \
0                          Deputy              744             740     E
1  Department of Veterans Affairs              724             720     B
2                Marble polishing              679             675     B
3                         printer              664             660     B
4                   Southern Mgmt              669             665     F

  home_ownership application_type  … pub_rec_bankruptcies  \
0       MORTGAGE       Individual  …                0.000
1           RENT       Individual  …                0.000
2       MORTGAGE       Individual  …                0.000
3       MORTGAGE       Individual  …                0.000
4           RENT       Individual  …                0.000

              purpose  revol_bal  revol_util  sub_grade        term  \
0  debt_consolidation       5338      93.600         E1   60 months
1  debt_consolidation      19944      60.300         B1   36 months
2  debt_consolidation      23199      88.500         B5   36 months
3  debt_consolidation      18425      69.000         B2   36 months
4  debt_consolidation      34370      90.000         F5   60 months

                 title  total_acc  verification_status loan_status
0   Debt consolidation          8      Source Verified           1
1          Credit Loan         12             Verified           0
2   Debt consolidation         16      Source Verified           0
3   Debt consolidation         19      Source Verified           0
4  Debt Connsolidation         59             Verified           0

[5 rows x 28 columns]
```

[93]:
```python
df1.home_ownership.replace('NONE','OTHER', inplace=True)
df1.home_ownership.value_counts().sort_values(ascending=False)
```

[93]:
```
home_ownership
MORTGAGE    39628
RENT        31688
OWN          8654
ANY            19
OTHER          11
Name: count, dtype: int64
```

[94]:
```
df1.verification_status.replace('Source Verified','Verified', inplace=True)
df1.verification_status.value_counts().sort_values(ascending=False)
```

[94]:
```
verification_status
Verified        55731
Not Verified    24269
Name: count, dtype: int64
```

[95]:
```
df1.head(20)
```

[95]:
```
   addr_state  annual_inc earliest_cr_line emp_length  \
0          CO   85000.000           Jul-97  10+ years
1          CA   40000.000           Apr-87  10+ years
2          FL   60000.000           Aug-07  10+ years
3          IL  100742.000           Sep-80  10+ years
4          MD   80000.000           Jul-99  10+ years
5          CA   51488.000           May-91        NaN
6          NY  100000.000           Oct-86  10+ years
7          PA   35028.000           Nov-95    3 years
8          FL   59292.000           Dec-07        NaN
9          CA   65000.000           Jun-04   < 1 year
10         WI   35000.000           Jul-99     1 year
11         UT   30000.000           Aug-96    8 years
12         NY  100000.000           Oct-98    7 years
13         CA   80000.000           May-07    4 years
14         CA   73000.000           Oct-00     1 year
15         TX   48500.000           Jan-05    8 years
16         AL   52512.000           Apr-04        NaN
17         KS   83840.000           Sep-00    2 years
18         AR  100000.000           Sep-93        NaN
19         CA  852000.000           Oct-01    3 years

                           emp_title  fico_range_high  fico_range_low grade  \
0                             Deputy              744             740     E
1   Department of Veterans Affairs              724             720     B
2                    Marble polishing              679             675     B
3                             printer              664             660     B
4                      Southern Mgmt              669             665     F
5                                NaN              679             675     D
```

```
6                        RN           699           695     C
7                      SHHC           679           675     C
8                       NaN           664           660     B
9                     Nurse           684           680     D
10                Carpenter           679           675     B
11           Office manager           749           745     B
12           Vice President           694           690     B
13          Executive chef            744           740     A
14                    Graye           674           670     E
15                  Manager           679           675     C
16                      NaN           684           680     C
17                  Manager           729           725     C
18                      NaN           744           740     B
19      Logistics Corrdinator           689           685     D

   home_ownership application_type  … pub_rec_bankruptcies  \
0        MORTGAGE       Individual  …                0.000
1            RENT       Individual  …                0.000
2        MORTGAGE       Individual  …                0.000
3        MORTGAGE       Individual  …                0.000
4            RENT       Individual  …                0.000
5        MORTGAGE       Individual  …                0.000
6        MORTGAGE       Individual  …                0.000
7            RENT       Individual  …                0.000
8        MORTGAGE       Individual  …                0.000
9            RENT       Individual  …                0.000
10       MORTGAGE       Individual  …                0.000
11            OWN       Individual  …                0.000
12           RENT       Individual  …                0.000
13           RENT       Individual  …                0.000
14           RENT       Individual  …                0.000
15            OWN       Individual  …                0.000
16       MORTGAGE       Individual  …                1.000
17           RENT       Individual  …                0.000
18       MORTGAGE       Individual  …                0.000
19       MORTGAGE       Individual  …                0.000

              purpose  revol_bal  revol_util  sub_grade        term  \
0   debt_consolidation       5338      93.600         E1   60 months
1   debt_consolidation      19944      60.300         B1   36 months
2   debt_consolidation      23199      88.500         B5   36 months
3   debt_consolidation      18425      69.000         B2   36 months
4   debt_consolidation      34370      90.000         F5   60 months
5     home_improvement      10747      53.900         D3   36 months
6          credit_card      32488      54.100         C1   36 months
7   debt_consolidation      13147      78.300         C4   36 months
8   debt_consolidation       1054      23.400         B4   36 months
```

```
9   debt_consolidation       8991      64.700         D4    36 months
10  debt_consolidation      23293      71.700         B4    36 months
11  debt_consolidation       8355      23.000         B5    36 months
12  debt_consolidation      16112      49.300         B5    36 months
13               other      12405      44.900         A5    36 months
14         credit_card      15343      84.800         E1    36 months
15  debt_consolidation       8236      30.800         C3    36 months
16               other       3575      16.800         C3    36 months
17  debt_consolidation      21963      40.100         C2    60 months
18    home_improvement       8879      48.000         B1    36 months
19  debt_consolidation      16245      67.400         D4    60 months

                        title  total_acc  verification_status loan_status
0          Debt consolidation          8             Verified           1
1                 Credit Loan         12             Verified           0
2          Debt consolidation         16             Verified           0
3          Debt consolidation         19             Verified           0
4         Debt Connsolidation         59             Verified           0
5            Home improvement         37             Verified           0
6       Credit card refinancing       36             Verified           0
7   Credit consolidation sought       19         Not Verified           0
8                         NaN         23             Verified           0
9          Debt consolidation         20             Verified           0
10         Debt consolidation         24             Verified           0
11         Debt consolidation         19         Not Verified           0
12         Debt consolidation         15             Verified           0
13                      Other          8         Not Verified           0
14            card consolidate          7         Not Verified           0
15         Debt consolidation         21             Verified           1
16                      Other         36             Verified           0
17         Debt consolidation         45             Verified           1
18           Home improvement         21             Verified           1
19         Debt consolidation         24             Verified           1

[20 rows x 28 columns]
```

[96]:

```python
def region_combining(state):
    midwest = ['IA', 'IL', 'IN', 'KS', 'MI', 'MN', 'MO', 'ND', 'NE', 'OH',␣
↪'SD', 'WI']
    northeast = ['CT', 'MA', 'ME', 'NH', 'NJ', 'NY', 'PA', 'RI', 'VT']
    south = ['AL', 'AR', 'DC', 'DE', 'FL', 'GA', 'KY', 'LA', 'MD', 'MS', 'NC',␣
↪'OK',
             'SC', 'TN', 'TX', 'VA', 'WV']
    west = ['AK', 'AZ', 'CA', 'CO', 'HI', 'ID', 'MT', 'NM', 'NV', 'OR', 'UT',␣
↪'WA', 'WY']

    if state in midwest:
```

```
        return 'Midwest'
    elif state in northeast:
        return 'Northeast'
    elif state in south:
        return 'South'
    elif state in west:
        return 'West'
    else:
        return 'Other'
```

[97]:

```
import pandas as pd


# Apply region_combining function to 'addr_state' column
df1['addr_state'] = df1['addr_state'].apply(region_combining)

# Convert 'addr_state' to categorical type
df1['addr_state'] = df1['addr_state'].astype('category')

# Check value counts
print(df1['addr_state'].value_counts(dropna=False))
```

```
addr_state
South        28323
West         21647
Northeast    16015
Midwest      14015
Name: count, dtype: int64
```

[98]:

```
df1.annual_inc.fillna(df.annual_inc.mean(), inplace=True)
df1.tot_cur_bal.fillna(df.tot_cur_bal.mean(), inplace=True)
df1.revol_util.fillna(df.revol_util.mean(), inplace=True)
df1.total_acc.fillna(df.total_acc.mean(), inplace=True)
```

[ ]:

[ ]:

[104]:

```
df1.head()
```

[104]:

```
  addr_state  annual_inc earliest_cr_line  emp_length  \
0       West   85000.000           Jul-97           2
1       West   40000.000           Apr-87           2
2      South   60000.000           Aug-07           2
3    Midwest  100742.000           Sep-80           2
4      South   80000.000           Jul-99           2

                       emp_title  fico_range_high  fico_range_low  grade  \
```

```
0                          Deputy            744              740       4
1  Department of Veterans Affairs            724              720       1
2                 Marble polishing           679              675       1
3                          printer           664              660       1
4                    Southern Mgmt           669              665       5

  home_ownership application_type  … pub_rec_bankruptcies  \
0       MORTGAGE       Individual  …                0.000
1           RENT       Individual  …                0.000
2       MORTGAGE       Individual  …                0.000
3       MORTGAGE       Individual  …                0.000
4           RENT       Individual  …                0.000

              purpose  revol_bal  revol_util  sub_grade  term  \
0  debt_consolidation       5338      93.600         E1  <NA>
1  debt_consolidation      19944      60.300         B1  <NA>
2  debt_consolidation      23199      88.500         B5  <NA>
3  debt_consolidation      18425      69.000         B2  <NA>
4  debt_consolidation      34370      90.000         F5  <NA>

                 title  total_acc  verification_status loan_status
0   Debt consolidation          8             Verified           1
1          Credit Loan         12             Verified           0
2   Debt consolidation         16             Verified           0
3   Debt consolidation         19             Verified           0
4  Debt Connsolidation         59             Verified           0

[5 rows x 28 columns]
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[108]:
```
df5 = df.copy()
```

[109]:
```
addr_state = {'AK':0,  'AL':1,  'AR':2,  'AZ':3,  'CA':4 , 'CO':5,  'CT':6, ␣
↪'DC':7,  'DE':8,
              'FL':9,  'GA':10, 'HI':11, 'IA':12, 'ID':13, 'IL':14, 'IN':15,␣
↪'KS':16, 'KY':17,
              'LA':18, 'MA':19, 'MD':20, 'ME':21, 'MI':22, 'MN':23, 'MO':24,␣
↪'MS':25, 'MT':26,
```

```
              'NC':27, 'ND':28, 'NE':29, 'NH':30, 'NJ':31, 'NM':32, 'NV':33,␣
↪'NY':34, 'OH':35,
              'OK':36, 'OR':37, 'PA':38, 'RI':39, 'SC':40, 'SD':41, 'TN':42,␣
↪'TX':43, 'UT':44,
              'VA':45, 'VT':46, 'WA':47, 'WI':48, 'WV':49, 'WY':50}
df5['addr_state'] = df5['addr_state'].map(addr_state).astype('Int32')
```

[110]:
```
print(df5.addr_state)
```

```
0         5
1         4
2         9
3        14
4        20
         ..
79995    40
79996    30
79997    34
79998    43
79999    34
Name: addr_state, Length: 80000, dtype: Int32
```

[111]:
```
df5.head()
```

[111]:
```
   addr_state  annual_inc earliest_cr_line emp_length  \
0           5   85000.000           Jul-97  10+ years
1           4   40000.000           Apr-87  10+ years
2           9   60000.000           Aug-07  10+ years
3          14  100742.000           Sep-80  10+ years
4          20   80000.000           Jul-99  10+ years

                        emp_title  fico_range_high  fico_range_low grade  \
0                          Deputy              744             740     E
1  Department of Veterans Affairs              724             720     B
2                 Marble polishing              679             675     B
3                          printer              664             660     B
4                    Southern Mgmt              669             665     F

  home_ownership application_type  … pub_rec_bankruptcies  \
0       MORTGAGE       Individual  …                0.000
1           RENT       Individual  …                0.000
2       MORTGAGE       Individual  …                0.000
3       MORTGAGE       Individual  …                0.000
4           RENT       Individual  …                0.000

             purpose  revol_bal  revol_util  sub_grade        term  \
0  debt_consolidation       5338      93.600         E1   60 months
```

```
1  debt_consolidation      19944      60.300          B1    36 months
2  debt_consolidation      23199      88.500          B5    36 months
3  debt_consolidation      18425      69.000          B2    36 months
4  debt_consolidation      34370      90.000          F5    60 months

                 title  total_acc  verification_status loan_status
0   Debt consolidation          8      Source Verified           1
1          Credit Loan         12             Verified           0
2   Debt consolidation         16      Source Verified           0
3   Debt consolidation         19      Source Verified           0
4  Debt Connsolidation         59             Verified           0

[5 rows x 28 columns]
```

[112]:
```
home_ownership = {'MORTGAGE':0, 'RENT':1, 'OWN':2, 'OTHER':3, 'NONE':4}
df5['home_ownership'] = df5['home_ownership'].map(home_ownership).
  ↪astype('Int32')
```

[113]:
```
print(df5.home_ownership)
```

```
0        0
1        1
2        0
3        0
4        1
        ..
79995    0
79996    0
79997    2
79998    0
79999    0
Name: home_ownership, Length: 80000, dtype: Int32
```

[114]:
```
df5.head()
```

[114]:
```
   addr_state  annual_inc earliest_cr_line emp_length  \
0           5   85000.000           Jul-97  10+ years
1           4   40000.000           Apr-87  10+ years
2           9   60000.000           Aug-07  10+ years
3          14  100742.000           Sep-80  10+ years
4          20   80000.000           Jul-99  10+ years

                        emp_title  fico_range_high  fico_range_low grade  \
0                          Deputy              744             740     E
1  Department of Veterans Affairs              724             720     B
2                 Marble polishing              679             675     B
3                          printer              664             660     B
4                    Southern Mgmt              669             665     F
```

```
  home_ownership application_type  … pub_rec_bankruptcies  \
0              0       Individual  …                0.000
1              1       Individual  …                0.000
2              0       Individual  …                0.000
3              0       Individual  …                0.000
4              1       Individual  …                0.000

             purpose  revol_bal  revol_util  sub_grade       term  \
0  debt_consolidation       5338      93.600         E1  60 months
1  debt_consolidation      19944      60.300         B1  36 months
2  debt_consolidation      23199      88.500         B5  36 months
3  debt_consolidation      18425      69.000         B2  36 months
4  debt_consolidation      34370      90.000         F5  60 months

                title  total_acc  verification_status loan_status
0  Debt consolidation          8      Source Verified           1
1         Credit Loan         12             Verified           0
2  Debt consolidation         16      Source Verified           0
3  Debt consolidation         19      Source Verified           0
4 Debt Connsolidation         59             Verified           0

[5 rows x 28 columns]
```

[ ]:

[ ]:

[115]:

```python
# Define the mapping dictionary
emp_length_mapping = {'< 1 year': 0, '1 year': 0, '2 years': 0, '3 years': 0,␣
↪'4 years': 0,
                      '5 years': 0, '6 years': 1, '7 years': 1, '8 years': 1,␣
↪'9 years': 1,
                      '10+ years': 2, 'NaN': -1}  # Use -1 to represent unknown␣
↪or missing values

# Map the values in the DataFrame
df5['emp_length'] = df5['emp_length'].map(emp_length_mapping).astype('Int32')
```

[116]:

```python
print(df5.emp_length)
```

```
0        2
1        2
2        2
3        2
4        2
        ..
79995    2
```

```
79996    2
79997    0
79998    0
79999    0
Name: emp_length, Length: 80000, dtype: Int32
```

[117]:
```python
df5.head()
```

[117]:
```
   addr_state  annual_inc earliest_cr_line  emp_length  \
0           5   85000.000           Jul-97           2
1           4   40000.000           Apr-87           2
2           9   60000.000           Aug-07           2
3          14  100742.000           Sep-80           2
4          20   80000.000           Jul-99           2

                     emp_title  fico_range_high  fico_range_low grade  \
0                       Deputy              744             740     E
1  Department of Veterans Affairs           724             720     B
2             Marble polishing              679             675     B
3                      printer              664             660     B
4                Southern Mgmt              669             665     F

   home_ownership application_type  … pub_rec_bankruptcies  \
0               0       Individual  …                0.000
1               1       Individual  …                0.000
2               0       Individual  …                0.000
3               0       Individual  …                0.000
4               1       Individual  …                0.000

              purpose  revol_bal  revol_util  sub_grade       term  \
0  debt_consolidation       5338      93.600         E1  60 months
1  debt_consolidation      19944      60.300         B1  36 months
2  debt_consolidation      23199      88.500         B5  36 months
3  debt_consolidation      18425      69.000         B2  36 months
4  debt_consolidation      34370      90.000         F5  60 months

                 title  total_acc  verification_status loan_status
0   Debt consolidation          8      Source Verified           1
1          Credit Loan         12             Verified           0
2   Debt consolidation         16      Source Verified           0
3   Debt consolidation         19      Source Verified           0
4  Debt Connsolidation         59             Verified           0

[5 rows x 28 columns]
```

[118]:
```python
grade = {'A':0, 'B':1, 'C':2, 'D':3, 'E':4, 'F':5, 'G':6}
df5['grade'] = df5['grade'].map(grade).astype('Int32')
```

[119]:
```
print(df5.grade)
```

```
0        4
1        1
2        1
3        1
4        5
        ..
79995    6
79996    2
79997    1
79998    3
79999    1
Name: grade, Length: 80000, dtype: Int32
```

[120]:
```
df5.head()
```

[120]:
```
   addr_state  annual_inc earliest_cr_line  emp_length  \
0           5   85000.000           Jul-97           2
1           4   40000.000           Apr-87           2
2           9   60000.000           Aug-07           2
3          14  100742.000           Sep-80           2
4          20   80000.000           Jul-99           2

                        emp_title  fico_range_high  fico_range_low  grade  \
0                          Deputy              744             740      4
1  Department of Veterans Affairs              724             720      1
2                Marble polishing              679             675      1
3                         printer              664             660      1
4                   Southern Mgmt              669             665      5

   home_ownership application_type  … pub_rec_bankruptcies  \
0               0       Individual  …                0.000
1               1       Individual  …                0.000
2               0       Individual  …                0.000
3               0       Individual  …                0.000
4               1       Individual  …                0.000

              purpose  revol_bal  revol_util  sub_grade        term  \
0  debt_consolidation       5338      93.600         E1   60 months
1  debt_consolidation      19944      60.300         B1   36 months
2  debt_consolidation      23199      88.500         B5   36 months
3  debt_consolidation      18425      69.000         B2   36 months
4  debt_consolidation      34370      90.000         F5   60 months

                title  total_acc verification_status loan_status
0  Debt consolidation          8     Source Verified           1
1         Credit Loan         12            Verified           0
```

```
2   Debt consolidation         16      Source Verified           0
3   Debt consolidation         19      Source Verified           0
4  Debt Connsolidation         59             Verified           0

[5 rows x 28 columns]
```

[121]:

```
sub_grade = {'A1':0,  'A2':1,  'A3':2,  'A4':3,  'A5':4,
                 'B1':5,  'B2':6,  'B3':7,  'B4':8,  'B5':9,
                 'C1':10, 'C2':11, 'C3':12, 'C4':13, 'C5':14,
                 'D1':15, 'D2':16, 'D3':17, 'D4':18, 'D5':19,
                 'E1':20, 'E2':21, 'E3':22, 'E4':23, 'E5':24,
                 'F1':25, 'F2':26, 'F3':27, 'F4':28, 'F5':29,
                 'G1':30, 'G2':31, 'G3':32, 'G4':33, 'G5':34}
df5['sub_grade'] = df5['sub_grade'].map(sub_grade).astype('Int32')
```

[122]:

```
print(df5.sub_grade)
```

```
0        20
1         5
2         9
3         6
4        29
         ..
79995    32
79996    10
79997     8
79998    19
79999     8
Name: sub_grade, Length: 80000, dtype: Int32
```

[123]:

```
# Step 1: Remove 'months' from the 'term' column
df5['term'] = df5['term'].str.replace(' months', '')

# Step 2: Convert the column to numeric (int or float)
df5['term'] = pd.to_numeric(df5['term'])
```

[124]:

```
term_mapping = {36: 0, 60: 1}
df5['term'] = df5['term'].map(term_mapping).astype('Int32')
```

[125]:

```
df5.head()
```

[125]:

```
   addr_state  annual_inc earliest_cr_line  emp_length  \
0           5   85000.000           Jul-97           2
1           4   40000.000           Apr-87           2
2           9   60000.000           Aug-07           2
3          14  100742.000           Sep-80           2
4          20   80000.000           Jul-99           2
```

```
                     emp_title  fico_range_high  fico_range_low  grade  \
0                       Deputy              744             740      4
1  Department of Veterans Affairs           724             720      1
2             Marble polishing              679             675      1
3                      printer              664             660      1
4                Southern Mgmt              669             665      5

   home_ownership application_type  … pub_rec_bankruptcies  \
0               0       Individual  …                0.000
1               1       Individual  …                0.000
2               0       Individual  …                0.000
3               0       Individual  …                0.000
4               1       Individual  …                0.000

              purpose  revol_bal  revol_util  sub_grade  term  \
0  debt_consolidation       5338      93.600         20     1
1  debt_consolidation      19944      60.300          5     0
2  debt_consolidation      23199      88.500          9     0
3  debt_consolidation      18425      69.000          6     0
4  debt_consolidation      34370      90.000         29     1

                 title  total_acc  verification_status loan_status
0   Debt consolidation          8      Source Verified           1
1          Credit Loan         12             Verified           0
2   Debt consolidation         16      Source Verified           0
3   Debt consolidation         19      Source Verified           0
4  Debt Connsolidation         59             Verified           0

[5 rows x 28 columns]
```

[126]:
```
for i in df5.select_dtypes(include=['category']).columns:
    print('Unique values in', i, 'are :')
    print(df5[i].value_counts(dropna=False))
    print('*'*50)
```

```
Unique values in earliest_cr_line are :
earliest_cr_line
Sep-03    547
Aug-03    545
Aug-01    544
Oct-01    541
Sep-02    539
         …
Jul-65      1
Sep-59      1
Sep-65      1
Jul-64      1
Nov-66      1
```

```
Name: count, Length: 640, dtype: int64
************************************************
Unique values in emp_title are :
emp_title
NaN                              5018
Teacher                          1278
Manager                          1194
Owner                             592
RN                                526
                                 …
Hotel Desk Coordinator              1
Hotel & Travel Credit Union         1
Hot oiler                           1
Hostler                             1
MyBuys                              1
Name: count, Length: 36662, dtype: int64
************************************************
Unique values in application_type are :
application_type
Individual    78446
Joint App      1554
Name: count, dtype: int64
************************************************
Unique values in initial_list_status are :
initial_list_status
w    46745
f    33255
Name: count, dtype: int64
************************************************
Unique values in purpose are :
purpose
debt_consolidation    46418
credit_card           17506
home_improvement       5268
other                  4683
major_purchase         1746
small_business          950
medical                 902
car                     868
moving                  548
vacation                518
house                   413
wedding                 110
renewable_energy         54
educational              16
Name: count, dtype: int64
************************************************
Unique values in title are :
```

```
title
Debt consolidation                           39396
Credit card refinancing                      14802
Home improvement                              4542
Other                                         4035
Major purchase                                1422
                                             …
Get on the right track                           1
Get me out of debt with lower interest!          1
Get it right                                     1
Get it done                                      1
Mama to Be                                       1
Name: count, Length: 5349, dtype: int64
**************************************************
Unique values in verification_status are :
verification_status
Source Verified    30855
Verified           24876
Not Verified       24269
Name: count, dtype: int64
**************************************************
```

[127]:

```python
# Define mapping for purpose column
purpose_mapping = {'debt_consolidation': 0,
                   'credit_card': 1,
                   'home_improvement': 2,
                   'other': 3,
                   'major_purchase': 4,
                   'small_business': 5,
                   'medical': 6,
                   'car': 7,
                   'moving': 8,
                   'vacation': 9,
                   'house': 10,
                   'wedding': 11,
                   'renewable_energy': 12,
                   'educational': 13}

# Map the values in the DataFrame
df5['purpose'] = df5['purpose'].map(purpose_mapping).astype('Int32')
```

[128]:

```python
print(df5.purpose)
```

```
0        0
1        0
2        0
3        0
4        0
```

```
        ..
79995    0
79996    0
79997    0
79998    1
79999    0
Name: purpose, Length: 80000, dtype: Int32
```

[129]:
```
# Define mapping for initial_list_status column
initial_list_status_mapping = {'w': 0, 'f': 1}

# Map the values in the DataFrame
df5['initial_list_status'] = df5['initial_list_status'].
  ↪map(initial_list_status_mapping)
```

[130]:
```
print(df5.initial_list_status)
```

```
0        0
1        0
2        0
3        0
4        1
        ..
79995    0
79996    0
79997    0
79998    0
79999    0
Name: initial_list_status, Length: 80000, dtype: category
Categories (2, int64): [1, 0]
```

[131]:
```
# Define mapping for application_type column
application_type_mapping = {'Individual': 0,
                            'Joint App': 1}

# Map the values in the DataFrame
df5['application_type'] = df5['application_type'].map(application_type_mapping).
  ↪astype('Int32')
```

[132]:
```
print(df5.application_type)
```

```
0        0
1        0
2        0
3        0
4        0
        ..
79995    0
79996    0
```

```
79997    0
79998    0
79999    0
Name: application_type, Length: 80000, dtype: Int32
```

[133]:
```
df5.head(50)
```

[133]:
```
    addr_state  annual_inc earliest_cr_line  emp_length  \
0            5   85000.000           Jul-97           2
1            4   40000.000           Apr-87           2
2            9   60000.000           Aug-07           2
3           14  100742.000           Sep-80           2
4           20   80000.000           Jul-99           2
5            4   51488.000           May-91        <NA>
6           34  100000.000           Oct-86           2
7           38   35028.000           Nov-95           0
8            9   59292.000           Dec-07        <NA>
9            4   65000.000           Jun-04           0
10          48   35000.000           Jul-99           0
11          44   30000.000           Aug-96           1
12          34  100000.000           Oct-98           1
13           4   80000.000           May-07           0
14           4   73000.000           Oct-00           0
15          43   48500.000           Jan-05           1
16           1   52512.000           Apr-04        <NA>
17          16   83840.000           Sep-00           0
18           2  100000.000           Sep-93        <NA>
19           4  852000.000           Oct-01           0
20           3   85000.000           May-01           0
21           3   50000.000           Jun-06           1
22          22   72000.000           Jan-98           2
23          23   24000.000           Jan-99           0
24          20   50000.000           Jul-98           2
25          38  221000.000           Jun-03           0
26           4   65000.000           Jul-09           0
27          14   80000.000           Nov-06           0
28           9   62000.000           Feb-06           1
29          18   48000.000           Aug-10           1
30          23   25000.000           Jul-99           0
31           9   54000.000           Apr-01           0
32           4   37000.000           Dec-02           1
33          11   65000.000           Jun-78           1
34          34   96596.000           Oct-04           0
35          27   68000.000           Feb-04           0
36           4   40000.000           Sep-06           1
37          20  152000.000           May-99           2
38          34   45000.000           Jan-88           2
```

| | | | | |
|---|---|---|---|---|
| 39 | 14 | 120000.000 | Mar-92 | 2 |
| 40 | 43 | 42000.000 | Jun-11 | 0 |
| 41 | 46 | 57408.000 | Jul-01 | 0 |
| 42 | 9 | 45000.000 | Feb-95 | 1 |
| 43 | 4 | 37000.000 | Dec-05 | 0 |
| 44 | 34 | 60000.000 | Nov-97 | 2 |
| 45 | 4 | 74000.000 | Sep-03 | 2 |
| 46 | 43 | 60000.000 | Feb-91 | 0 |
| 47 | 34 | 117000.000 | Dec-00 | 0 |
| 48 | 34 | 80000.000 | May-03 | 0 |
| 49 | 9 | 78000.000 | Jul-03 | <NA> |

| | emp_title | fico_range_high | fico_range_low | \ |
|---|---|---|---|---|
| 0 | Deputy | 744 | 740 | |
| 1 | Department of Veterans Affairs | 724 | 720 | |
| 2 | Marble polishing | 679 | 675 | |
| 3 | printer | 664 | 660 | |
| 4 | Southern Mgmt | 669 | 665 | |
| 5 | NaN | 679 | 675 | |
| 6 | RN | 699 | 695 | |
| 7 | SHHC | 679 | 675 | |
| 8 | NaN | 664 | 660 | |
| 9 | Nurse | 684 | 680 | |
| 10 | Carpenter | 679 | 675 | |
| 11 | Office manager | 749 | 745 | |
| 12 | Vice President | 694 | 690 | |
| 13 | Executive chef | 744 | 740 | |
| 14 | Graye | 674 | 670 | |
| 15 | Manager | 679 | 675 | |
| 16 | NaN | 684 | 680 | |
| 17 | Manager | 729 | 725 | |
| 18 | NaN | 744 | 740 | |
| 19 | Logistics Corrdinator | 689 | 685 | |
| 20 | Home mortgage loan officer | 689 | 685 | |
| 21 | Truck Driver | 674 | 670 | |
| 22 | Inspector | 684 | 680 | |
| 23 | Special Education Para-Professional | 669 | 665 | |
| 24 | Service Master Chesapeake | 684 | 680 | |
| 25 | Senior Director of Engineering | 669 | 665 | |
| 26 | CFO | 679 | 675 | |
| 27 | Foreman | 689 | 685 | |
| 28 | Extension Agent | 674 | 670 | |
| 29 | Operations Manager | 724 | 720 | |
| 30 | teller | 664 | 660 | |
| 31 | Satellite Manager | 719 | 715 | |
| 32 | BAKER | 664 | 660 | |
| 33 | Instructor | 709 | 705 | |

| | | | |
|---|---|---|---|
| 34 | Financial Analyst | 699 | 695 |
| 35 | Tech Coordinator, FileMaker Developer | 719 | 715 |
| 36 | Mechanic | 694 | 690 |
| 37 | special agent | 734 | 730 |
| 38 | Creative Director | 684 | 680 |
| 39 | Jerico Inc | 739 | 735 |
| 40 | Chiropractor | 679 | 675 |
| 41 | Civil Service | 709 | 705 |
| 42 | Owner | 689 | 685 |
| 43 | PriMed Management | 669 | 665 |
| 44 | Sales Rep | 714 | 710 |
| 45 | Psychiatric Technician | 699 | 695 |
| 46 | Design Team Memeber | 704 | 700 |
| 47 | Accounting Manager | 664 | 660 |
| 48 | Sales Rep | 719 | 715 |
| 49 | NaN | 754 | 750 |

| | grade | home_ownership | application_type | … | pub_rec_bankruptcies | \ |
|---|---|---|---|---|---|---|
| 0 | 4 | 0 | 0 | … | 0.000 | |
| 1 | 1 | 1 | 0 | … | 0.000 | |
| 2 | 1 | 0 | 0 | … | 0.000 | |
| 3 | 1 | 0 | 0 | … | 0.000 | |
| 4 | 5 | 1 | 0 | … | 0.000 | |
| 5 | 3 | 0 | 0 | … | 0.000 | |
| 6 | 2 | 0 | 0 | … | 0.000 | |
| 7 | 2 | 1 | 0 | … | 0.000 | |
| 8 | 1 | 0 | 0 | … | 0.000 | |
| 9 | 3 | 1 | 0 | … | 0.000 | |
| 10 | 1 | 0 | 0 | … | 0.000 | |
| 11 | 1 | 2 | 0 | … | 0.000 | |
| 12 | 1 | 1 | 0 | … | 0.000 | |
| 13 | 0 | 1 | 0 | … | 0.000 | |
| 14 | 4 | 1 | 0 | … | 0.000 | |
| 15 | 2 | 2 | 0 | … | 0.000 | |
| 16 | 2 | 0 | 0 | … | 1.000 | |
| 17 | 2 | 1 | 0 | … | 0.000 | |
| 18 | 1 | 0 | 0 | … | 0.000 | |
| 19 | 3 | 0 | 0 | … | 0.000 | |
| 20 | 4 | 0 | 0 | … | 0.000 | |
| 21 | 2 | 1 | 0 | … | 1.000 | |
| 22 | 1 | 1 | 0 | … | 0.000 | |
| 23 | 3 | 0 | 0 | … | 0.000 | |
| 24 | 4 | 1 | 0 | … | 0.000 | |
| 25 | 2 | 2 | 0 | … | 0.000 | |
| 26 | 1 | 1 | 0 | … | 0.000 | |
| 27 | 5 | 0 | 0 | … | 1.000 | |
| 28 | 2 | 0 | 0 | … | 0.000 | |

| | | | | | |
|---|---|---|---|---|---|
| 29 | 1 | 0 | 0 | … | 0.000 |
| 30 | 4 | 0 | 0 | … | 0.000 |
| 31 | 2 | 0 | 0 | … | 0.000 |
| 32 | 1 | 0 | 0 | … | 1.000 |
| 33 | 0 | 1 | 0 | … | 0.000 |
| 34 | 1 | 1 | 0 | … | 0.000 |
| 35 | 2 | 0 | 0 | … | 0.000 |
| 36 | 1 | 2 | 0 | … | 0.000 |
| 37 | 0 | 0 | 0 | … | 0.000 |
| 38 | 2 | 0 | 0 | … | 0.000 |
| 39 | 0 | 0 | 0 | … | 0.000 |
| 40 | 2 | 0 | 0 | … | 0.000 |
| 41 | 1 | 1 | 0 | … | 0.000 |
| 42 | 1 | 0 | 0 | … | 0.000 |
| 43 | 4 | 1 | 0 | … | 1.000 |
| 44 | 0 | 2 | 0 | … | 0.000 |
| 45 | 2 | 1 | 0 | … | 0.000 |
| 46 | 1 | 0 | 0 | … | 0.000 |
| 47 | 1 | 1 | 0 | … | 0.000 |
| 48 | 1 | 1 | 0 | … | 0.000 |
| 49 | 0 | 1 | 0 | … | 0.000 |

| | purpose | revol_bal | revol_util | sub_grade | term | \ |
|---|---|---|---|---|---|---|
| 0 | 0 | 5338 | 93.600 | 20 | 1 | |
| 1 | 0 | 19944 | 60.300 | 5 | 0 | |
| 2 | 0 | 23199 | 88.500 | 9 | 0 | |
| 3 | 0 | 18425 | 69.000 | 6 | 0 | |
| 4 | 0 | 34370 | 90.000 | 29 | 1 | |
| 5 | 2 | 10747 | 53.900 | 17 | 0 | |
| 6 | 1 | 32488 | 54.100 | 10 | 0 | |
| 7 | 0 | 13147 | 78.300 | 13 | 0 | |
| 8 | 0 | 1054 | 23.400 | 8 | 0 | |
| 9 | 0 | 8991 | 64.700 | 18 | 0 | |
| 10 | 0 | 23293 | 71.700 | 8 | 0 | |
| 11 | 0 | 8355 | 23.000 | 9 | 0 | |
| 12 | 0 | 16112 | 49.300 | 9 | 0 | |
| 13 | 3 | 12405 | 44.900 | 4 | 0 | |
| 14 | 1 | 15343 | 84.800 | 20 | 0 | |
| 15 | 0 | 8236 | 30.800 | 12 | 0 | |
| 16 | 3 | 3575 | 16.800 | 12 | 0 | |
| 17 | 0 | 21963 | 40.100 | 11 | 1 | |
| 18 | 2 | 8879 | 48.000 | 5 | 0 | |
| 19 | 0 | 16245 | 67.400 | 18 | 1 | |
| 20 | 0 | 21251 | 72.500 | 23 | 1 | |
| 21 | 1 | 3294 | 36.600 | 11 | 0 | |
| 22 | 0 | 14438 | 59.000 | 9 | 0 | |
| 23 | 0 | 7774 | 40.900 | 18 | 0 | |

| | | | | | |
|---|---|---|---|---|---|
| 24 | 0 | 21717 | 85.200 | 23 | 1 |
| 25 | 2 | 32543 | 67.700 | 10 | 1 |
| 26 | 1 | 7952 | 59.800 | 8 | 0 |
| 27 | 5 | 2347 | 17.800 | 27 | 1 |
| 28 | 0 | 9454 | 85.900 | 13 | 0 |
| 29 | 1 | 6051 | 28.000 | 9 | 0 |
| 30 | 1 | 14436 | 52.100 | 21 | 0 |
| 31 | 0 | 21495 | 52.400 | 11 | 1 |
| 32 | 1 | 5656 | 67.300 | 6 | 0 |
| 33 | 1 | 27276 | 61.600 | 0 | 0 |
| 34 | 0 | 15147 | 84.100 | 9 | 0 |
| 35 | 2 | 512 | 4.700 | 14 | 1 |
| 36 | 1 | 3982 | 76.600 | 6 | 0 |
| 37 | 1 | 17526 | 39.100 | 4 | 1 |
| 38 | 1 | 48144 | 92.800 | 11 | 0 |
| 39 | 1 | 21543 | 33.000 | 4 | 0 |
| 40 | 0 | 8842 | 48.300 | 13 | 0 |
| 41 | 1 | 14913 | 37.900 | 5 | 0 |
| 42 | 1 | 11504 | 41.100 | 7 | 0 |
| 43 | 0 | 6616 | 76.000 | 24 | 0 |
| 44 | 1 | 9502 | 48.000 | 3 | 0 |
| 45 | 3 | 12445 | 79.800 | 12 | 0 |
| 46 | 0 | 46475 | 66.800 | 9 | 0 |
| 47 | 0 | 9098 | 52.600 | 7 | 0 |
| 48 | 0 | 5568 | 19.500 | 9 | 0 |
| 49 | 0 | 12137 | 33.100 | 3 | 0 |

| | title | total_acc | verification_status | loan_status |
|---|---|---|---|---|
| 0 | Debt consolidation | 8 | Source Verified | 1 |
| 1 | Credit Loan | 12 | Verified | 0 |
| 2 | Debt consolidation | 16 | Source Verified | 0 |
| 3 | Debt consolidation | 19 | Source Verified | 0 |
| 4 | Debt Connsolidation | 59 | Verified | 0 |
| 5 | Home improvement | 37 | Verified | 0 |
| 6 | Credit card refinancing | 36 | Verified | 0 |
| 7 | Credit consolidation sought | 19 | Not Verified | 0 |
| 8 | NaN | 23 | Verified | 0 |
| 9 | Debt consolidation | 20 | Source Verified | 0 |
| 10 | Debt consolidation | 24 | Source Verified | 0 |
| 11 | Debt consolidation | 19 | Not Verified | 0 |
| 12 | Debt consolidation | 15 | Source Verified | 0 |
| 13 | Other | 8 | Not Verified | 0 |
| 14 | card consolidate | 7 | Not Verified | 0 |
| 15 | Debt consolidation | 21 | Verified | 1 |
| 16 | Other | 36 | Verified | 0 |
| 17 | Debt consolidation | 45 | Source Verified | 1 |
| 18 | Home improvement | 21 | Source Verified | 1 |

```
19          Debt consolidation         24            Verified           1
20          Debt consolidation         18     Source Verified           1
21     Credit card refinancing         14            Verified           0
22                         NaN         26            Verified           1
23          Debt consolidation         25     Source Verified           0
24                        Mine         18            Verified           1
25            Home improvement         28     Source Verified           0
26     Credit card refinancing         11     Source Verified           0
27                    Business         21     Source Verified           1
28            Credit Liberator         14            Verified           0
29     Credit card refinancing         13     Source Verified           0
30     Credit card refinancing         30        Not Verified           0
31          Debt consolidation         27        Not Verified           0
32     Credit card refinancing         14     Source Verified           0
33     Credit card refinancing         22        Not Verified           1
34          Debt consolidation         26            Verified           0
35            Home improvement         19     Source Verified           1
36     Credit card refinancing         18     Source Verified           0
37     Credit card refinancing         35     Source Verified           0
38     Credit card refinancing         19     Source Verified           0
39     Credit card refinancing         49     Source Verified           0
40          Debt consolidation         14     Source Verified           0
41          Credit card payoff         16            Verified           0
42     Credit card refinancing         31        Not Verified           0
43              Debt Free Soon         24     Source Verified           0
44     Credit card refinancing         22            Verified           0
45                       Other         22     Source Verified           0
46          Debt consolidation         39        Not Verified           0
47          Debt consolidation         56        Not Verified           1
48                         NaN         18     Source Verified           0
49          Debt consolidation         22        Not Verified           0

[50 rows x 28 columns]
```

[134]:
```python
pd.set_option('display.max_columns', None)
```

[135]:
```python
# Define mapping for verification_status column
verification_status_mapping = {
    'Source Verified': 0,
    'Verified': 1,
    'Not Verified': 2
}

# Map the values in the DataFrame
df5['verification_status'] = df5['verification_status'].
 ↪map(verification_status_mapping).astype('Int32')
```

[136]:
```python
print(df5.verification_status)
```

```
0        0
1        1
2        0
3        0
4        1
        ..
79995    1
79996    2
79997    0
79998    0
79999    0
Name: verification_status, Length: 80000, dtype: Int32
```

[137]:
```python
# Check for null values in each column
null_counts = df5.isnull().sum()

# Print the null counts
print(null_counts)
```

```
addr_state                 0
annual_inc                 0
earliest_cr_line           0
emp_length              4588
emp_title               5018
fico_range_high            0
fico_range_low             0
grade                      0
home_ownership            19
application_type           0
initial_list_status        0
int_rate                   0
loan_amnt                  0
num_actv_bc_tl          3948
mort_acc                2771
tot_cur_bal             3948
open_acc                   0
pub_rec                    0
pub_rec_bankruptcies      31
purpose                    0
revol_bal                  0
revol_util                53
sub_grade                  0
term                       0
title                    970
total_acc                  0
verification_status        0
```

```
loan_status               0
dtype: int64
```

[138]:
```
df5.isnull().sum()
```

[138]:
```
addr_state                0
annual_inc                0
earliest_cr_line          0
emp_length             4588
emp_title              5018
fico_range_high           0
fico_range_low            0
grade                     0
home_ownership           19
application_type          0
initial_list_status       0
int_rate                  0
loan_amnt                 0
num_actv_bc_tl         3948
mort_acc               2771
tot_cur_bal            3948
open_acc                  0
pub_rec                   0
pub_rec_bankruptcies     31
purpose                   0
revol_bal                 0
revol_util               53
sub_grade                 0
term                      0
title                   970
total_acc                 0
verification_status       0
loan_status               0
dtype: int64
```

[139]:
```
df5['num_actv_bc_tl'].fillna(df1['num_actv_bc_tl'].mean(), inplace=True)
df5['mort_acc'].fillna(df1['mort_acc'].mean(), inplace=True)
df5['tot_cur_bal'].fillna(df1['tot_cur_bal'].mean(), inplace=True)
df5['emp_length'].fillna(0, inplace=True)
revol_util_mean = df5['revol_util'].mean()
df5['revol_util'].fillna(revol_util_mean, inplace=True)
```

[140]:
```
df5.isnull().sum()
```

[140]:
```
addr_state                0
annual_inc                0
earliest_cr_line          0
emp_length                0
```

```
emp_title               5018
fico_range_high            0
fico_range_low             0
grade                      0
home_ownership            19
application_type           0
initial_list_status        0
int_rate                   0
loan_amnt                  0
num_actv_bc_tl             0
mort_acc                   0
tot_cur_bal                0
open_acc                   0
pub_rec                    0
pub_rec_bankruptcies      31
purpose                    0
revol_bal                  0
revol_util                 0
sub_grade                  0
term                       0
title                    970
total_acc                  0
verification_status        0
loan_status                0
dtype: int64
```

[141]:
```python
# Replace null values in 'home_ownership' with 3
df5['home_ownership'] = df5['home_ownership'].fillna(3)
```

[ ]:
```python

```

[147]:
```python
# Drop specified columns
df5.drop(['emp_title', 'title', 'earliest_cr_line'], axis=1, inplace=True)
```

[148]:
```python
df5.head()
```

[148]:
```
   addr_state  annual_inc  emp_length  fico_range_high  fico_range_low  grade  \
0           5   85000.000           2              744             740      4
1           4   40000.000           2              724             720      1
2           9   60000.000           2              679             675      1
3          14  100742.000           2              664             660      1
4          20   80000.000           2              669             665      5

   home_ownership  application_type initial_list_status  int_rate  loan_amnt  \
0               0                 0                   0    18.990      18075
1               1                 0                   0    10.160       8800
2               0                 0                   0    11.470      18000
```

```
3             0              0               0    9.160     20000
4             1              0               1   23.830     35000

   num_actv_bc_tl  mort_acc  tot_cur_bal  open_acc  pub_rec  \
0           1.000     1.000   319479.000         7        0
1           4.000     0.000    19944.000         5        0
2           4.000     2.000    23199.000         7        0
3           4.000     1.000    72651.000        12        0
4          14.000     7.000    64631.000        23        0

   pub_rec_bankruptcies  purpose  revol_bal  revol_util  sub_grade  term  \
0                 0.000        0       5338      93.600         20     1
1                 0.000        0      19944      60.300          5     0
2                 0.000        0      23199      88.500          9     0
3                 0.000        0      18425      69.000          6     0
4                 0.000        0      34370      90.000         29     1

   total_acc  verification_status  loan_status
0          8                    0            1
1         12                    1            0
2         16                    0            0
3         19                    0            0
4         59                    1            0
```

[149]:
```python
imputer = KNNImputer(n_neighbors=5)
```

[150]:
```python
X = df5.drop(['loan_status'], axis=1)
y = df5['loan_status']
```

[151]:
```python
# Splitting data into training and test set:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=7, stratify=y)
print(X_train.shape, X_test.shape)
```

```
(56000, 24) (24000, 24)
```

[152]:
```python
#Fit and transform the train data
X_train = pd.DataFrame(imputer.fit_transform(X_train), columns=X_train.columns)
```

[153]:
```python
X_test = pd.DataFrame(imputer.transform(X_test),columns=X_test.columns)
```

[154]:
```python
#Checking that no column has missing values in train or test sets
print(X_train.isna().sum())
print('-'*30)
print(X_test.isna().sum())
```

```
addr_state              0
annual_inc              0
```

```
emp_length              0
fico_range_high         0
fico_range_low          0
grade                   0
home_ownership          0
application_type        0
initial_list_status     0
int_rate                0
loan_amnt               0
num_actv_bc_tl          0
mort_acc                0
tot_cur_bal             0
open_acc                0
pub_rec                 0
pub_rec_bankruptcies    0
purpose                 0
revol_bal               0
revol_util              0
sub_grade               0
term                    0
total_acc               0
verification_status     0
dtype: int64
------------------------------
addr_state              0
annual_inc              0
emp_length              0
fico_range_high         0
fico_range_low          0
grade                   0
home_ownership          0
application_type        0
initial_list_status     0
int_rate                0
loan_amnt               0
num_actv_bc_tl          0
mort_acc                0
tot_cur_bal             0
open_acc                0
pub_rec                 0
pub_rec_bankruptcies    0
purpose                 0
revol_bal               0
revol_util              0
sub_grade               0
term                    0
total_acc               0
verification_status     0
```

```
dtype: int64
```

[162]:
```
import numpy as np

def inverse_mapping(x, y):
    # Create a mapping from numerical values to original categories
    inv_dict = {v: k for k, v in x.items()}

    # Convert the categorical column to numerical
    X_train[y] = X_train[y].astype(float)
    X_test[y] = X_test[y].astype(float)

    # Round the numerical values
    X_train[y] = np.round(X_train[y])
    X_test[y] = np.round(X_test[y])

    # Map the rounded numerical values back to original categories
    X_train[y] = X_train[y].map(inv_dict).astype('category')
    X_test[y] = X_test[y].map(inv_dict).astype('category')
```

[167]:
```
import numpy as np

def inverse_mapping(x, y):
    # Create a mapping from numerical values to original categories
    inv_dict = {v: k for k, v in x.items()}

    # Preprocess the column to convert string values to numerical
    if y == 'emp_length':
        X_train[y] = X_train[y].replace({'< 1 year': 0, '10+ years': 10}).
 ↪astype(float)
        X_test[y] = X_test[y].replace({'< 1 year': 0, '10+ years': 10}).
 ↪astype(float)
    elif y == 'term':
        X_train[y] = X_train[y].replace({'36 months': 0, '60 months': 1}).
 ↪astype(float)
        X_test[y] = X_test[y].replace({'36 months': 0, '60 months': 1}).
 ↪astype(float)
    elif y in ['grade', 'sub_grade', 'home_ownership', 'verification_status',␣
 ↪'purpose', 'application_type']:
        X_train[y] = X_train[y].astype(float)
        X_test[y] = X_test[y].astype(float)

    # Round the numerical values
    X_train[y] = np.round(X_train[y])
    X_test[y] = np.round(X_test[y])

    # Map the rounded numerical values back to original categories
```

```
    X_train[y] = X_train[y].map(inv_dict).astype('category')
    X_test[y] = X_test[y].map(inv_dict).astype('category')
```

[168]:
```
cols = X_train.select_dtypes(include=['object','category'])
for i in cols.columns:
    print(X_train[i].value_counts(dropna=False))
    print('*'*30)
```

```
emp_length
<5 Years      27953
10+ years     18353
6-10 years     9694
Name: count, dtype: int64
******************************
grade
B    16531
C    15715
A     9705
D     8400
E     3945
F     1324
G      380
Name: count, dtype: int64
******************************
home_ownership
MORTGAGE    27559
RENT        22294
OWN          6126
OTHER          17
NONE            4
Name: count, dtype: int64
******************************
term
3 years    42576
5 years    13424
Name: count, dtype: int64
******************************
```

## 1.1 Encoding categorical variables

[169]:
```
X_train = pd.get_dummies(X_train, drop_first=True)
X_test = pd.get_dummies(X_test, drop_first=True)
print(X_train.shape, X_test.shape)
```

```
(56000, 33) (24000, 32)
```

[170]:
```
X_train.columns
```

[170]: Index(['addr_state', 'annual_inc', 'fico_range_high', 'fico_range_low',
       'application_type', 'initial_list_status', 'int_rate', 'loan_amnt',
       'num_actv_bc_tl', 'mort_acc', 'tot_cur_bal', 'open_acc', 'pub_rec',
       'pub_rec_bankruptcies', 'purpose', 'revol_bal', 'revol_util',
       'sub_grade', 'total_acc', 'verification_status',
       'emp_length_6-10 years', 'emp_length_<5 Years', 'grade_B', 'grade_C',
       'grade_D', 'grade_E', 'grade_F', 'grade_G', 'home_ownership_NONE',
       'home_ownership_OTHER', 'home_ownership_OWN', 'home_ownership_RENT',
       'term_5 years'],
      dtype='object')

## 2 Model Building

[171]:
```
## Function to calculate different metric scores of the model - Accuracy,␣
↪Recall and Precision
def get_metrics_score(model, flag=True):
    '''
    model : classifier to predict values of X
    flag: Flag to print metric score dataframe. (default=True)
    '''
    # defining an empty list to store train and test results
    scores = []
    pred_train = model.predict(X_train)
    pred_test = model.predict(X_test)
    train_acc = model.score(X_train,y_train)
    test_acc = model.score(X_test,y_test)
    train_recall = metrics.recall_score(y_train,pred_train)
    test_recall = metrics.recall_score(y_test,pred_test)
    train_precision = metrics.precision_score(y_train,pred_train)
    test_precision = metrics.precision_score(y_test,pred_test)
    train_f1 = f1_score(y_train,pred_train)
    test_f1 = f1_score(y_test,pred_test)
    scores.extend(
        (
            train_acc, test_acc,
            train_recall, test_recall,
            train_precision, test_precision,
            train_f1, test_f1
        )
    )

    # If the flag is set to True then only the following print statements will␣
↪be dispayed. The default value is set to True.
    if flag == True:
        metric_names = [
            'Train Accuracy', 'Test Accuracy', 'Train Recall', 'Test Recall',
```

```
            'Train Precision', 'Test Precision', 'Train F1-Score', 'Test␣
↪F1-Score'
        ]
        cols = ['Metric', 'Score']
        records = [(name, score) for name, score in zip(metric_names, scores)]
        display(pd.DataFrame.from_records(records, columns=cols,␣
↪index='Metric').T)

    return scores # returning the list with train and test scores
```

[172]:
```
## Function to create confusion matrix
def make_confusion_matrix(model,  y_actual, labels=[1, 0], xtest=X_test):
    """
    model : classifier to predict values of X
    y_actual : ground truth
    """
    y_predict = model.predict(xtest)
    cm = metrics.confusion_matrix(y_actual, y_predict, labels=[0, 1])
    df_cm = pd.DataFrame(cm, index=["Yes", "No"], columns=["Yes", "No"])

    group_counts = [f"{value:0.0f}" for value in cm.flatten()]
    group_percentages = [f"{value:.2%}" for value in cm.flatten()/np.sum(cm)]

    labels = [f"{gc}\n{gp}" for gc, gp in zip(group_counts, group_percentages)]
    labels = np.asarray(labels).reshape(2,2)

    plt.figure(figsize = (10, 7))
    sns.heatmap(df_cm, annot=labels, fmt='')
    plt.ylabel("Actual", fontsize=14)
    plt.xlabel("Predicted", fontsize=14);
```

[173]:
```
def show_model_performance(model: list, model_names: list):
    results = []
    for model, name in zip(models, model_names):
        (acc_train, acc_test,
         recall_train, recall_test,
         precision_train, precision_test,
         f1_train, f1_test) = get_metrics_score(model, False)

        results.append((name, acc_train, acc_test, recall_train, recall_test,
                        precision_train, precision_test, f1_train, f1_test))

    cols = [
        'Model', 'Train Acc', 'Test Accuracy', 'Train Recall',
        'Test Recall', 'Train Precision', 'Test Precision',
        'Train F1-Score', 'Test F1-Score'
    ]
```

```
    comparison_frame = pd.DataFrame.from_records(results, columns=cols,␣
↪index='Model')
    # Sorting models in decreasing order of test f1-score
    display(comparison_frame.sort_values(by='Test F1-Score', ascending=False))
```

# 3 Cross Validation Scores

[174]:
```
lr = LogisticRegression(random_state=1)
lr.fit(X_train, y_train)
```

[174]: LogisticRegression(random_state=1)

[175]:
```
scoring = 'recall'
#Setting number of splits equal to 5
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
cv_result_bfr = cross_val_score(estimator=lr, X=X_train, y=y_train,␣
↪scoring=scoring, cv=kfold)
#Plotting boxplots for CV scores of model defined above
plt.boxplot(cv_result_bfr);
```


[177]:
```
import pandas as pd

# Assuming X_test is your test data DataFrame
# Check if the column 'home_ownership_NONE' exists
if 'home_ownership_NONE' not in X_test.columns:
    # If the column doesn't exist, add it with appropriate values
    X_test['home_ownership_NONE'] = 0  # or any default value you want to assign
```

```
# Now, the 'home_ownership_NONE' column exists in your test data with␣
↪appropriate values
```

[192]:
```
get_metrics_score(lr)
```

```
Metric  Train Accuracy  Test Accuracy  Train Recall  Test Recall  \
Score            0.800          0.800         0.000        0.000

Metric  Train Precision  Test Precision  Train F1-Score  Test F1-Score
Score             0.000           0.000           0.000          0.000
```

[192]:
```
[0.800375, 0.800375, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

## 3.1 Oversampling train data

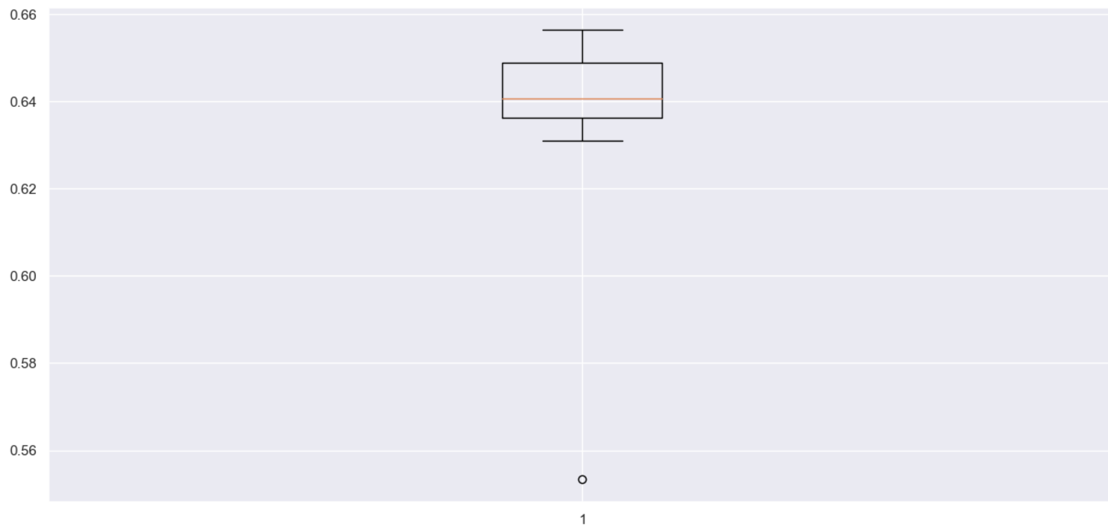
[193]:
```
from imblearn.over_sampling import SMOTE

print("Before UpSampling, counts of label 'Yes': {}".format(sum(y_train==1)))
print("Before UpSampling, counts of label 'No': {} \n".format(sum(y_train==0)))

sm = SMOTE(sampling_strategy = 1 ,k_neighbors = 5, random_state=1)   #Synthetic␣
↪Minority Over Sampling Technique
X_train_over, y_train_over = sm.fit_resample(X_train, y_train)

print("After UpSampling, counts of label 'Yes': {}".
↪format(sum(y_train_over==1)))
print("After UpSampling, counts of label 'No': {} \n".
↪format(sum(y_train_over==0)))

print('After UpSampling, the shape of train_X: {}'.format(X_train_over.shape))
print('After UpSampling, the shape of train_y: {} \n'.format(y_train_over.
↪shape))
```

```
Before UpSampling, counts of label 'Yes': 11179
Before UpSampling, counts of label 'No': 44821

After UpSampling, counts of label 'Yes': 44821
After UpSampling, counts of label 'No': 44821

After UpSampling, the shape of train_X: (89642, 33)
After UpSampling, the shape of train_y: (89642,)
```

[181]:
```
log_reg_over = LogisticRegression(random_state=1)

# Training the basic logistic regression model with the training set
log_reg_over.fit(X_train_over, y_train_over)
```

[181]: LogisticRegression(random_state=1)

[182]:
```
scoring = 'recall'
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)     #Setting␣
 ↪number of splits equal to 5
cv_result_over = cross_val_score(estimator=log_reg_over, X=X_train_over,␣
 ↪y=y_train_over, scoring=scoring, cv=kfold)
#Plotting boxplots for CV scores of model defined above
plt.boxplot(cv_result_over);
```




[194]:
```
get_metrics_score(log_reg_over)
```

```
Metric   Train Accuracy   Test Accuracy   Train Recall   Test Recall   \
Score             0.651           0.659          0.614         0.626

Metric   Train Precision   Test Precision   Train F1-Score   Test F1-Score
Score              0.311            0.320            0.413           0.423
```

[194]:
```
[0.651375,
 0.6595,
 0.6144556758207353,
 0.6257566270089752,
 0.3110678380581469,
 0.31971846006185345,
 0.4130362887465801,
 0.4232072275550536]
```

## 4 Regularisation

[185]:
```
# Choose the type of classifier.
lr_estimator = LogisticRegression(random_state=1)

# Grid of parameters to choose from
parameters = {'C': np.arange(0.1, 1.1, 0.1)}

# Run the grid search
grid_obj = GridSearchCV(lr_estimator, parameters, scoring='recall', n_jobs=-1)
grid_obj = grid_obj.fit(X_train_over, y_train_over)

# Set the clf to the best combination of parameters
lr_estimator = grid_obj.best_estimator_

# Fit the best algorithm to the data.
lr_estimator.fit(X_train_over, y_train_over)
```

[185]:
```
LogisticRegression(C=0.30000000000000004, random_state=1)
```

[187]:
```
# Assuming X_train and X_test are your train and test data DataFrames
# Assuming lr_estimator is your trained logistic regression model

# 1. Check if the test data contains all the columns present during model␣
↪training
missing_columns = set(X_train.columns) - set(X_test.columns)
if missing_columns:
    # If there are missing columns, add them to the test data with appropriate␣
↪values
    for col in missing_columns:
        X_test[col] = 0  # or any default value you want to assign

# 2. Ensure that the order of columns in the test data matches the order in the␣
↪training data
X_test = X_test[X_train.columns]

# Now, the test data should have the same columns and column order as the␣
↪training data
# You should be able to use lr_estimator.predict(X_test) without any issues
```

[195]:
```
get_metrics_score(lr_estimator)
```

```
Metric  Train Accuracy  Test Accuracy  Train Recall  Test Recall  \
Score            0.649          0.657         0.618        0.628

Metric  Train Precision  Test Precision  Train F1-Score  Test F1-Score
Score             0.310           0.318           0.413          0.423
```

[195]:
```
[0.6486607142857143,
 0.6570833333333334,
 0.6183916271580642,
 0.6284700480066792,
 0.3096944718215214,
 0.31825388436740304,
 0.4127040983851228,
 0.422537187763121]
```

[196]:
```python
models = []  # Empty list to store all the models

# Appending pipelines for each model into the list

models.append(
    (
        "DTREE",
        Pipeline(
            steps=[
                ("scaler", StandardScaler()),
                ("decision_tree", DecisionTreeClassifier(random_state=1)),
            ]
        ),
    )
)


models.append(
    (
        "BAGGING",
        Pipeline(
            steps=[
                ("scaler", StandardScaler()),
                ("random_forest", BaggingClassifier(random_state=1)),
            ]
        ),
    )
)


models.append(
    (
        "RF",
        Pipeline(
            steps=[
                ("scaler", StandardScaler()),
                ("random_forest", RandomForestClassifier(random_state=1)),
            ]
```

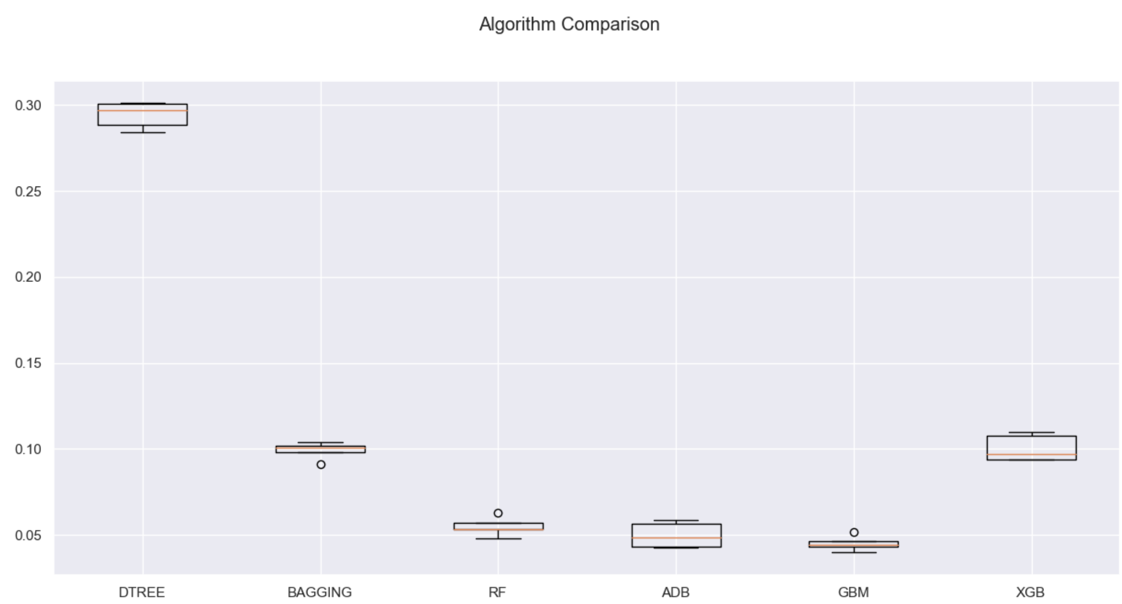
```
        ),
    )
)

models.append(
    (
        "ADB",
        Pipeline(
            steps=[
                ("scaler", StandardScaler()),
                ("adaboost", AdaBoostClassifier(random_state=1)),
            ]
        ),
    )
)


models.append(
    (
        "GBM",
        Pipeline(
            steps=[
                ("scaler", StandardScaler()),
                ("gradient_boosting",␣
 ↪GradientBoostingClassifier(random_state=1)),
            ]
        ),
    )
)


models.append(
    (
        "XGB",
        Pipeline(
            steps=[
                ("scaler", StandardScaler()),
                ("xgboost", XGBClassifier(random_state=1,␣
 ↪eval_metric='logloss')),
            ]
        ),
    )
)

results = []  # Empty list to store all model's CV scores
names = []   # Empty list to store name of the models
```

```
# loop through all models to get the mean cross validated score
for name, model in models:
    scoring = "recall"
    kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
    cv_result = cross_val_score(estimator=model, X=X_train, y=y_train,␣
 ↪scoring=scoring, cv=kfold)
    results.append(cv_result)
    names.append(name)

    print(f"{name}: {cv_result.mean() * 100}")
```

```
DTREE: 29.41231345523526
BAGGING: 9.92041156907709
RF: 5.483433584260805
ADB: 4.991443653375915
GBM: 4.5173508142136205
XGB: 10.045563146078207
```

[197]:

```
# Plotting boxplots for CV scores of all models defined above
fig = plt.figure(figsize=(15, 7))

fig.suptitle("Algorithm Comparison")
ax = fig.add_subplot(111)

plt.boxplot(results)
ax.set_xticklabels(names);
```



# 5 XGBoost Classifier

## 5.1 Hyperparameter Tuning using RandomizedSearchCV

[198]:
```
%%time

#Creating pipeline
pipe=make_pipeline(StandardScaler(),XGBClassifier(random_state=1,eval_metric='logloss',␣
 ↪n_estimators=50))

#Parameter grid to pass in RandomizedSearchCV
param_grid={'xgbclassifier__n_estimators':np.arange(50,300,50),
            'xgbclassifier__scale_pos_weight':[0,1,2,5,10],
            'xgbclassifier__learning_rate':[0.01,0.1,0.2,0.05],
            'xgbclassifier__gamma':[0,1,3,5],
            'xgbclassifier__subsample':[0.7,0.8,0.9,1],
            'xgbclassifier__max_depth':np.arange(1,10,1),
            'xgbclassifier__reg_lambda':[0,1,2,5,10]}

# Type of scoring used to compare parameter combinations
scorer = metrics.make_scorer(metrics.f1_score)

#Calling RandomizedSearchCV
randomized_cv = RandomizedSearchCV(estimator=pipe,␣
 ↪param_distributions=param_grid, n_iter=50,
                                   scoring=scorer, cv=5, random_state=1,␣
 ↪n_jobs=-1)

#Fitting parameters in RandomizedSearchCV
randomized_cv.fit(X_train,y_train)

print(f"Best Parameters:{randomized_cv.best_params_} \nScore: {randomized_cv.
 ↪best_score_}")
```

```
Best Parameters:{'xgbclassifier__subsample': 0.9,
'xgbclassifier__scale_pos_weight': 5, 'xgbclassifier__reg_lambda': 0,
'xgbclassifier__n_estimators': 100, 'xgbclassifier__max_depth': 3,
'xgbclassifier__learning_rate': 0.2, 'xgbclassifier__gamma': 1}
Score: 0.4191839303895996
CPU times: total: 15.6 s
Wall time: 2min 20s
```

[200]:
```
# Creating new pipeline with best parameters
xgb_tuned = make_pipeline(
    StandardScaler(),
    XGBClassifier(
        random_state=1,
        n_estimators=150,
```

```
        scale_pos_weight=2,
        reg_lambda=2,
        max_depth=7,
        subsample=1,
        learning_rate=0.1,
        gamma=0,
        eval_metric='logloss',
        n_jobs=-1
    )
)

# Fit the model on training data
xgb_tuned.fit(X_train, y_train)

#Calculating different metrics
get_metrics_score(xgb_tuned)
```

```
Metric  Train Accuracy  Test Accuracy  Train Recall  Test Recall  \
Score            0.844          0.778         0.499        0.301

Metric  Train Precision  Test Precision  Train F1-Score  Test F1-Score
Score             0.641           0.422           0.561          0.351
```

[200]:
```
[0.844125,
 0.7780416666666666,
 0.49932909920386437,
 0.300772281360885,
 0.6405783796190039,
 0.42159157401989467,
 0.5612024330166391,
 0.3510780850286272]
```

# 6 Decision Tree Classifier

## 6.1 Hyperparameter Tuning using GridSearchCV

[201]:
```
%%time

# Creating pipeline
pipe = make_pipeline(StandardScaler(), DecisionTreeClassifier(random_state=1))

# Parameter grid to pass in GridSearchCV
param_grid = {
    'decisiontreeclassifier__max_depth': np.arange(2, 30),
    'decisiontreeclassifier__min_samples_leaf': [1, 2, 5, 7, 10],
    'decisiontreeclassifier__max_leaf_nodes' : [2, 3, 5, 10, 15],
    'decisiontreeclassifier__min_impurity_decrease': [0.0001,0.001,0.01,0.1]
```

```
}

# Type of scoring used to compare parameter combinations
scorer = metrics.make_scorer(metrics.f1_score)

# Calling GridSearchCV
grid_cv = GridSearchCV(estimator=pipe, param_grid=param_grid, scoring=scorer,␣
↪cv=5, n_jobs=-1)

# Fitting parameters in GridSeachCV
grid_cv.fit(X_train, y_train)

print(f"Best parameters are {grid_cv.best_params_} \nScore={grid_cv.
↪best_score_}:")
```

```
Best parameters are {'decisiontreeclassifier__max_depth': 2,
'decisiontreeclassifier__max_leaf_nodes': 2,
'decisiontreeclassifier__min_impurity_decrease': 0.0001,
'decisiontreeclassifier__min_samples_leaf': 1}
Score=0.0:
CPU times: total: 6min 19s
Wall time: 23min 15s
```

[202]:
```
# Creating new pipeline with best parameters
dtree_tuned = make_pipeline(
    StandardScaler(),
    DecisionTreeClassifier(
        max_depth=7,
        max_leaf_nodes=15,
        random_state=1,
        min_impurity_decrease=0.0001,
        min_samples_leaf=1,
    )
)

# Fit the model on training data
dtree_tuned.fit(X_train, y_train)

#Calculating different metrics
get_metrics_score(dtree_tuned)
```

```
Metric  Train Accuracy  Test Accuracy  Train Recall  Test Recall  \
Score            0.801          0.800         0.003        0.003

Metric  Train Precision  Test Precision  Train F1-Score  Test F1-Score
Score             0.682           0.571           0.005          0.005
```

[202]:
```
[0.8006607142857143,
 0.8005,
 0.0026836031845424457,
 0.002504696305572949,
 0.6818181818181818,
 0.5714285714285714,
 0.005346164127238707,
 0.004987531172069825]
```

# 7 Test Set Prediction

[327]:
```python
test = pd.read_csv('test_loan_data (1).csv')
test.columns
```

[327]:
```
Index(['addr_state', 'annual_inc', 'earliest_cr_line', 'emp_length',
       'emp_title', 'fico_range_high', 'fico_range_low', 'grade',
       'home_ownership', 'application_type', 'initial_list_status', 'int_rate',
       'loan_amnt', 'num_actv_bc_tl', 'mort_acc', 'tot_cur_bal', 'open_acc',
       'pub_rec', 'pub_rec_bankruptcies', 'purpose', 'revol_bal', 'revol_util',
       'sub_grade', 'term', 'title', 'total_acc', 'verification_status'],
      dtype='object')
```

[328]:
```python
test.head()
```

[328]:
```
  addr_state  annual_inc earliest_cr_line emp_length  \
0         MO   50000.000         May-2012     1 year
1         HI   92000.000         Dec-2001  10+ years
2         TX   89000.000         Mar-1989  10+ years
3         CA   33000.000         Nov-2004    9 years
4         MI   35580.000         Feb-1997        NaN

                          emp_title  fico_range_high  fico_range_low grade  \
0                  Tower technician          719.000         715.000     C
1                        Supervisor          684.000         680.000     B
2            APPLICATIONS PROGRAMMER          679.000         675.000     B
3  San Diego Unified School District          674.000         670.000     C
4                               NaN          704.000         700.000     B

  home_ownership application_type initial_list_status  int_rate  loan_amnt  \
0            OWN       Individual                   f    13.990   5000.000
1           RENT       Individual                   f    10.990  30000.000
2       MORTGAGE       Individual                   w    10.150  16000.000
3           RENT       Individual                   f    13.680  10000.000
4       MORTGAGE       Individual                   f    14.090   4000.000

   num_actv_bc_tl  mort_acc  tot_cur_bal  open_acc  pub_rec  \
0           1.000     0.000    33395.000     9.000    0.000
```

```
1           2.000     2.000   229832.000     11.000     0.000
2           5.000     2.000   181616.000     15.000     0.000
3           6.000     0.000    30603.000     12.000     1.000
4           2.000     4.000   124597.000      8.000     0.000

   pub_rec_bankruptcies             purpose  revol_bal  revol_util sub_grade  \
0                 0.000  debt_consolidation   2568.000       9.800        C4
1                 0.000  debt_consolidation  30394.000      75.400        B2
2                 0.000         credit_card  38400.000      75.300        B2
3                 1.000  debt_consolidation  21224.000      69.400        C1
4                 0.000  debt_consolidation   3471.000      39.400        B5

        term                  title  total_acc verification_status
0  36 months     Debt consolidation     11.000     Source Verified
1  36 months     Debt consolidation     35.000     Source Verified
2  60 months  Credit card refinancing     41.000        Not Verified
3  36 months          Breathing Room     16.000        Not Verified
4  36 months       debitconsolidation     19.000            Verified
```

[329]:
```python
test2 = test.copy()
```

[330]:
```python
addr_state = {'AK':0,  'AL':1,  'AR':2,  'AZ':3,  'CA':4 , 'CO':5,  'CT':6,
 'DC':7,  'DE':8,
              'FL':9,  'GA':10, 'HI':11, 'IA':12, 'ID':13, 'IL':14, 'IN':15,
 'KS':16, 'KY':17,
              'LA':18, 'MA':19, 'MD':20, 'ME':21, 'MI':22, 'MN':23, 'MO':24,
 'MS':25, 'MT':26,
              'NC':27, 'ND':28, 'NE':29, 'NH':30, 'NJ':31, 'NM':32, 'NV':33,
 'NY':34, 'OH':35,
              'OK':36, 'OR':37, 'PA':38, 'RI':39, 'SC':40, 'SD':41, 'TN':42,
 'TX':43, 'UT':44,
              'VA':45, 'VT':46, 'WA':47, 'WI':48, 'WV':49, 'WY':50}
test2['addr_state'] = test2['addr_state'].map(addr_state).astype('Int32')
```

[331]:
```python
print(test2.addr_state)
```

```
0        24
1        11
2        43
3         4
4        22
         ..
19995    24
19996    45
19997    43
19998     9
19999     9
```

```
Name: addr_state, Length: 20000, dtype: Int32
```

[332]:
```python
home_ownership = {'MORTGAGE':0, 'RENT':1, 'OWN':2, 'OTHER':3, 'NONE':4}
test2['home_ownership'] = test2['home_ownership'].map(home_ownership).
 ↪astype('Int32')
```

[333]:
```python
print(test2.home_ownership)
```

```
0        2
1        1
2        0
3        1
4        0
        ..
19995    1
19996    0
19997    0
19998    0
19999    0
Name: home_ownership, Length: 20000, dtype: Int32
```

[334]:
```python
# Define the mapping dictionary
emp_length_mapping = {'< 1 year': 0, '1 year': 0, '2 years': 0, '3 years': 0,␣
 ↪'4 years': 0,
                      '5 years': 0, '6 years': 1, '7 years': 1, '8 years': 1,␣
 ↪'9 years': 1,
                      '10+ years': 2, 'NaN': -1}  # Use -1 to represent unknown␣
 ↪or missing values

# Map the values in the DataFrame
test2['emp_length'] = test2['emp_length'].map(emp_length_mapping).
 ↪astype('Int32')
```

[335]:
```python
print(test2.emp_length)
```

```
0           0
1           2
2           2
3           1
4        <NA>
         …
19995       0
19996       0
19997       2
19998       2
19999       0
Name: emp_length, Length: 20000, dtype: Int32
```

[336]:
```
grade = {'A':0, 'B':1, 'C':2, 'D':3, 'E':4, 'F':5, 'G':6}
test2['grade'] = test2['grade'].map(grade).astype('Int32')
```

[337]:
```
print(test2.grade)
```

```
0        2
1        1
2        1
3        2
4        1
        ..
19995    3
19996    3
19997    1
19998    0
19999    4
Name: grade, Length: 20000, dtype: Int32
```

[338]:
```
sub_grade = {'A1':0,  'A2':1,  'A3':2,  'A4':3,  'A5':4,
                 'B1':5,  'B2':6,  'B3':7,  'B4':8,  'B5':9,
                 'C1':10, 'C2':11, 'C3':12, 'C4':13, 'C5':14,
                 'D1':15, 'D2':16, 'D3':17, 'D4':18, 'D5':19,
                 'E1':20, 'E2':21, 'E3':22, 'E4':23, 'E5':24,
                 'F1':25, 'F2':26, 'F3':27, 'F4':28, 'F5':29,
                 'G1':30, 'G2':31, 'G3':32, 'G4':33, 'G5':34}
test2['sub_grade'] = test2['sub_grade'].map(sub_grade).astype('Int32')
```

[339]:
```
print(test2.sub_grade)
```

```
0        13
1         6
2         6
3        10
4         9
         ..
19995    18
19996    18
19997     9
19998     4
19999    20
Name: sub_grade, Length: 20000, dtype: Int32
```

[340]:
```
# Step 1: Remove 'months' from the 'term' column
test2['term'] = test2['term'].str.replace(' months', '')

# Step 2: Convert the column to numeric (int or float)
test2['term'] = pd.to_numeric(test2['term'])
```

[341]:
```python
term_mapping = {36: 0, 60: 1}
test2['term'] = test2['term'].map(term_mapping).astype('Int32')
```

[342]:
```python
tests.head()
```

[342]:
```
   addr_state  annual_inc earliest_cr_line  emp_length  \
0          24   50000.000         May-2012           0
1          11   92000.000         Dec-2001           2
2          43   89000.000         Mar-1989           2
3           4   33000.000         Nov-2004           1
4          22   35580.000         Feb-1997        <NA>

                         emp_title  fico_range_high  fico_range_low  grade  \
0                Tower technician          719.000         715.000      2
1                       Supervisor         684.000         680.000      1
2           APPLICATIONS PROGRAMMER        679.000         675.000      1
3  San Diego Unified School District       674.000         670.000      2
4                              NaN         704.000         700.000      1

   home_ownership application_type initial_list_status  int_rate  loan_amnt  \
0               2       Individual                   f    13.990   5000.000
1               1       Individual                   f    10.990  30000.000
2               0       Individual                   w    10.150  16000.000
3               1       Individual                   f    13.680  10000.000
4               0       Individual                   f    14.090   4000.000

   num_actv_bc_tl  mort_acc  tot_cur_bal  open_acc  pub_rec  \
0           1.000     0.000    33395.000     9.000    0.000
1           2.000     2.000   229832.000    11.000    0.000
2           5.000     2.000   181616.000    15.000    0.000
3           6.000     0.000    30603.000    12.000    1.000
4           2.000     4.000   124597.000     8.000    0.000

   pub_rec_bankruptcies             purpose  revol_bal  revol_util  sub_grade  \
0                 0.000  debt_consolidation   2568.000       9.800         13
1                 0.000  debt_consolidation  30394.000      75.400          6
2                 0.000         credit_card  38400.000      75.300          6
3                 1.000  debt_consolidation  21224.000      69.400         10
4                 0.000  debt_consolidation   3471.000      39.400          9

   term                    title  total_acc  verification_status
0  <NA>        Debt consolidation     11.000                    0
1  <NA>        Debt consolidation     35.000                    0
2  <NA>  Credit card refinancing      41.000                    2
3  <NA>             Breathing Room     16.000                    2
4  <NA>        debitconsolidation     19.000                    1
```

[343]:
```python
for i in tests.select_dtypes(include=['category']).columns:
    print('Unique values in', i, 'are :')
    print(tests[i].value_counts(dropna=False))
    print('*'*50)
```

[344]:
```python
# Define mapping for purpose column
purpose_mapping = {'debt_consolidation': 0,
                   'credit_card': 1,
                   'home_improvement': 2,
                   'other': 3,
                   'major_purchase': 4,
                   'small_business': 5,
                   'medical': 6,
                   'car': 7,
                   'moving': 8,
                   'vacation': 9,
                   'house': 10,
                   'wedding': 11,
                   'renewable_energy': 12,
                   'educational': 13}

# Map the values in the DataFrame
test2['purpose'] = test2['purpose'].map(purpose_mapping).astype('Int32')
```

[345]:
```python
print(test2.purpose)
```

```
0        0
1        0
2        1
3        0
4        0
        ..
19995    0
19996    0
19997    0
19998    7
19999    0
Name: purpose, Length: 20000, dtype: Int32
```

[346]:
```python
# Define mapping for initial_list_status column
initial_list_status_mapping = {'w': 0, 'f': 1}

# Map the values in the DataFrame
test2['initial_list_status'] = test2['initial_list_status'].
 ↪map(initial_list_status_mapping)
```

[347]:
```python
print(test2.initial_list_status)
```

```
0        1
1        1
2        0
3        1
4        1
        ..
19995    0
19996    1
19997    1
19998    0
19999    1
Name: initial_list_status, Length: 20000, dtype: int64
```

[348]:
```python
# Define mapping for application_type column
application_type_mapping = {'Individual': 0,
                            'Joint App': 1}

# Map the values in the DataFrame
test2['application_type'] = test2['application_type'].
  ↪map(application_type_mapping).astype('Int32')
```

[349]:
```python
test2.head(10)
```

[349]:
```
   addr_state  annual_inc earliest_cr_line  emp_length  \
0          24   50000.000         May-2012           0
1          11   92000.000         Dec-2001           2
2          43   89000.000         Mar-1989           2
3           4   33000.000         Nov-2004           1
4          22   35580.000         Feb-1997        <NA>
5          24   32510.000         Aug-2000           2
6          31   38000.000         Mar-2006           0
7           9   45000.000         Aug-1991           2
8           4   50000.000         Sep-1998           0
9          10   67000.000         Nov-1993        <NA>

                        emp_title  fico_range_high  fico_range_low  grade  \
0               Tower technician          719.000         715.000      2
1                     Supervisor          684.000         680.000      1
2         APPLICATIONS PROGRAMMER          679.000         675.000      1
3  San Diego Unified School District          674.000         670.000      2
4                            NaN          704.000         700.000      1
5           Order processing tech          724.000         720.000      1
6              Script Coordinator          814.000         810.000      0
7               Ruffe Systems Inc          674.000         670.000      2
8                  Member/Manager          684.000         680.000      3
9                            NaN          744.000         740.000      1
```

| | home_ownership | application_type | initial_list_status | int_rate | loan_amnt |
|---|---|---|---|---|---|
| 0 | 2 | 0 | 1 | 13.990 | 5000.000 |
| 1 | 1 | 0 | 1 | 10.990 | 30000.000 |
| 2 | 0 | 0 | 0 | 10.150 | 16000.000 |
| 3 | 1 | 0 | 1 | 13.680 | 10000.000 |
| 4 | 0 | 0 | 1 | 14.090 | 4000.000 |
| 5 | 0 | 0 | 0 | 9.170 | 14950.000 |
| 6 | 1 | 0 | 1 | 6.720 | 2800.000 |
| 7 | 1 | 0 | 0 | 16.290 | 19750.000 |
| 8 | 1 | 0 | 0 | 16.990 | 9675.000 |
| 9 | 0 | 0 | 0 | 10.420 | 5000.000 |

| | num_actv_bc_tl | mort_acc | tot_cur_bal | open_acc | pub_rec |
|---|---|---|---|---|---|
| 0 | 1.000 | 0.000 | 33395.000 | 9.000 | 0.000 |
| 1 | 2.000 | 2.000 | 229832.000 | 11.000 | 0.000 |
| 2 | 5.000 | 2.000 | 181616.000 | 15.000 | 0.000 |
| 3 | 6.000 | 0.000 | 30603.000 | 12.000 | 1.000 |
| 4 | 2.000 | 4.000 | 124597.000 | 8.000 | 0.000 |
| 5 | 5.000 | 0.000 | 15111.000 | 15.000 | 0.000 |
| 6 | 1.000 | 0.000 | 15216.000 | 12.000 | 0.000 |
| 7 | 8.000 | 0.000 | 47322.000 | 14.000 | 1.000 |
| 8 | 6.000 | 0.000 | 33271.000 | 12.000 | 2.000 |
| 9 | 5.000 | 7.000 | 288613.000 | 14.000 | 0.000 |

| | pub_rec_bankruptcies | purpose | revol_bal | revol_util | sub_grade | term |
|---|---|---|---|---|---|---|
| 0 | 0.000 | 0 | 2568.000 | 9.800 | 13 | 0 |
| 1 | 0.000 | 0 | 30394.000 | 75.400 | 6 | 0 |
| 2 | 0.000 | 1 | 38400.000 | 75.300 | 6 | 1 |
| 3 | 1.000 | 0 | 21224.000 | 69.400 | 10 | 0 |
| 4 | 0.000 | 0 | 3471.000 | 39.400 | 9 | 0 |
| 5 | 0.000 | 0 | 15111.000 | 41.400 | 6 | 0 |
| 6 | 0.000 | 0 | 651.000 | 1.800 | 2 | 0 |
| 7 | 1.000 | 1 | 15643.000 | 72.800 | 13 | 0 |
| 8 | 1.000 | 0 | 9048.000 | 45.000 | 17 | 0 |
| 9 | 0.000 | 0 | 8149.000 | 10.300 | 7 | 0 |

| | title | total_acc | verification_status |
|---|---|---|---|
| 0 | Debt consolidation | 11.000 | Source Verified |
| 1 | Debt consolidation | 35.000 | Source Verified |
| 2 | Credit card refinancing | 41.000 | Not Verified |
| 3 | Breathing Room | 16.000 | Not Verified |
| 4 | debitconsolidation | 19.000 | Verified |
| 5 | Debt consolidation | 25.000 | Not Verified |
| 6 | Debt consolidation | 16.000 | Not Verified |
| 7 | Credit Card Payoff | 25.000 | Verified |
| 8 | Debt consolidation | 13.000 | Source Verified |
| 9 | Debt consolidation | 54.000 | Not Verified |

[350]:
```
# Define mapping for verification_status column
verification_status_mapping = {
    'Source Verified': 0,
    'Verified': 1,
    'Not Verified': 2
}

# Map the values in the DataFrame
tests['verification_status'] = tests['verification_status'].
 ↪map(verification_status_mapping).astype('Int32')
```

[351]:
```
print(test2.verification_status)
```

```
0        Source Verified
1        Source Verified
2           Not Verified
3           Not Verified
4               Verified
              …
19995       Not Verified
19996    Source Verified
19997    Source Verified
19998       Not Verified
19999           Verified
Name: verification_status, Length: 20000, dtype: object
```

[352]:
```
# Check for null values in each column
null_counts = test2.isnull().sum()

# Print the null counts
print(null_counts)
```

```
addr_state                0
annual_inc                0
earliest_cr_line          0
emp_length             1258
emp_title              1378
fico_range_high           0
fico_range_low            0
grade                     0
home_ownership            2
application_type          0
initial_list_status       0
int_rate                  0
loan_amnt                 0
num_actv_bc_tl         1011
mort_acc                704
tot_cur_bal            1011
```

```
open_acc                  0
pub_rec                   0
pub_rec_bankruptcies     11
purpose                   0
revol_bal                 0
revol_util               13
sub_grade                 0
term                      0
title                   247
total_acc                 0
verification_status       0
dtype: int64
```

[353]:
```
test2['num_actv_bc_tl'].fillna(test2['num_actv_bc_tl'].mean(), inplace=True)
test2['mort_acc'].fillna(test2['mort_acc'].mean(), inplace=True)
test2['tot_cur_bal'].fillna(test2['tot_cur_bal'].mean(), inplace=True)
test2['emp_length'].fillna(0, inplace=True)
revol_util_mean = test2['revol_util'].mean()
test2['revol_util'].fillna(revol_util_mean, inplace=True)
```

[354]:
```
test2.isnull().sum()
```

[354]:
```
addr_state                0
annual_inc                0
earliest_cr_line          0
emp_length                0
emp_title              1378
fico_range_high           0
fico_range_low            0
grade                     0
home_ownership            2
application_type          0
initial_list_status       0
int_rate                  0
loan_amnt                 0
num_actv_bc_tl            0
mort_acc                  0
tot_cur_bal               0
open_acc                  0
pub_rec                   0
pub_rec_bankruptcies     11
purpose                   0
revol_bal                 0
revol_util                0
sub_grade                 0
term                      0
title                   247
```

```
total_acc                0
verification_status      0
dtype: int64
```

[355]:
```
# Replace null values in 'home_ownership' with 3
test2['home_ownership'] = test2['home_ownership'].fillna(3)
```

[356]:
```
# Drop specified columns
test2.drop(['emp_title', 'title', 'earliest_cr_line', 'pub_rec_bankruptcies'],␣
 ↪axis=1, inplace=True)
```

[357]:
```
imputer = KNNImputer(n_neighbors=5)
```

[358]:
```
print(test2.columns)
test2.shape
```

```
Index(['addr_state', 'annual_inc', 'emp_length', 'fico_range_high',
       'fico_range_low', 'grade', 'home_ownership', 'application_type',
       'initial_list_status', 'int_rate', 'loan_amnt', 'num_actv_bc_tl',
       'mort_acc', 'tot_cur_bal', 'open_acc', 'pub_rec', 'purpose',
       'revol_bal', 'revol_util', 'sub_grade', 'term', 'total_acc',
       'verification_status'],
      dtype='object')
```

[358]:
```
(20000, 23)
```

[359]:
```
test2.head()
```

[359]:
```
   addr_state  annual_inc  emp_length  fico_range_high  fico_range_low  grade  \
0          24   50000.000           0          719.000         715.000      2
1          11   92000.000           2          684.000         680.000      1
2          43   89000.000           2          679.000         675.000      1
3           4   33000.000           1          674.000         670.000      2
4          22   35580.000           0          704.000         700.000      1

   home_ownership  application_type  initial_list_status  int_rate  loan_amnt  \
0               2                 0                    1    13.990   5000.000
1               1                 0                    1    10.990  30000.000
2               0                 0                    0    10.150  16000.000
3               1                 0                    1    13.680  10000.000
4               0                 0                    1    14.090   4000.000

   num_actv_bc_tl  mort_acc  tot_cur_bal  open_acc  pub_rec  purpose  \
0           1.000     0.000    33395.000     9.000    0.000        0
1           2.000     2.000   229832.000    11.000    0.000        0
2           5.000     2.000   181616.000    15.000    0.000        1
3           6.000     0.000    30603.000    12.000    1.000        0
4           2.000     4.000   124597.000     8.000    0.000        0
```

```
   revol_bal  revol_util  sub_grade  term  total_acc verification_status
0   2568.000       9.800         13     0     11.000     Source Verified
1  30394.000      75.400          6     0     35.000     Source Verified
2  38400.000      75.300          6     1     41.000        Not Verified
3  21224.000      69.400         10     0     16.000        Not Verified
4   3471.000      39.400          9     0     19.000            Verified
```

[360]:
```python
# Define mapping for verification_status column
verification_status_mapping = {
    'Source Verified': 0,
    'Verified': 1,
    'Not Verified': 2
}

# Map the values in the DataFrame
test2['verification_status'] = test2['verification_status'].
 ↪map(verification_status_mapping).astype('Int32')
```

[361]:
```python
test2.head()
```

[361]:
```
   addr_state  annual_inc  emp_length  fico_range_high  fico_range_low  grade  \
0          24   50000.000           0          719.000         715.000      2
1          11   92000.000           2          684.000         680.000      1
2          43   89000.000           2          679.000         675.000      1
3           4   33000.000           1          674.000         670.000      2
4          22   35580.000           0          704.000         700.000      1

   home_ownership  application_type  initial_list_status  int_rate  loan_amnt  \
0               2                 0                    1    13.990   5000.000
1               1                 0                    1    10.990  30000.000
2               0                 0                    0    10.150  16000.000
3               1                 0                    1    13.680  10000.000
4               0                 0                    1    14.090   4000.000

   num_actv_bc_tl  mort_acc  tot_cur_bal  open_acc  pub_rec  purpose  \
0           1.000     0.000    33395.000     9.000    0.000        0
1           2.000     2.000   229832.000    11.000    0.000        0
2           5.000     2.000   181616.000    15.000    0.000        1
3           6.000     0.000    30603.000    12.000    1.000        0
4           2.000     4.000   124597.000     8.000    0.000        0

   revol_bal  revol_util  sub_grade  term  total_acc  verification_status
0   2568.000       9.800         13     0     11.000                    0
1  30394.000      75.400          6     0     35.000                    0
2  38400.000      75.300          6     1     41.000                    2
3  21224.000      69.400         10     0     16.000                    2
```

```
4   3471.000      39.400          9      0      19.000                    1
```

[367]:
```python
test2.drop(columns=['addr_state'], inplace=True)
```

[368]:
```python
test2.head()
```

[368]:
```
   annual_inc  emp_length  fico_range_high  fico_range_low  grade  \
0   50000.000           0          719.000         715.000      2
1   92000.000           2          684.000         680.000      1
2   89000.000           2          679.000         675.000      1
3   33000.000           1          674.000         670.000      2
4   35580.000           0          704.000         700.000      1

   home_ownership  application_type  initial_list_status  int_rate  loan_amnt  \
0               2                 0                    1    13.990   5000.000
1               1                 0                    1    10.990  30000.000
2               0                 0                    0    10.150  16000.000
3               1                 0                    1    13.680  10000.000
4               0                 0                    1    14.090   4000.000

   num_actv_bc_tl  mort_acc  tot_cur_bal  open_acc  pub_rec  purpose  \
0           1.000     0.000    33395.000     9.000    0.000        0
1           2.000     2.000   229832.000    11.000    0.000        0
2           5.000     2.000   181616.000    15.000    0.000        1
3           6.000     0.000    30603.000    12.000    1.000        0
4           2.000     4.000   124597.000     8.000    0.000        0

   revol_bal  revol_util  sub_grade  term  total_acc  verification_status
0   2568.000       9.800         13     0     11.000                    0
1  30394.000      75.400          6     0     35.000                    0
2  38400.000      75.300          6     1     41.000                    2
3  21224.000      69.400         10     0     16.000                    2
4   3471.000      39.400          9     0     19.000                    1
```

[372]:
```python
test2 = pd.DataFrame(imputer.fit_transform(test2), columns=test2.columns)
test2.head()
```

[372]:
```
   annual_inc  emp_length  fico_range_high  fico_range_low  grade  \
0   50000.000       0.000          719.000         715.000  2.000
1   92000.000       2.000          684.000         680.000  1.000
2   89000.000       2.000          679.000         675.000  1.000
3   33000.000       1.000          674.000         670.000  2.000
4   35580.000       0.000          704.000         700.000  1.000

   home_ownership  application_type  initial_list_status  int_rate  loan_amnt  \
0           2.000             0.000                1.000    13.990   5000.000
1           1.000             0.000                1.000    10.990  30000.000
```

```
2          0.000            0.000                0.000    10.150  16000.000
3          1.000            0.000                1.000    13.680  10000.000
4          0.000            0.000                1.000    14.090   4000.000

   num_actv_bc_tl  mort_acc  tot_cur_bal  open_acc  pub_rec  purpose  \
0           1.000     0.000    33395.000     9.000    0.000    0.000
1           2.000     2.000   229832.000    11.000    0.000    0.000
2           5.000     2.000   181616.000    15.000    0.000    1.000
3           6.000     0.000    30603.000    12.000    1.000    0.000
4           2.000     4.000   124597.000     8.000    0.000    0.000

   revol_bal  revol_util  sub_grade  term  total_acc  verification_status
0   2568.000       9.800     13.000 0.000     11.000                0.000
1  30394.000      75.400      6.000 0.000     35.000                0.000
2  38400.000      75.300      6.000 1.000     41.000                2.000
3  21224.000      69.400     10.000 0.000     16.000                2.000
4   3471.000      39.400      9.000 0.000     19.000                1.000
```

[374]:

```python
#Checking that no column has missing values in train or test sets
print(test2.isna().sum())
```

```
annual_inc             0
emp_length             0
fico_range_high        0
fico_range_low         0
grade                  0
home_ownership         0
application_type       0
initial_list_status    0
int_rate               0
loan_amnt              0
num_actv_bc_tl         0
mort_acc               0
tot_cur_bal            0
open_acc               0
pub_rec                0
purpose                0
revol_bal              0
revol_util             0
sub_grade              0
term                   0
total_acc              0
verification_status    0
dtype: int64
```

[375]:

```python
## Function to inverse the encoding
def test_inverse_mapping(x, y):
    inv_dict = {v: k for k, v in x.items()}
```

```
    test2[y] = np.round(test[y]).map(inv_dict).astype('category')
```

[376]:
```
import numpy as np

def inverse_mapping(x, y):
    # Create a mapping from numerical values to original categories
    inv_dict = {v: k for k, v in x.items()}

    # Convert the categorical column to numerical
    X_train[y] = X_train[y].astype(float)
    X_test[y] = X_test[y].astype(float)

    # Round the numerical values
    X_train[y] = np.round(X_train[y])
    X_test[y] = np.round(X_test[y])

    # Map the rounded numerical values back to original categories
    X_train[y] = X_train[y].map(inv_dict).astype('category')
    X_test[y] = X_test[y].map(inv_dict).astype('category')
```

[377]:
```
import numpy as np

def inverse_mapping(x, y):
    # Create a mapping from numerical values to original categories
    inv_dict = {v: k for k, v in x.items()}

    # Preprocess the column to convert string values to numerical
    if y == 'emp_length':
        X_train[y] = X_train[y].replace({'< 1 year': 0, '10+ years': 10}).
 ↪astype(float)
        X_test[y] = X_test[y].replace({'< 1 year': 0, '10+ years': 10}).
 ↪astype(float)
    elif y == 'term':
        X_train[y] = X_train[y].replace({'36 months': 0, '60 months': 1}).
 ↪astype(float)
        X_test[y] = X_test[y].replace({'36 months': 0, '60 months': 1}).
 ↪astype(float)
    elif y in ['grade', 'sub_grade', 'home_ownership', 'verification_status',␣
 ↪'purpose', 'application_type']:
        X_train[y] = X_train[y].astype(float)
        X_test[y] = X_test[y].astype(float)

    # Round the numerical values
    X_train[y] = np.round(X_train[y])
    X_test[y] = np.round(X_test[y])

    # Map the rounded numerical values back to original categories
```

```
    X_train[y] = X_train[y].map(inv_dict).astype('category')
    X_test[y] = X_test[y].map(inv_dict).astype('category')
```

[379]:

```
cols = test.select_dtypes(include=['object','category'])
for i in cols.columns:
    print(test[i].value_counts(dropna=False))
    print('*'*30)
```

```
earliest_cr_line
Oct-2001    160
Sep-2004    143
Aug-2001    142
Aug-2000    136
Sep-2003    136
           …
Mar-1968      1
Nov-1972      1
Jun-1963      1
Jul-1973      1
Dec-1959      1
Name: count, Length: 568, dtype: int64
******************************
emp_length
10+ years    6579
2 years      1810
< 1 year     1583
3 years      1580
1 year       1336
NaN          1258
5 years      1228
4 years      1190
6 years       957
7 years       874
8 years       836
9 years       769
Name: count, dtype: int64
******************************
emp_title
NaN                                  1378
Teacher                               357
Manager                               240
Registered Nurse                      142
Owner                                 142
                                    …
Project Production Coordinator          1
los angeles school district             1
claims adjuster                         1
Graduate Research Assistant             1
```

```
rv technician                          1
Name: count, Length: 11181, dtype: int64
******************************
grade
B    5756
C    5704
A    3495
D    3042
E    1418
F     467
G     118
Name: count, dtype: int64
******************************
home_ownership
MORTGAGE    9900
RENT        7917
OWN         2181
ANY            2
Name: count, dtype: int64
******************************
application_type
Individual    19610
Joint App       390
Name: count, dtype: int64
******************************
initial_list_status
w    11582
f     8418
Name: count, dtype: int64
******************************
purpose
debt_consolidation    11611
credit_card            4304
home_improvement       1288
other                  1217
major_purchase          434
medical                 238
small_business          227
car                     220
moving                  151
vacation                150
house                    99
wedding                  43
renewable_energy         13
educational               5
Name: count, dtype: int64
******************************
sub_grade
```

```
C1    1294
B4    1218
B5    1197
B3    1168
C4    1157
C2    1149
B2    1116
C3    1109
B1    1057
C5     995
A5     936
A4     794
D1     757
D2     647
A1     639
D3     609
A3     576
A2     550
D4     546
D5     483
E1     377
E2     304
E3     277
E4     237
E5     223
F1     154
F2      94
F3      81
F4      71
F5      67
G1      39
G2      25
G3      22
G4      17
G5      15
Name: count, dtype: int64
******************************
term
 36 months    15209
 60 months     4791
Name: count, dtype: int64
******************************
title
Debt consolidation            9855
Credit card refinancing       3645
Home improvement              1106
Other                         1043
Major purchase                 370
```

```
                                        …
Credit Card repayment           1
Eliminate Debt                  1
Debt Consollidation             1
DEPT DESTROYER                  1
credit card refinance loan      1
Name: count, Length: 1624, dtype: int64
******************************
verification_status
Source Verified    7722
Verified           6166
Not Verified       6112
Name: count, dtype: int64
******************************
```

[380]:
```
test2 = pd.get_dummies(test2, drop_first=True)
test2.shape
```

[380]:
```
(20000, 22)
```

[381]:
```
test2.columns
```

[381]:
```
Index(['annual_inc', 'emp_length', 'fico_range_high', 'fico_range_low',
       'grade', 'home_ownership', 'application_type', 'initial_list_status',
       'int_rate', 'loan_amnt', 'num_actv_bc_tl', 'mort_acc', 'tot_cur_bal',
       'open_acc', 'pub_rec', 'purpose', 'revol_bal', 'revol_util',
       'sub_grade', 'term', 'total_acc', 'verification_status'],
      dtype='object')
```

pred = xgb_tuned.predict(test) print(f"Prediction has length: {len(pred)}") Prediction has length: 20000

[ ]:
```
submit_df.to_csv('Final_submission.csv', index=False)
```