Data Wrangling

August 26, 2023

**

Data Wrangling**

Import Packages

[99]:
```
import os
from dotenv import load_dotenv

import datetime as dt

from tqdm import tqdm

import math
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV

from statsmodels.stats.outliers_influence import variance_inflation_factor
```

Set Correct Options for the Notebook

[100]:
```
# Load in the dotenv variables
load_dotenv()
# Show all the columns in the .head() method
pd.set_option('display.max_columns', None)
# Turn all plots dark to help my eyes
plt.style.use('dark_background')
```

The data was taken from kaggle and contains Loan Data information on the borrowers. Below I used Kaggle's API to download the dataset. I had trouble getting it to work inline in the notebook so I used these commands in the terminal.

[101]:
```
'''
# Download the data from the kaggle api
kaggle datasets download -d ranadeep/credit-risk-dataset
# unzip the data
unzip credit-risk-dataset.zip -d credit-risk-dataset
'''
```

[101]:
```
'\n# Download the data from the kaggle api\nkaggle datasets download -d
ranadeep/credit-risk-dataset\n# unzip the data \nunzip credit-risk-dataset.zip
-d credit-risk-dataset \n'
```

Set Directory

[102]:
```
# Get the path from the environment from the dotenv file without extra add-ons
project_path = os.getenv('Project_Path')[2:78]

# Change notebook directory back one so that it can acess the data
os.chdir(project_path)
```

Load the Data

[103]:
```
# Load in the data with low_memory as False so that it can understand dtype
loan = pd.read_csv('./data/raw/loan.csv', low_memory = False)
# Print out the shape of the dataset
print(loan.shape)
# Print out a sample of the dataset
loan.head()
```

```
(887379, 74)
```

[103]:
```
        id  member_id  loan_amnt  funded_amnt  funded_amnt_inv        term  \
0  1077501    1296599     5000.0       5000.0           4975.0   36 months
1  1077430    1314167     2500.0       2500.0           2500.0   60 months
2  1077175    1313524     2400.0       2400.0           2400.0   36 months
3  1076863    1277178    10000.0      10000.0          10000.0   36 months
4  1075358    1311748     3000.0       3000.0           3000.0   60 months

   int_rate  installment grade sub_grade                 emp_title emp_length  \
0     10.65       162.87     B        B2                       NaN  10+ years
1     15.27        59.83     C        C4                     Ryder   < 1 year
2     15.96        84.33     C        C5                       NaN  10+ years
3     13.49       339.31     C        C1       AIR RESOURCES BOARD  10+ years
4     12.69        67.79     B        B5  University Medical Group     1 year

  home_ownership  annual_inc verification_status   issue_d  loan_status  \
0           RENT     24000.0            Verified  Dec-2011   Fully Paid
1           RENT     30000.0     Source Verified  Dec-2011  Charged Off
2           RENT     12252.0        Not Verified  Dec-2011   Fully Paid
3           RENT     49200.0     Source Verified  Dec-2011   Fully Paid
```

```
4          RENT     80000.0     Source Verified  Dec-2011      Current

  pymnt_plan                                                url  \
0          n  https://www.lendingclub.com/browse/loanDetail…
1          n  https://www.lendingclub.com/browse/loanDetail…
2          n  https://www.lendingclub.com/browse/loanDetail…
3          n  https://www.lendingclub.com/browse/loanDetail…
4          n  https://www.lendingclub.com/browse/loanDetail…

                                                desc         purpose  \
0   Borrower added on 12/22/11 > I need to upgra…     credit_card
1   Borrower added on 12/22/11 > I plan to use t…             car
2                                                NaN  small_business
3   Borrower added on 12/21/11 > to pay for prop…           other
4   Borrower added on 12/21/11 > I plan on combi…           other

                 title zip_code addr_state    dti  delinq_2yrs  \
0             Computer    860xx         AZ  27.65          0.0
1                 bike    309xx         GA   1.00          0.0
2  real estate business    606xx         IL   8.72          0.0
3             personel    917xx         CA  20.00          0.0
4             Personal    972xx         OR  17.94          0.0

  earliest_cr_line  inq_last_6mths  mths_since_last_delinq  \
0         Jan-1985             1.0                     NaN
1         Apr-1999             5.0                     NaN
2         Nov-2001             2.0                     NaN
3         Feb-1996             1.0                    35.0
4         Jan-1996             0.0                    38.0

   mths_since_last_record  open_acc  pub_rec  revol_bal  revol_util  \
0                     NaN       3.0      0.0    13648.0        83.7
1                     NaN       3.0      0.0     1687.0         9.4
2                     NaN       2.0      0.0     2956.0        98.5
3                     NaN      10.0      0.0     5598.0        21.0
4                     NaN      15.0      0.0    27783.0        53.9

   total_acc initial_list_status  out_prncp  out_prncp_inv   total_pymnt  \
0        9.0                   f        0.0            0.0   5861.071414
1        4.0                   f        0.0            0.0   1008.710000
2       10.0                   f        0.0            0.0   3003.653644
3       37.0                   f        0.0            0.0  12226.302212
4       38.0                   f      766.9          766.9   3242.170000

   total_pymnt_inv  total_rec_prncp  total_rec_int  total_rec_late_fee  \
0          5831.78          5000.00         861.07                0.00
1          1008.71           456.46         435.17                0.00
```

```
2         3003.65          2400.00         603.65                0.00
3        12226.30         10000.00        2209.33               16.97
4         3242.17          2233.10        1009.07                0.00

   recoveries  collection_recovery_fee last_pymnt_d  last_pymnt_amnt  \
0        0.00                     0.00     Jan-2015           171.62
1      117.08                     1.11     Apr-2013           119.66
2        0.00                     0.00     Jun-2014           649.91
3        0.00                     0.00     Jan-2015           357.48
4        0.00                     0.00     Jan-2016            67.79

  next_pymnt_d last_credit_pull_d  collections_12_mths_ex_med  \
0          NaN           Jan-2016                         0.0
1          NaN           Sep-2013                         0.0
2          NaN           Jan-2016                         0.0
3          NaN           Jan-2015                         0.0
4     Feb-2016           Jan-2016                         0.0

   mths_since_last_major_derog  policy_code application_type  \
0                          NaN          1.0       INDIVIDUAL
1                          NaN          1.0       INDIVIDUAL
2                          NaN          1.0       INDIVIDUAL
3                          NaN          1.0       INDIVIDUAL
4                          NaN          1.0       INDIVIDUAL

   annual_inc_joint  dti_joint verification_status_joint  acc_now_delinq  \
0               NaN        NaN                       NaN             0.0
1               NaN        NaN                       NaN             0.0
2               NaN        NaN                       NaN             0.0
3               NaN        NaN                       NaN             0.0
4               NaN        NaN                       NaN             0.0

   tot_coll_amt  tot_cur_bal  open_acc_6m  open_il_6m  open_il_12m  \
0           NaN          NaN          NaN         NaN          NaN
1           NaN          NaN          NaN         NaN          NaN
2           NaN          NaN          NaN         NaN          NaN
3           NaN          NaN          NaN         NaN          NaN
4           NaN          NaN          NaN         NaN          NaN

   open_il_24m  mths_since_rcnt_il  total_bal_il  il_util  open_rv_12m  \
0          NaN                 NaN           NaN      NaN          NaN
1          NaN                 NaN           NaN      NaN          NaN
2          NaN                 NaN           NaN      NaN          NaN
3          NaN                 NaN           NaN      NaN          NaN
4          NaN                 NaN           NaN      NaN          NaN

   open_rv_24m  max_bal_bc  all_util  total_rev_hi_lim  inq_fi  total_cu_tl  \
```

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

   inq_last_12m
0           NaN
1           NaN
2           NaN
3           NaN
4           NaN
```

The data is large and complex. In the `read_csv()` function I had to specify `low_memory = False` because some columns had too many different data types for the `read_csv()` function to be able to quickly determine the data type of each column.

The first step in data wrangling is to separate out the features (X) from the target variable (y). When I initially started this project I had believed that y was `loan_status` because it was listed on the kaggle website, and was a clear representation of how the loan had fared. After much data wrangling and looking closely at the dataset I realized that it was hard to use many of the features while having `loan_status` be the target variable. Too many of the features have information on the loan after origination. If the purpose of the project was to try to find the best loans to invest in, then I cannot use features that are collected during the term of the loan.

In order to use many of the features of the dataset, I need a question that has a later time frame. If I try to find how much each borrower will still pay back on their loans, I can use the features that are during the term of the loan. This can create a market or evaluation for the value of the existing loans that would be valuable for loan investors or institutions who are thinking of selling off their loans for immediate liquidity.

Below I separated out every column of the dataset that had information on the amount that the borrower has already paid and the amount that they owe. With the remaining columns I'll be able to create a multiplier that will provide an estimate on how much the borrower will pay.

[104]:
```
# Create the features dataframe
X = loan.drop(['out_prncp', 'out_prncp_inv', 'total_pymnt', 'total_pymnt_inv',
               'total_rec_prncp', 'total_rec_int',␣
 ↪'total_rec_late_fee','recoveries',
               'collection_recovery_fee', 'loan_status'], axis = 1)
# Separate out the target variable
y = loan[['out_prncp', 'out_prncp_inv', 'total_pymnt', 'total_pymnt_inv',␣
 ↪'total_rec_prncp',
          'total_rec_int', 'total_rec_late_fee','recoveries',␣
 ↪'collection_recovery_fee']]
```

0.1 Cleaning up the Dataset

Unique Values The easiest features to clean are ones that have constant values since they provide no information. Below I looked at how many unique values each feature has. I will drop every feature that has only 1 unique value.

[105]:
```
X.nunique().sort_values(ascending = True)
```

[105]:
```
policy_code                  1
pymnt_plan                   2
application_type             2
initial_list_status          2
term                         2
                          …
emp_title               299271
tot_cur_bal             327342
url                     887379
member_id               887379
id                      887379
Length: 64, dtype: int64
```

[106]:
```
# Drop the 'policy_code' feature since there is only one policy code
X.drop('policy_code', axis = 1, inplace = True)
```

0.1.1 Categorical Columns

In order to analyze the data, all of the features have to have only numerical values. Some columns contain numbers but need to be cleaned in order to be fully numeric where as others are purely categorical and either need to be one-hot encoded or dropped if they have too many unique values.

Below I printed out a list of all of the categorical columns.

[107]:
```
# Use dtypes to find all of the columns that are not int or float in X
[X.dtypes.index[i] for i,type in enumerate(X.dtypes) if type not in␣
 ↪('int64','float64', 'uint8')]
```

[107]:
```
['term',
 'grade',
 'sub_grade',
 'emp_title',
 'emp_length',
 'home_ownership',
 'verification_status',
 'issue_d',
 'pymnt_plan',
 'url',
 'desc',
 'purpose',
 'title',
```

 'zip_code',
 'addr_state',
 'earliest_cr_line',
 'initial_list_status',
 'last_pymnt_d',
 'next_pymnt_d',
 'last_credit_pull_d',
 'application_type',
 'verification_status_joint']

term of the loan

[108]:
```
X['term'].unique()
```

[108]:
```
array([' 36 months', ' 60 months'], dtype=object)
```

The `term` variable represents how long loan is for and has two values: `36 months` and `60 months`. I can convert those strings to 3, and 5 respectively to represent the years.

[109]:
```
# replace the string values with numerical values, use a dictionary to be␣
 ↪concise
X['term'].replace({' 36 months':3, ' 60 months':5}, inplace = True)
# Show the column
X['term']
```

[109]:
```
0         3
1         5
2         3
3         3
4         5
         ..
887374    3
887375    3
887376    5
887377    5
887378    3
Name: term, Length: 887379, dtype: int64
```

grade and sub_grade for the loans

[110]:
```
# Show all the different possible grades
print('Grade:', X['grade'].unique())

# Show all the different possible sub grades
print('Grade:', X['sub_grade'].unique())
```

```
Grade: ['B' 'C' 'A' 'E' 'F' 'D' 'G']
Grade: ['B2' 'C4' 'C5' 'C1' 'B5' 'A4' 'E1' 'F2' 'C3' 'B1' 'D1' 'A1' 'B3' 'B4'
 'C2' 'D2' 'A3' 'A5' 'D5' 'A2' 'E4' 'D3' 'D4' 'F3' 'E3' 'F4' 'F1' 'E5'
 'G4' 'E2' 'G3' 'G2' 'G1' 'F5' 'G5']
```

There is also the subgrade column which differentiates the quality of loans within the different grades. Since there are 5 subgrades for each grade, I can convert the subgrade to .2 downgrade of the grade

[111]:
```
# Convert the grades into numbers that can be analyzed
X['grade'] = X['grade'].replace({'A':7, 'B':6, 'C':5, 'D':4, 'E':3, 'F':2, 'G':
    ↪1})
# Add in the subgrade as a .2 change in the grade
X['grade'] = [X['grade'][i] - ((int(sub[-1]) - 1)/5) for i, sub in␣
    ↪enumerate(X['sub_grade'])]
# Show the grade
X['grade']
```

[111]:
```
0         5.8
1         4.4
2         4.2
3         5.0
4         5.2
         …
887374    5.2
887375    5.2
887376    3.8
887377    2.6
887378    5.2
Name: grade, Length: 887379, dtype: float64
```

With grade incorporating both `grade` and `sub_grade`, I can drop `sub_grade`.

[112]:
```
# Drop sub_grade from X
X.drop('sub_grade', axis = 1, inplace = True)
```

emp_title: The type of employment for the borrower.

[113]:
```
X['emp_title'].unique()
```

[113]:
```
array([nan, 'Ryder', 'AIR RESOURCES BOARD', …, 'machining Cell Lead',
       'KYC Business Analyst', 'Manager Hotel Operations Oasis '],
      dtype=object)
```

Unfortunately it appears that many of the borrowers put the name of their employer instead of their profession. I can confirm this better by using the `value_counts()` method from pandas.

[114]:
```
X['emp_title'].value_counts()
```

[114]:
```
Teacher                                        13469
Manager                                        11240
Registered Nurse                                5525
Owner                                           5376
RN                                              5355
```

```
                                              …
Thomas J. Paul, Inc.                             1
Piggie Toes Preschool                            1
greystone park psychiatric hospital              1
Las Vegas Metropolitan Police Department         1
Manager Hotel Operations Oasis                   1
Name: emp_title, Length: 299271, dtype: int64
```

While the most common values are types of professions, it appears that there are many values that have just the name of the employers. If I take a look at all of the values in `emp_title` that appear over 1000 times I'll get a better sense of the common professions I should try to simplify the column into.

[115]:
```python
# find all the different types of jobs that the borrower could have
print(len(X['emp_title'].value_counts()[X['emp_title'].value_counts() > 1000]))
# See how common each of the different employment types
X['emp_title'].value_counts()[X['emp_title'].value_counts() > 1000]
```

```
48
```

[115]:
```
Teacher                     13469
Manager                     11240
Registered Nurse             5525
Owner                        5376
RN                           5355
Supervisor                   4983
Sales                        4212
Project Manager              3988
Driver                       3569
Office Manager               3510
General Manager              3178
Director                     3156
manager                      3138
teacher                      2925
owner                        2849
Engineer                     2671
President                    2598
driver                       2429
Vice President               2351
Attorney                     2136
Operations Manager           2071
Accountant                   2035
Administrative Assistant     2019
Sales Manager                1846
Account Manager              1725
sales                        1724
Police Officer               1720
supervisor                   1675
```

```
Executive Assistant          1603
Analyst                      1538
Store Manager                1515
Technician                   1462
Nurse                        1426
truck driver                 1387
Truck Driver                 1387
Software Engineer            1344
Paralegal                    1299
Controller                   1246
Consultant                   1242
Assistant Manager            1199
Program Manager              1188
Branch Manager               1177
Server                       1120
Administrator                1103
Principal                    1073
Account Executive            1072
Mechanic                     1044
Business Analyst             1013
Name: emp_title, dtype: int64
```

What stands out in this column is that many of the employment types seem to be variations on the same profession. For instance there are a lot of different kinds of managers even though they are all in the managerial profession. I can combine all of these to create a much more consolidated column.

Below I created a dictionary where the keys are the profession and the values are lists of all phrases that would be associated with that profession.

[116]:
```
Professions = {'Manager':['manag','supervisor','superintendent', ' lead ',␣
↪'foreman'],
               'Healer':['nurse', 'nursing', 'cna', 'lpn', 'physician',␣
↪'doctor', 'pharmacist', 'counselor', 'therapist', 'rn', 'paramedic'],
               'Technical': ['engineer','mechanic', 'electrician', 'machinist',␣
↪'machining', 'technician', 'it', 'software', 'tech','welder'],
               'Executive':['president', 'owner', 'ceo', 'partner', 'vp'],␣
↪'Designer': ['designer'],
               'Vol': ['firefighter', 'social worker', 'army', 'officer',␣
↪'sheriff', 'deputy', 'sergeant', 'agent', 'colonel','USAF'],
               'Director':['director', 'coordinator'], 'Accountant':
↪['accountant', 'bookkeeper', 'controller','accounting'], 'Sales':['sales',␣
↪'realt'],
               'Finance':['financial','underwriter', 'broker', 'cfo','bank'],␣
↪'Analyst':['analyst','consult', 'analysis','estimator'],
               'Clergy':['pastor','rabbi','priest', 'imam', 'minister'],␣
↪'Clerk':['clerk'],
```

```
                'Service':['bartender', 'server', 'service', 'diner', 'grill',␣
↪'attendant', 'cashier','aesthetician', 'dealer', 'cook','chef'],
                'Manlab':['maintenance','maintenence','laborer','lumber',␣
↪'custodian','carpenter'],
                'Operator':['operator','pilot', 'driver'], 'Assistant':
↪['assistant', 'secretary','receptionist'], 'Law':
↪['attorney','esq','paralegal'],
                'Education':
↪['teacher','principal','professor','school','educator'], 'Admin':['admin']}
```

Below I created a new column that assumed that all missing values were unemployed. Then I iterated through the Professions dictionary and for each value I searched through the entire column to see if any cells contained that value. If they did, then I changed the entire cell to be the corresponding key from the dictionary.

[117]:
```python
# Fill in the missing values with unemployed for analysis
X['emp_title_cons'] = X['emp_title'].fillna('Unemployed')
for key, value in tqdm(Professions.items()):
    for v in value:
        # Create a boolean mask for all employments that are managerial
        prof = X['emp_title_cons'].str.contains(v, case=False)
        # Change the value of those to 'Manager'
        X.loc[prof, 'emp_title_cons'] = key
```

```
  0%|          | 0/20 [00:00<?, ?it/s]100%|     | 20/20 [00:32<00:00,
1.65s/it]
```

With the employment titles consolidated, I can look at the employment titles with over 1000 instances again to see how effective the grouping was.

[118]:
```python
# find all the different types of jobs that the borrower could have
print(len(X['emp_title_cons'].value_counts()[X['emp_title_cons'].value_counts()␣
↪> 1000]))

# See how common each of the different employment types
X['emp_title_cons'].value_counts()[X['emp_title_cons'].value_counts() > 1000]
```

```
22
```

[118]:
```
Manager        141239
Technical      119770
Unemployed      51462
Healer          50322
Director        32973
Education       29821
Executive       28111
Vol             25647
Operator        24758
Service         22252
```

```
Analyst                 21481
Assistant               18877
Sales                   17653
Finance                 12007
Accountant              10458
Admin                    9648
Manlab                   5969
Clerk                    5333
Designer                 2466
Law                      1942
Clergy                   1392
Account Executive        1072
Name: emp_title_cons, dtype: int64
```

There is a marked improvement here with "Manager" accounting for over 150,000 of the observations. It is possible that some of the profession types are mistakes but I intentionally put spaces before and after 'lead' so that the code would only capture instances of the word lead itself.

Below I changed the `emp_type` column to have only professions with more than 1000 instances. The rest I classified as other.

[119]:
```
common_proffs = X['emp_title_cons'].value_counts()[X['emp_title_cons'].
    ↪value_counts() > 1000]
# Create a feature that has the most common profession types, store the rest as␣
    ↪other.
X['emp_type'] = [x if x in common_proffs.index else 'other' for x in␣
    ↪X['emp_title_cons']]
```

With a workable list of professions that the borrowers have, I can one-hot encode them for later analysis and drop the columns that created `emp_type`.

[120]:
```
# Convert all of the columns into dummy variables
X = pd.get_dummies(X, columns = ['emp_type'], drop_first = True)

# Drop emp_title and emp_title_cons now that there is no need for them.
X.drop(['emp_title', 'emp_title_cons'], axis = 1, inplace = True)
```

emp_length: employment length of the borrower.

[121]:
```
# Show all of the different employement lengths
X['emp_length'].unique()
```

[121]:
```
array(['10+ years', '< 1 year', '1 year', '3 years', '8 years', '9 years',
       '4 years', '5 years', '6 years', '2 years', '7 years', nan],
      dtype=object)
```

This column appears to have numerical values that are encased in strings. If I can separate the number from the surrounding string and change its type to a number than this column can become numerical.

[122]:
```
# Replace the non-existant values with 0
X['emp_length'] = X['emp_length'].fillna(0)
# Replace the less than one year with the average of .5
X['emp_length'] = X['emp_length'].replace('< 1 year', 0.5)
# Get rid of all non-numerical characters to convert the values into numerical
X['emp_length'] = X['emp_length'].replace(r'[a-zA-Z+]', '', regex = True)
# Change the type from string to numerical
X['emp_length'] = X['emp_length'].astype('float')
```

`home_ownership`: The housing situation of the borrower

[123]:
```
# Look at all the types of home ownership
X['home_ownership'].unique()
```

[123]:
```
array(['RENT', 'OWN', 'MORTGAGE', 'OTHER', 'NONE', 'ANY'], dtype=object)
```

Since `OTHER`, `NONE`, and `ANY` can all mean the same thing, I think it best to combine the three of them into `OTHER`

[124]:
```
# Switch home_ownership rare values to other.
X['home_ownership'].replace({'NONE': 'OTHER','ANY':'OTHER'}, inplace = True)
```

With a consolidated `home_ownership` column, I can now one-hot encode it for later analysis

[125]:
```
# Create dummy columns for the different types of home ownership
X = pd.get_dummies(X, columns = ['home_ownership'], drop_first = True)
```

`verification_status`: Whether or not Lending club was able to verify the information the borrower submitted.

[126]:
```
X['verification_status'].unique()
```

[126]:
```
array(['Verified', 'Source Verified', 'Not Verified'], dtype=object)
```

`verification_status` has 3 unique values that determine if the information has been verified, the source of the information has been verified or if neither the source nor the information has been verified. I will make this into a dummy variable later on.

[127]:
```
# Create dummy variables for the different verification statuses
X = pd.get_dummies(X, columns = ['verification_status'], drop_first = True)
```

`issue_d`: the month and year in which the loan was issued.

[128]:
```
X['issue_d'].head()
```

[128]:
```
0    Dec-2011
1    Dec-2011
2    Dec-2011
3    Dec-2011
4    Dec-2011
Name: issue_d, dtype: object
```

issue_d is a very important feature because I'll be able to use it to determine how much time is left on the loan but it isn't a feature that will influence a borrower's ability to pay back the loan. For that reason I'll drop it from this dataset.

```
[129]: # Drop issue_d from the X dataset
X.drop('issue_d', axis = 1, inplace = True)
```

pymnt_plan: Indicates whether there has been a payment plan put in place for the borrower to pay off the loan that wasn't the original schedule.

```
[130]: X['pymnt_plan'].unique()
```

```
[130]: array(['n', 'y'], dtype=object)
```

There are only two values in this column either a 'n' for no or a 'y' for yes. Since it is a binary choice I can represent it numerically as 0 for n and 1 for y.

```
[131]: # Convert pyment_plan to a binary variable
X['pymnt_plan'] = np.where(X['pymnt_plan'] == 'y', 1, 0)
```

url of the page that contains the loan. Needs a login to access it.

```
[132]: X['url'].unique()
```

```
[132]: array(['https://www.lendingclub.com/browse/loanDetail.action?loan_id=1077501',
        'https://www.lendingclub.com/browse/loanDetail.action?loan_id=1077430',
        'https://www.lendingclub.com/browse/loanDetail.action?loan_id=1077175',
        …,
        'https://www.lendingclub.com/browse/loanDetail.action?loan_id=36271333',
        'https://www.lendingclub.com/browse/loanDetail.action?loan_id=36490806',
        'https://www.lendingclub.com/browse/loanDetail.action?loan_id=36271262'],
       dtype=object)
```

The URL column appears to be the same website with the id column as the loan_id at the end of it. I can test this by trying to see if the characters up until the id= are always the same.

```
[133]: # Grab the first part of the url and see if it always the same.
X['url'].str[:61].unique()
```

```
[133]: array(['https://www.lendingclub.com/browse/loanDetail.action?loan_id='],
       dtype=object)
```

Now that I know that the start of the url is always the same I can check to see if the ID part of the url is the exact same as the id column itself.

```
[134]: # subtract the set of the id part of the url column from the id column.
set(X['id']) - set(X['url'].str[61:].astype('int'))
```

```
[134]: set()
```

The code above returned an empty set which means that there is no difference between the id column and the number at the end of the `url`. This means that the url is redundant and can be dropped.

[135]:
```
# Drop the url column from X
X.drop('url', axis = 1, inplace = True)
```

`desc`: A summary for the reason the borrower is taking the loan

[136]:
```
# Find how many people didn't put a description
print(X['desc'].isna().sum())

# Print a sample of some of the descriptions
X['desc'].unique()
```

```
761351
```

[136]:
```
array(['  Borrower added on 12/22/11 > I need to upgrade my business
technologies.<br>',
       '  Borrower added on 12/22/11 > I plan to use this money to finance the
motorcycle i am looking at. I plan to have it paid off as soon as possible/when
i sell my old bike. I only need this money because the deal im looking at is to
good to pass up.<br><br>  Borrower added on 12/22/11 > I plan to use this money
to finance the motorcycle i am looking at. I plan to have it paid off as soon as
possible/when i sell my old bike.I only need this money because the deal im
looking at is to good to pass up. I have finished college with an associates
degree in business and its takingmeplaces<br>',
       nan, …,
       'I need a lower interest loan to pay off my citifinancial loan.  ',
       'I am looking for a loan to pay my credit cards off as well as making
some very much needed auto repairs',
       'I am in my senior year of college in obtaining a bachelors degree in
criminal justice.  I do not qualify for financial aid and have used all stafford
loans available.  My tuition is approx. $1200 a month and I have 10 courses left
which adds up to over $10,000.  I need some assistance to cover my tuition until
my graduation date which is August of 2009.'],
      dtype=object)
```

The `desc` variable is significantly more in depth than the `title` or `purpose` variable but unfortunately most of the borrowers never submitted one. Since most of the important information that could be gleaned from the `desc` column is most likely contained within `title` and `purpose` which have litle to no missing values, I will drop this column.

[137]:
```
X.drop('desc', axis = 1, inplace = True)
```

`purpose`: The reason the borrower applied and took the loan

[138]:
```
# View all the unique reasons for why people are getting a loan.
X['purpose'].unique()
```

[138]:
```
array(['credit_card', 'car', 'small_business', 'other', 'wedding',
       'debt_consolidation', 'home_improvement', 'major_purchase',
       'medical', 'moving', 'vacation', 'house', 'renewable_energy',
       'educational'], dtype=object)
```

This column needs to be one-hot encoded since there are no clear numerical map for the categorical values of this column. .

[139]:
```
X = pd.get_dummies(X, columns = ['purpose'], drop_first = True)
```

title: The title that the borrower put for the loan

[140]:
```
print(len(X['title'].unique()))
X['title'].unique()
```

```
63145
```

[140]:
```
array(['Computer', 'bike', 'real estate business', …,
       'new kitchen for momma!',
       'New Baby and New House (CC Consolidate)',
       'Credit Card/Auto Repair'], dtype=object)
```

This appears to be a worse version of purpose as there are over 63,000 different titles and the content from them appears to be very similar to the content in the `purpose` variable. For that reason I'll drop `title`

[141]:
```
X.drop('title', axis = 1, inplace = True)
```

zip_code: The borrower's zipcode

[142]:
```
print(len(X['zip_code'].unique()))
X['zip_code'].head()
```

```
935
```

[142]:
```
0    860xx
1    309xx
2    606xx
3    917xx
4    972xx
Name: zip_code, dtype: object
```

While it looks like there are numbers to be used in this column, these are zipcodes and not orders of magnitude. Therefore if I removed the xs I would still have a categorical column. I could one-hot encode the zip codes but that would produce over 800 million values and I'm not sure that my laptop would be able to easily run calculations on that amount of data so the simplest thing to do is to drop them.

[143]:
```
# Drop zip_code from the X dataset
X.drop('zip_code', axis = 1, inplace = True)
```

addr_state: Initials for the state the borrower is from

[144]:
```python
# Print out the number of states that are used
print(len(X['addr_state'].unique()))
# Print out the list of states used.
(X['addr_state'].unique())
```

```
51
```

[144]:
```
array(['AZ', 'GA', 'IL', 'CA', 'OR', 'NC', 'TX', 'VA', 'MO', 'CT', 'UT',
       'FL', 'NY', 'PA', 'MN', 'NJ', 'KY', 'OH', 'SC', 'RI', 'LA', 'MA',
       'WA', 'WI', 'AL', 'CO', 'KS', 'NV', 'AK', 'MD', 'WV', 'VT', 'MI',
       'DC', 'SD', 'NH', 'AR', 'NM', 'MT', 'HI', 'WY', 'OK', 'DE', 'MS',
       'TN', 'IA', 'NE', 'ID', 'IN', 'ME', 'ND'], dtype=object)
```

While there are 50 states, there are borrowers from Washington, D.C. brings the total to 51. Unlike zipcodes, 51 is a much more reasonable number to one-hot encode.

[145]:
```python
# Create a dummy variable for each state that a borrower is from
X = pd.get_dummies(X, columns = ['addr_state'], drop_first = True)
```

earliest_cr_line: The date of the earliest line of credit that the borrower had

[146]:
```python
print(X['earliest_cr_line'].isna().sum())
print(X['earliest_cr_line'].dtype)
X['earliest_cr_line'].head()
```

```
29
object
```

[146]:
```
0    Jan-1985
1    Apr-1999
2    Nov-2001
3    Feb-1996
4    Jan-1996
Name: earliest_cr_line, dtype: object
```

While it does appear that this column is in a datetime format, it is actually on object that needs to be converted. There are still 29 missing values but I will fill them in with the `issue_d` as the loan itself is a form of credit.

[147]:
```python
# Fill in missing dates for the earliest credit line column
X['earliest_cr_line'] = X['earliest_cr_line'].fillna(loan['issue_d'])

# Convert the earliest credit line of the borrower
X['earliest_cr_line'] = pd.to_datetime(X['earliest_cr_line'])
```

In order to include this variable in the multicollinearity and lasso calculations I need to convert it from datetime to numeric.

[148]:
```python
# Convert the variable from datetime to numeric.
X['earliest_cr_line'] = X['earliest_cr_line'].map(dt.datetime.toordinal)
```

initial_list_status: Listing status for if the loan was offered as whole or fractional.

[149]:
```
X['initial_list_status'].unique()
```

[149]:
```
array(['f', 'w'], dtype=object)
```

'w' represents loans that were offered to institutions that would be bought as a full loan. 'f' represents loans that went straight to investors who would by a portion of the loan instead of all of it. Since there are only to values, I can binary encode this variable to have 'f' –> 0 and 'w' –> 1.

[150]:
```
# Convert pyment_plan to a binary variable
X['initial_list_status'] = np.where(X['initial_list_status'] == 'w', 1, 0)
```

last_pymnt_d: The last time that the borrower paid lending club in mm-yyyy

[151]:
```
print(X['last_pymnt_d'].isna().sum())
X['last_pymnt_d'].head(10)
```

```
17659
```

[151]:
```
0    Jan-2015
1    Apr-2013
2    Jun-2014
3    Jan-2015
4    Jan-2016
5    Jan-2015
6    Jan-2016
7    Jan-2015
8    Apr-2012
9    Nov-2012
Name: last_pymnt_d, dtype: object
```

This variable cannot be consistent over the training and test sets that I plan to train the models on. I will drop this variable

[152]:
```
X.drop('last_pymnt_d', axis = 1, inplace = True)
```

next_pymnt_d: The next time that the borrower is expected to pay lending club in mm-yyyy

[153]:
```
print(X['next_pymnt_d'].isna().sum())
X['next_pymnt_d'].head(10)
```

```
252971
```

[153]:
```
0         NaN
1         NaN
2         NaN
3         NaN
4    Feb-2016
5         NaN
6    Feb-2016
```

```
7          NaN
8          NaN
9          NaN
Name: next_pymnt_d, dtype: object
```

This variable has a large amount of missing values from all of the loans that have been fully paid or charged off. It is not a variable that will be consistent over the training and test sets.

[154]:
```
X.drop('next_pymnt_d', axis = 1, inplace = True)
```

`last_credit_pull_d`: The last time lending club pulled the credit score of the borrower mm-yyyy

[155]:
```
print(X['last_credit_pull_d'].isna().sum())
print(X['last_credit_pull_d'].dtype)
X['last_credit_pull_d'].head()
```

```
53
object
```

[155]:
```
0    Jan-2016
1    Sep-2013
2    Jan-2016
3    Jan-2015
4    Jan-2016
Name: last_credit_pull_d, dtype: object
```

There are 53 missing values that I can fill by the issue date because lending club pulls credit for every single loan so presumably there should have been a credit pull around the issue date.

[156]:
```
# Fill in missing dates for the last credit pull column
X['last_credit_pull_d'] = X['last_credit_pull_d'].fillna(loan['issue_d'])
```

Similar to what I did with `earliest_cr_line` I will convert this column to numeric so that I can analyze its multicollinearity and feature importance.

[157]:
```
# Convert the date that the lending company got a credit report
X['last_credit_pull_d'] = pd.to_datetime(X['last_credit_pull_d'])

X['last_credit_pull_d'] = X['last_credit_pull_d'].map(dt.datetime.toordinal)
```

`application_type`: Whether or not the loan application is joint or individual

[158]:
```
X['application_type'].unique()
```

[158]:
```
array(['INDIVIDUAL', 'JOINT'], dtype=object)
```

Since the only two values are 'INDIVIDUAL' and 'JOINT', I can use binary encoding to transform it into a numerical column by having 'INDIVIDUAL' –> 0 and 'JOINT' –> 1

[159]:
```
# Convert pyment_plan to a binary variable
X['application_type'] = np.where(X['application_type'] == 'JOINT', 1, 0)
```

verification_status_joint: Same as the verification_status but for joint applications.

[160]:
```
X['verification_status_joint'].unique()
```

[160]:
```
array([nan, 'Not Verified', 'Verified', 'Source Verified'], dtype=object)
```

The reason why there are missing values in this column is because most of the loans are to individual people. I can fill in the nan with some value that will allow me to use the pandas funtion `get_dummies`

[161]:
```
# Fill the missing values in the application.
X['verification_status_joint'].fillna('indiv')

# One-hot encode the joint verification status
X = pd.get_dummies(X, columns = ['verification_status_joint'], drop_first =␣
↪False)
```

0.1.2 Columns to drop:

- **id**: This is not a relational database so there is no value in having a variable that can tie this dataset to others.
- **member_id**: same as ID, there are no other datasets for that need a bridge variable.

[162]:
```
# drop all of the columns that I had mentioned earlier.
X.drop(['id', 'member_id'], axis = 1, inplace = True)
```

[163]:
```
X.head()
```

[163]:
```
   loan_amnt  funded_amnt  funded_amnt_inv  term  int_rate  installment  \
0     5000.0       5000.0           4975.0     3     10.65       162.87
1     2500.0       2500.0           2500.0     5     15.27        59.83
2     2400.0       2400.0           2400.0     3     15.96        84.33
3    10000.0      10000.0          10000.0     3     13.49       339.31
4     3000.0       3000.0           3000.0     5     12.69        67.79

   grade  emp_length  annual_inc  pymnt_plan    dti  delinq_2yrs  \
0    5.8        10.0     24000.0           0  27.65          0.0
1    4.4         0.5     30000.0           0   1.00          0.0
2    4.2        10.0     12252.0           0   8.72          0.0
3    5.0        10.0     49200.0           0  20.00          0.0
4    5.2         1.0     80000.0           0  17.94          0.0

   earliest_cr_line  inq_last_6mths  mths_since_last_delinq  \
0            724642             1.0                     NaN
1            729845             5.0                     NaN
2            730790             2.0                     NaN
```

```
3           728690            1.0                     35.0
4           728659            0.0                     38.0

   mths_since_last_record  open_acc  pub_rec  revol_bal  revol_util  \
0                     NaN       3.0      0.0    13648.0        83.7
1                     NaN       3.0      0.0     1687.0         9.4
2                     NaN       2.0      0.0     2956.0        98.5
3                     NaN      10.0      0.0     5598.0        21.0
4                     NaN      15.0      0.0    27783.0        53.9

   total_acc  initial_list_status  last_pymnt_amnt  last_credit_pull_d  \
0        9.0                    0           171.62              735964
1        4.0                    0           119.66              735112
2       10.0                    0           649.91              735964
3       37.0                    0           357.48              735599
4       38.0                    0            67.79              735964

   collections_12_mths_ex_med  mths_since_last_major_derog  application_type  \
0                         0.0                          NaN                 0
1                         0.0                          NaN                 0
2                         0.0                          NaN                 0
3                         0.0                          NaN                 0
4                         0.0                          NaN                 0

   annual_inc_joint  dti_joint  acc_now_delinq  tot_coll_amt  tot_cur_bal  \
0               NaN        NaN             0.0           NaN          NaN
1               NaN        NaN             0.0           NaN          NaN
2               NaN        NaN             0.0           NaN          NaN
3               NaN        NaN             0.0           NaN          NaN
4               NaN        NaN             0.0           NaN          NaN

   open_acc_6m  open_il_6m  open_il_12m  open_il_24m  mths_since_rcnt_il  \
0          NaN         NaN          NaN          NaN                 NaN
1          NaN         NaN          NaN          NaN                 NaN
2          NaN         NaN          NaN          NaN                 NaN
3          NaN         NaN          NaN          NaN                 NaN
4          NaN         NaN          NaN          NaN                 NaN

   total_bal_il  il_util  open_rv_12m  open_rv_24m  max_bal_bc  all_util  \
0           NaN      NaN          NaN          NaN         NaN       NaN
1           NaN      NaN          NaN          NaN         NaN       NaN
2           NaN      NaN          NaN          NaN         NaN       NaN
3           NaN      NaN          NaN          NaN         NaN       NaN
4           NaN      NaN          NaN          NaN         NaN       NaN

   total_rev_hi_lim  inq_fi  total_cu_tl  inq_last_12m  emp_type_Accountant  \
0               NaN     NaN          NaN           NaN                    0
```

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

   emp_type_Admin  emp_type_Analyst  emp_type_Assistant  emp_type_Clergy  \
0               0                 0                   0                0
1               0                 0                   0                0
2               0                 0                   0                0
3               0                 0                   0                0
4               0                 0                   0                0

   emp_type_Clerk  emp_type_Designer  emp_type_Director  emp_type_Education  \
0               0                  0                  0                   0
1               0                  0                  0                   0
2               0                  0                  0                   0
3               0                  0                  0                   0
4               0                  0                  0                   0

   emp_type_Executive  emp_type_Finance  emp_type_Healer  emp_type_Law  \
0                   0                 0                0             0
1                   0                 0                0             0
2                   0                 0                0             0
3                   0                 0                0             0
4                   0                 0                0             0

   emp_type_Manager  emp_type_Manlab  emp_type_Operator  emp_type_Sales  \
0                 0                0                  0               0
1                 0                0                  0               0
2                 0                0                  0               0
3                 0                0                  0               0
4                 0                0                  0               0

   emp_type_Service  emp_type_Technical  emp_type_Unemployed  emp_type_Vol  \
0                 0                   0                    1             0
1                 0                   0                    0             0
2                 0                   0                    1             0
3                 0                   0                    0             0
4                 0                   1                    0             0

   emp_type_other  home_ownership_OTHER  home_ownership_OWN  \
0               0                     0                   0
1               1                     0                   0
2               0                     0                   0
3               1                     0                   0
4               0                     0                   0
```

```
   home_ownership_RENT  verification_status_Source Verified  \
0                    1                                    0
1                    1                                    1
2                    1                                    0
3                    1                                    1
4                    1                                    1

   verification_status_Verified  purpose_credit_card  \
0                             1                    1
1                             0                    0
2                             0                    0
3                             0                    0
4                             0                    0

   purpose_debt_consolidation  purpose_educational  purpose_home_improvement  \
0                           0                    0                         0
1                           0                    0                         0
2                           0                    0                         0
3                           0                    0                         0
4                           0                    0                         0

   purpose_house  purpose_major_purchase  purpose_medical  purpose_moving  \
0              0                       0                0               0
1              0                       0                0               0
2              0                       0                0               0
3              0                       0                0               0
4              0                       0                0               0

   purpose_other  purpose_renewable_energy  purpose_small_business  \
0              0                         0                       0
1              0                         0                       0
2              0                         0                       1
3              1                         0                       0
4              1                         0                       0

   purpose_vacation  purpose_wedding  addr_state_AL  addr_state_AR  \
0                 0                0              0              0
1                 0                0              0              0
2                 0                0              0              0
3                 0                0              0              0
4                 0                0              0              0

   addr_state_AZ  addr_state_CA  addr_state_CO  addr_state_CT  addr_state_DC  \
0              1              0              0              0              0
1              0              0              0              0              0
2              0              0              0              0              0
3              0              1              0              0              0
```

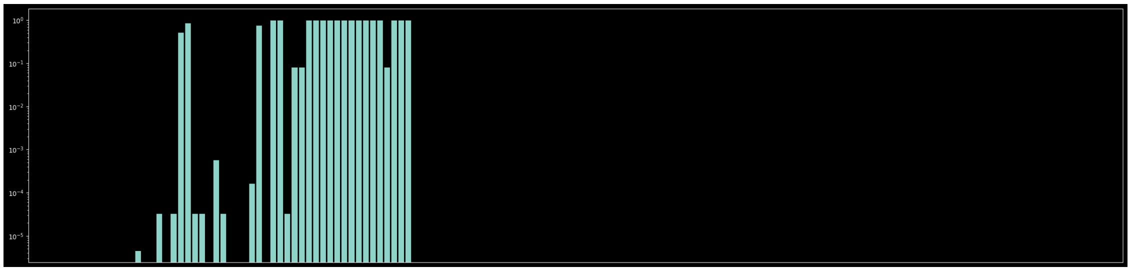
```
4            0            0            0            0            0

   addr_state_DE  addr_state_FL  addr_state_GA  addr_state_HI  addr_state_IA  \
0              0              0              0              0              0
1              0              0              1              0              0
2              0              0              0              0              0
3              0              0              0              0              0
4              0              0              0              0              0

   addr_state_ID  addr_state_IL  addr_state_IN  addr_state_KS  addr_state_KY  \
0              0              0              0              0              0
1              0              0              0              0              0
2              0              1              0              0              0
3              0              0              0              0              0
4              0              0              0              0              0

   addr_state_LA  addr_state_MA  addr_state_MD  addr_state_ME  addr_state_MI  \
0              0              0              0              0              0
1              0              0              0              0              0
2              0              0              0              0              0
3              0              0              0              0              0
4              0              0              0              0              0

   addr_state_MN  addr_state_MO  addr_state_MS  addr_state_MT  addr_state_NC  \
0              0              0              0              0              0
1              0              0              0              0              0
2              0              0              0              0              0
3              0              0              0              0              0
4              0              0              0              0              0

   addr_state_ND  addr_state_NE  addr_state_NH  addr_state_NJ  addr_state_NM  \
0              0              0              0              0              0
1              0              0              0              0              0
2              0              0              0              0              0
3              0              0              0              0              0
4              0              0              0              0              0

   addr_state_NV  addr_state_NY  addr_state_OH  addr_state_OK  addr_state_OR  \
0              0              0              0              0              0
1              0              0              0              0              0
2              0              0              0              0              0
3              0              0              0              0              0
4              0              0              0              0              1

   addr_state_PA  addr_state_RI  addr_state_SC  addr_state_SD  addr_state_TN  \
0              0              0              0              0              0
1              0              0              0              0              0
```

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

   addr_state_TX  addr_state_UT  addr_state_VA  addr_state_VT  addr_state_WA  \
0              0              0              0              0              0
1              0              0              0              0              0
2              0              0              0              0              0
3              0              0              0              0              0
4              0              0              0              0              0

   addr_state_WI  addr_state_WV  addr_state_WY  \
0              0              0              0
1              0              0              0
2              0              0              0
3              0              0              0
4              0              0              0

   verification_status_joint_Not Verified  \
0                                       0
1                                       0
2                                       0
3                                       0
4                                       0

   verification_status_joint_Source Verified  \
0                                          0
1                                          0
2                                          0
3                                          0
4                                          0

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

0.2 Missing Values

[164]:
```
plt.figure(figsize = (30,7));
# Plot missing values for each column
plt.bar(x = X.columns, height = X.isna().sum()/X.shape[0])
# hide the names of the columns for a cleaner graph
plt.xticks([]);
# Set the scale to log so I can see every column that has a missing value
```

```python
plt.yscale('log');
```



[165]:
```python
list(X.columns[X.isna().sum()/X.shape[0] > 0])
```

[165]:
```
['annual_inc',
 'delinq_2yrs',
 'inq_last_6mths',
 'mths_since_last_delinq',
 'mths_since_last_record',
 'open_acc',
 'pub_rec',
 'revol_util',
 'total_acc',
 'collections_12_mths_ex_med',
 'mths_since_last_major_derog',
 'annual_inc_joint',
 'dti_joint',
 'acc_now_delinq',
 'tot_coll_amt',
 'tot_cur_bal',
 'open_acc_6m',
 'open_il_6m',
 'open_il_12m',
 'open_il_24m',
 'mths_since_rcnt_il',
 'total_bal_il',
 'il_util',
 'open_rv_12m',
 'open_rv_24m',
 'max_bal_bc',
 'all_util',
 'total_rev_hi_lim',
 'inq_fi',
 'total_cu_tl',
 'inq_last_12m']
```

The imputation of values for this dataset is the most challenging aspect of this project. There are many columns that have large swaths of missing values that need reasonable values filled in so as to allow the algorithms to capture the existing patterns in the data instead of artificial ones coming from synthetic data.

My assumption was that most missing values were zeroes that people forgot to fill out on forms or didn't bother with. For instance 'open_il_12m' refers to the number of opened installment accounts in the last 12 months. If there isn't any information on the number of installment accounts, then there likely aren't any installment accounts.

Values such as 'mths_since_last_delinq' or months since last delinquent are different because if someone was never delinquent than the correct answer isn't 0 which would imply that they are currently delinquent but a very high number. The higher the value, the better the outcome on their loan. I can use 1000 since it is 83 years worth of non-delinquency.

[166]:
```
# Create a list with all of the columns that have missing values to be filled␣
↪with 0
fill_with_zero = ['annual_inc', 'delinq_2yrs', 'inq_last_6mths', 'open_acc',
                  'pub_rec', 'revol_util', 'total_acc',␣
↪'collections_12_mths_ex_med',
                  'acc_now_delinq', 'tot_coll_amt', 'tot_cur_bal',␣
↪'open_acc_6m',
                  'open_il_6m', 'open_il_12m', 'open_il_24m','total_bal_il',␣
↪'il_util',
                  'open_rv_12m', 'open_rv_24m', 'max_bal_bc', 'all_util',␣
↪'total_rev_hi_lim',
                  'inq_fi', 'total_cu_tl', 'inq_last_12m']

# Fill those columns with 0
X[fill_with_zero] = X[fill_with_zero].fillna(0)
```

[167]:
```
# Create a list with all of the columns that have missing values to be filled␣
↪with 1000
fill_with_1000 = ['mths_since_last_delinq', 'mths_since_last_record',
                  'mths_since_last_major_derog', 'mths_since_rcnt_il']
# Fill those columns with 0
X[fill_with_1000] = X[fill_with_1000].fillna(1000)
```

[168]:
```
# List out all the columns that still having missing values
list(X.columns[X.isna().sum()/X.shape[0] > 0])
```

[168]:
```
['annual_inc_joint', 'dti_joint']
```

[169]:
```
# Combine annual inc columns joint and indiv into one.
X['annual_inc'] = [loan['annual_inc'][i] if math.isnan(x) else x for i, x in␣
↪enumerate(loan['annual_inc_joint'])]
# Create a column that documents if the column above is joint or indiv
X['inc_joint'] = np.where(loan['annual_inc_joint'].isnull(), 0, 1)
```

[170]:
```
# Combine dti columns joint and indiv into one.
X['dti'] = [X['dti'][i] if math.isnan(x) else x for i, x in␣
↪enumerate(X['dti_joint'])]
# Create a column that documents if the column above is joint or indiv
X['dti_joint'] = np.where(loan['dti_joint'].isnull(), 0, 1)
```

[171]:
```
# Check to see if there are still missing columns
list(X.columns[X.isna().sum()/X.shape[0] > 0])
```

[171]: ['annual_inc', 'annual_inc_joint']

[172]:
```
# Drop 'annual_inc_joint' now that it has been added to 'annual_inc'
X.drop('annual_inc_joint', axis = 1, inplace = True)
# Fill in the missing values with 0 as the assumption is there is no income
X['annual_inc'].fillna(0, inplace = True)
```

[173]:
```
# Check to see if there are still missing columns
list(X.columns[X.isna().sum()/X.shape[0] > 0])
```

[173]: []

[174]:
```
# Initialize the standard scaler
scaler = StandardScaler()
# Fit and transform X into a dataframe so that I can keep using pandas
X_scaled = pd.DataFrame(scaler.fit_transform(X), columns = X.columns)
```

[175]:
```
X_scaled.head()
```

[175]:
```
   loan_amnt  funded_amnt  funded_amnt_inv      term  int_rate  installment  \
0  -1.156460    -1.155635        -1.152256 -0.654724 -0.592611    -1.121467
1  -1.452829    -1.452198        -1.445430  1.527360  0.461735    -1.543440
2  -1.464683    -1.464061        -1.457275 -0.654724  0.619202    -1.443107
3  -0.563724    -0.562507        -0.557025 -0.654724  0.055515    -0.398905
4  -1.393555    -1.392886        -1.386203  1.527360 -0.127055    -1.510842

      grade  emp_length  annual_inc  pymnt_plan       dti  delinq_2yrs  \
0  0.763889    1.134870   -0.789014   -0.003357  1.147690    -0.364672
1 -0.314193   -1.398891   -0.696292   -0.003357 -2.063920    -0.364672
2 -0.468204    1.134870   -0.970563   -0.003357 -1.133577    -0.364672
3  0.147842    1.134870   -0.399582   -0.003357  0.225784    -0.364672
4  0.301854   -1.265535    0.076390   -0.003357 -0.022468    -0.364672

   earliest_cr_line  inq_last_6mths  mths_since_last_delinq  \
0         -1.771866        0.305877                0.975849
1          0.143584        4.312132                0.975849
2          0.491479        1.307441                0.975849
3         -0.281622        0.305877               -1.021784
```

```
4        -0.293034       -0.695687               -1.015574

   mths_since_last_record  open_acc   pub_rec  revol_bal  revol_util  \
0                0.427154 -1.607499 -0.335522  -0.145932    1.201140
1                0.427154 -1.607499 -0.335522  -0.679268   -1.912396
2                0.427154 -1.795553 -0.335522  -0.622684    1.821333
3                0.427154 -0.291124 -0.335522  -0.504878   -1.426299
4                0.427154  0.649144 -0.335522   0.484341   -0.047627

   total_acc  initial_list_status  last_pymnt_amnt  last_credit_pull_d  \
0  -1.373775             -0.97077        -0.415561            0.302329
1  -1.796028             -0.97077        -0.426398           -3.540893
2  -1.289324             -0.97077        -0.315809            0.302329
3   0.990842             -0.97077        -0.376798           -1.344122
4   1.075293             -0.97077        -0.437216            0.302329

   collections_12_mths_ex_med  mths_since_last_major_derog  application_type  \
0                   -0.107149                     0.576897         -0.024004
1                   -0.107149                     0.576897         -0.024004
2                   -0.107149                     0.576897         -0.024004
3                   -0.107149                     0.576897         -0.024004
4                   -0.107149                     0.576897         -0.024004

   dti_joint  acc_now_delinq  tot_coll_amt  tot_cur_bal  open_acc_6m  \
0  -0.023957       -0.064298     -0.021004    -0.843347    -0.103891
1  -0.023957       -0.064298     -0.021004    -0.843347    -0.103891
2  -0.023957       -0.064298     -0.021004    -0.843347    -0.103891
3  -0.023957       -0.064298     -0.021004    -0.843347    -0.103891
4  -0.023957       -0.064298     -0.021004    -0.843347    -0.103891

   open_il_6m  open_il_12m  open_il_24m  mths_since_rcnt_il  total_bal_il  \
0   -0.107375    -0.094692    -0.109934            0.154904     -0.100884
1   -0.107375    -0.094692    -0.109934            0.154904     -0.100884
2   -0.107375    -0.094692    -0.109934            0.154904     -0.100884
3   -0.107375    -0.094692    -0.109934            0.154904     -0.100884
4   -0.107375    -0.094692    -0.109934            0.154904     -0.100884

    il_util  open_rv_12m  open_rv_24m  max_bal_bc  all_util  total_rev_hi_lim  \
0 -0.139209    -0.105267    -0.117038   -0.116273 -0.149047          -0.79786
1 -0.139209    -0.105267    -0.117038   -0.116273 -0.149047          -0.79786
2 -0.139209    -0.105267    -0.117038   -0.116273 -0.149047          -0.79786
3 -0.139209    -0.105267    -0.117038   -0.116273 -0.149047          -0.79786
4 -0.139209    -0.105267    -0.117038   -0.116273 -0.149047          -0.79786

     inq_fi  total_cu_tl  inq_last_12m  emp_type_Accountant  emp_type_Admin  \
0 -0.085105    -0.076504     -0.088306            -0.109205       -0.104843
1 -0.085105    -0.076504     -0.088306            -0.109205       -0.104843
```

```
2 -0.085105    -0.076504     -0.088306            -0.109205        -0.104843
3 -0.085105    -0.076504     -0.088306            -0.109205        -0.104843
4 -0.085105    -0.076504     -0.088306            -0.109205        -0.104843

   emp_type_Analyst  emp_type_Assistant  emp_type_Clergy  emp_type_Clerk  \
0         -0.157505           -0.147428        -0.039637       -0.077757
1         -0.157505           -0.147428        -0.039637       -0.077757
2         -0.157505           -0.147428        -0.039637       -0.077757
3         -0.157505           -0.147428        -0.039637       -0.077757
4         -0.157505           -0.147428        -0.039637       -0.077757

   emp_type_Designer  emp_type_Director  emp_type_Education  \
0          -0.052789          -0.196448           -0.186479
1          -0.052789          -0.196448           -0.186479
2          -0.052789          -0.196448           -0.186479
3          -0.052789          -0.196448           -0.186479
4          -0.052789          -0.196448           -0.186479

   emp_type_Executive  emp_type_Finance  emp_type_Healer  emp_type_Law  \
0           -0.180873         -0.117117        -0.245189     -0.046832
1           -0.180873         -0.117117        -0.245189     -0.046832
2           -0.180873         -0.117117        -0.245189     -0.046832
3           -0.180873         -0.117117        -0.245189     -0.046832
4           -0.180873         -0.117117        -0.245189     -0.046832

   emp_type_Manager  emp_type_Manlab  emp_type_Operator  emp_type_Sales  \
0         -0.435078        -0.082293          -0.169413       -0.142468
1         -0.435078        -0.082293          -0.169413       -0.142468
2         -0.435078        -0.082293          -0.169413       -0.142468
3         -0.435078        -0.082293          -0.169413       -0.142468
4         -0.435078        -0.082293          -0.169413       -0.142468

   emp_type_Service  emp_type_Technical  emp_type_Unemployed  emp_type_Vol  \
0         -0.160378           -0.395006             4.030308     -0.172517
1         -0.160378           -0.395006            -0.248120     -0.172517
2         -0.160378           -0.395006             4.030308     -0.172517
3         -0.160378           -0.395006            -0.248120     -0.172517
4         -0.160378            2.531605            -0.248120     -0.172517

   emp_type_other  home_ownership_OTHER  home_ownership_OWN  \
0       -0.631040             -0.016276           -0.330681
1        1.584686             -0.016276           -0.330681
2       -0.631040             -0.016276           -0.330681
3        1.584686             -0.016276           -0.330681
4       -0.631040             -0.016276           -0.330681

   home_ownership_RENT  verification_status_Source Verified  \
```

```
0            1.2214                            -0.768632
1            1.2214                             1.301013
2            1.2214                            -0.768632
3            1.2214                             1.301013
4            1.2214                             1.301013

   verification_status_Verified  purpose_credit_card  \
0                      1.431317             1.817653
1                     -0.698657            -0.550160
2                     -0.698657            -0.550160
3                     -0.698657            -0.550160
4                     -0.698657            -0.550160

   purpose_debt_consolidation  purpose_educational  purpose_home_improvement  \
0                   -1.201443            -0.021838                 -0.249058
1                   -1.201443            -0.021838                 -0.249058
2                   -1.201443            -0.021838                 -0.249058
3                   -1.201443            -0.021838                 -0.249058
4                   -1.201443            -0.021838                 -0.249058

   purpose_house  purpose_major_purchase  purpose_medical  purpose_moving  \
0      -0.064769               -0.140912        -0.098577       -0.078349
1      -0.064769               -0.140912        -0.098577       -0.078349
2      -0.064769               -0.140912        -0.098577       -0.078349
3      -0.064769               -0.140912        -0.098577       -0.078349
4      -0.064769               -0.140912        -0.098577       -0.078349

   purpose_other  purpose_renewable_energy  purpose_small_business  \
0      -0.225373                 -0.025464               -0.108777
1      -0.225373                 -0.025464               -0.108777
2      -0.225373                 -0.025464                9.193151
3       4.437084                 -0.025464               -0.108777
4       4.437084                 -0.025464               -0.108777

   purpose_vacation  purpose_wedding  addr_state_AL  addr_state_AR  \
0         -0.073251        -0.051496      -0.113061      -0.086828
1         -0.073251        -0.051496      -0.113061      -0.086828
2         -0.073251        -0.051496      -0.113061      -0.086828
3         -0.073251        -0.051496      -0.113061      -0.086828
4         -0.073251        -0.051496      -0.113061      -0.086828

   addr_state_AZ  addr_state_CA  addr_state_CO  addr_state_CT  addr_state_DC  \
0       6.517162      -0.413398      -0.147149      -0.124436      -0.052423
1      -0.153441      -0.413398      -0.147149      -0.124436      -0.052423
2      -0.153441      -0.413398      -0.147149      -0.124436      -0.052423
3      -0.153441       2.418977      -0.147149      -0.124436      -0.052423
4      -0.153441      -0.413398      -0.147149      -0.124436      -0.052423
```

	addr_state_DE	addr_state_FL	addr_state_GA	addr_state_HI	addr_state_IA	\
0	-0.05327	-0.271536	-0.184084	-0.071949	-0.003972	
1	-0.05327	-0.271536	5.432297	-0.071949	-0.003972	
2	-0.05327	-0.271536	-0.184084	-0.071949	-0.003972	
3	-0.05327	-0.271536	-0.184084	-0.071949	-0.003972	
4	-0.05327	-0.271536	-0.184084	-0.071949	-0.003972	

	addr_state_ID	addr_state_IL	addr_state_IN	addr_state_KS	addr_state_KY	\
0	-0.003677	-0.204067	-0.125636	-0.094934	-0.098635	
1	-0.003677	-0.204067	-0.125636	-0.094934	-0.098635	
2	-0.003677	4.900357	-0.125636	-0.094934	-0.098635	
3	-0.003677	-0.204067	-0.125636	-0.094934	-0.098635	
4	-0.003677	-0.204067	-0.125636	-0.094934	-0.098635	

	addr_state_LA	addr_state_MA	addr_state_MD	addr_state_ME	addr_state_MI	\
0	-0.109885	-0.154136	-0.155806	-0.024331	-0.163067	
1	-0.109885	-0.154136	-0.155806	-0.024331	-0.163067	
2	-0.109885	-0.154136	-0.155806	-0.024331	-0.163067	
3	-0.109885	-0.154136	-0.155806	-0.024331	-0.163067	
4	-0.109885	-0.154136	-0.155806	-0.024331	-0.163067	

	addr_state_MN	addr_state_MO	addr_state_MS	addr_state_MT	addr_state_NC	\
0	-0.13532	-0.127556	-0.065744	-0.053768	-0.16928	
1	-0.13532	-0.127556	-0.065744	-0.053768	-0.16928	
2	-0.13532	-0.127556	-0.065744	-0.053768	-0.16928	
3	-0.13532	-0.127556	-0.065744	-0.053768	-0.16928	
4	-0.13532	-0.127556	-0.065744	-0.053768	-0.16928	

	addr_state_ND	addr_state_NE	addr_state_NH	addr_state_NJ	addr_state_NM	\
0	-0.02324	-0.036428	-0.069732	-0.197322	-0.074813	
1	-0.02324	-0.036428	-0.069732	-0.197322	-0.074813	
2	-0.02324	-0.036428	-0.069732	-0.197322	-0.074813	
3	-0.02324	-0.036428	-0.069732	-0.197322	-0.074813	
4	-0.02324	-0.036428	-0.069732	-0.197322	-0.074813	

	addr_state_NV	addr_state_NY	addr_state_OH	addr_state_OK	addr_state_OR	\
0	-0.119254	-0.301818	-0.185863	-0.09589	-0.111481	
1	-0.119254	-0.301818	-0.185863	-0.09589	-0.111481	
2	-0.119254	-0.301818	-0.185863	-0.09589	-0.111481	
3	-0.119254	-0.301818	-0.185863	-0.09589	-0.111481	
4	-0.119254	-0.301818	-0.185863	-0.09589	8.970130	

	addr_state_PA	addr_state_RI	addr_state_SC	addr_state_SD	addr_state_TN	\
0	-0.191506	-0.066381	-0.110158	-0.045272	-0.121394	
1	-0.191506	-0.066381	-0.110158	-0.045272	-0.121394	
2	-0.191506	-0.066381	-0.110158	-0.045272	-0.121394	

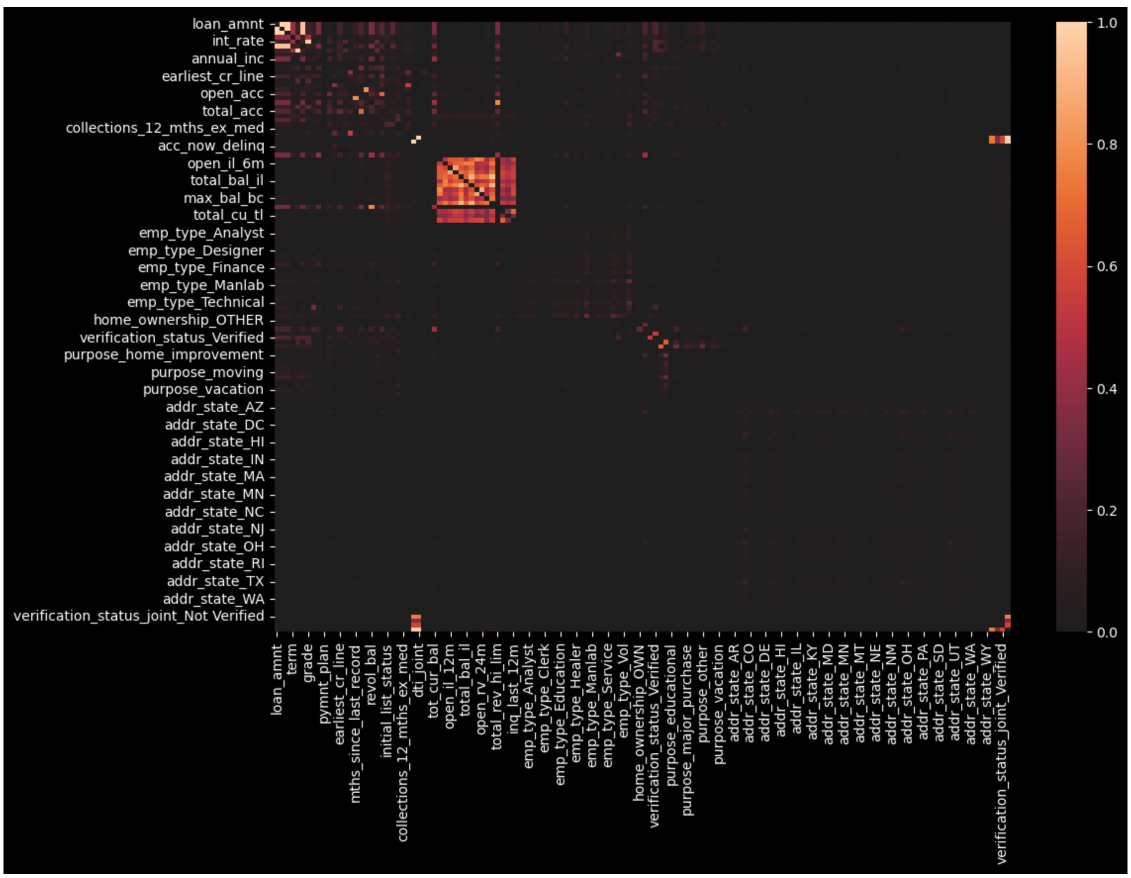
```
3      -0.191506      -0.066381      -0.110158      -0.045272      -0.121394
4      -0.191506      -0.066381      -0.110158      -0.045272      -0.121394

   addr_state_TX  addr_state_UT  addr_state_VA  addr_state_VT  addr_state_WA  \
0      -0.295217      -0.084316      -0.174612      -0.045046      -0.149636
1      -0.295217      -0.084316      -0.174612      -0.045046      -0.149636
2      -0.295217      -0.084316      -0.174612      -0.045046      -0.149636
3      -0.295217      -0.084316      -0.174612      -0.045046      -0.149636
4      -0.295217      -0.084316      -0.174612      -0.045046      -0.149636

   addr_state_WI  addr_state_WV  addr_state_WY  \
0      -0.114958      -0.070478       -0.04786
1      -0.114958      -0.070478       -0.04786
2      -0.114958      -0.070478       -0.04786
3      -0.114958      -0.070478       -0.04786
4      -0.114958      -0.070478       -0.04786

   verification_status_joint_Not Verified  \
0                               -0.017861
1                               -0.017861
2                               -0.017861
3                               -0.017861
4                               -0.017861

   verification_status_joint_Source Verified  \
0                                  -0.008291
1                                  -0.008291
2                                  -0.008291
3                                  -0.008291
4                                  -0.008291

   verification_status_joint_Verified  inc_joint
0                            -0.01372  -0.024004
1                            -0.01372  -0.024004
2                            -0.01372  -0.024004
3                            -0.01372  -0.024004
4                            -0.01372  -0.024004
```

Multicollinearity

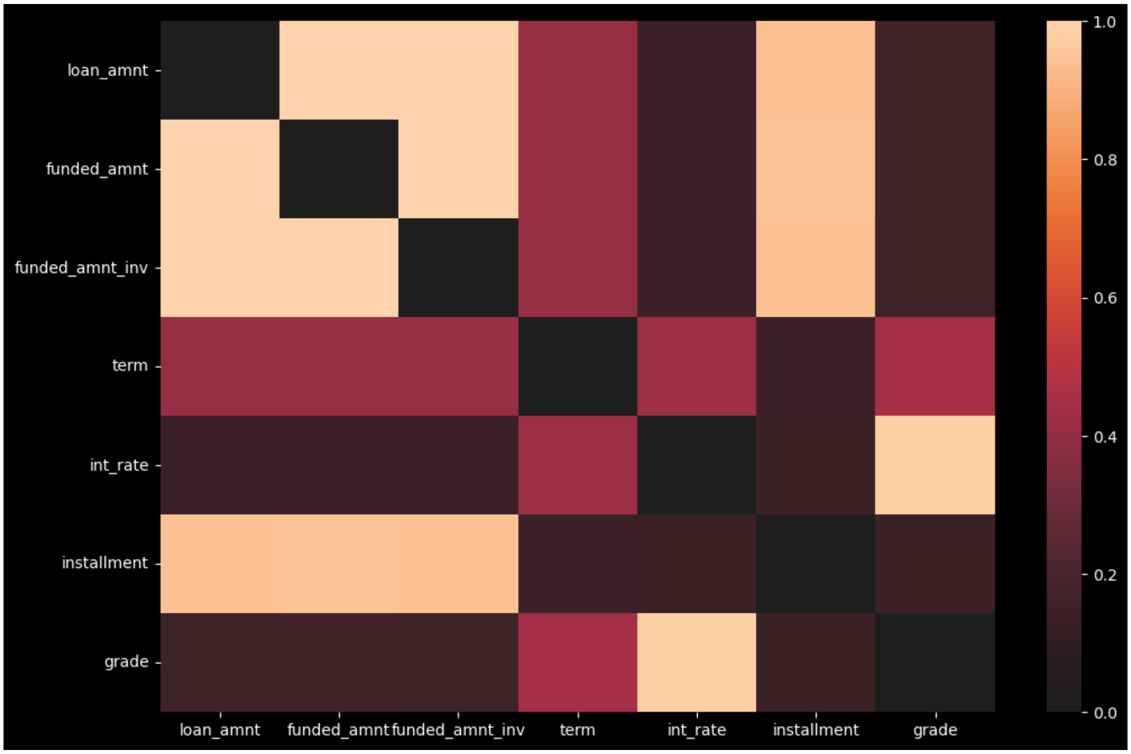
[176]:
```python
# Function to get rid of all the diagonals in correlation heatmaps
def zero_diagonal_heatmap(dfx, min = 0, max = None, color_map = None,
 ↪threshhold = 0):
    # give the max a base value
    if max is None:
        max = dfx.shape[1]
    # Create a numpy matrix that is easy to manipulate
    cm = np.array(abs(dfx.iloc[:,min:max].corr(numeric_only = False)))
```

```
    # Create a loop that replaces all the diagonal values with 0
    for i in range(len(cm)):
        # Loop thorough the other axis of values
        for j in range(len(cm)):
            # All diagonal values happen when i == j
            if i == j:
                # set the diagonal value equal to 0
                cm[i,j] = 0
    df_cm = pd.DataFrame(cm, columns = list(dfx.columns)[min:max], index =
 ↪list(dfx.columns)[min:max])
    # Make the plot bigger
    plt.figure(figsize=(12, 8))
    # Plot the heatmap
    ax = sns.heatmap(df_cm, cmap = color_map, center = threshhold, vmin = 0,
 ↪vmax = 1);
```

[177]:
```
zero_diagonal_heatmap(X_scaled)
```



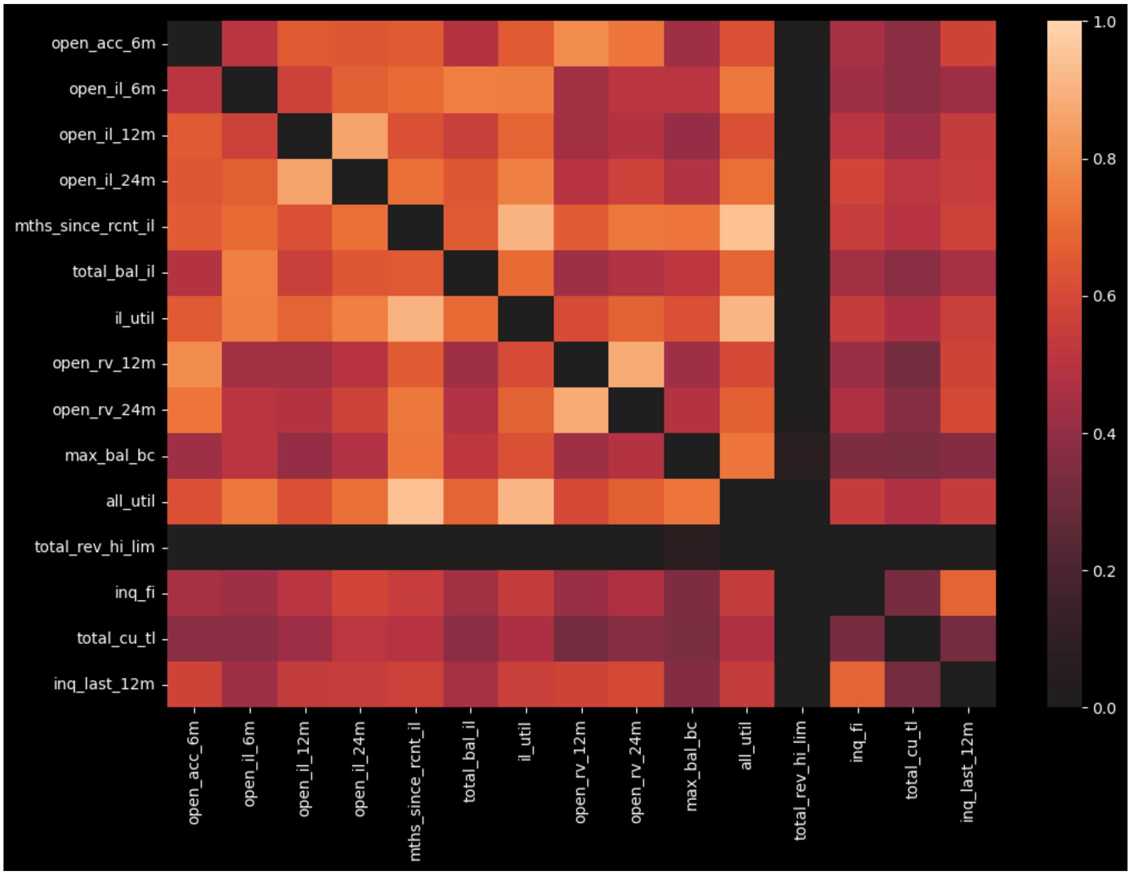
[178]:
```
# Create a heatmap of the first 7 columns
zero_diagonal_heatmap(X_scaled, 0, 7)
```



'loan_amnt' is the amount the requested amount of money for the loan from the borrower, 'funded_amnt' is the amount of money that has been committed to the loan, and 'funded_amnt_inv is the amount of money investors have committed to the loan. Since loan amount is the only variable that the applicant has control over, I believe it is the only relevant variable to keep. 'installment' is the monthly payment if the loan originates, this is a function of the loan amount and the term and can be dropped. 'int_rate' is the interest rate of the loan and it is directly determined by the grade that lending club gives the loan. I can pick one of the two and it will contain the information of the other.

[179]:
```
# Drop all the columns that are clearly collinear that are the least interesting
X_scaled.drop(['funded_amnt','funded_amnt_inv', 'installment','int_rate'], axis␣
↪= 1, inplace = True)
```

[180]:
```
zero_diagonal_heatmap(X_scaled,27,42)
```



[181]:
```
# Create a function that takes a dataframe and a correlation level
def corr_dict(dfx, corr = 0.7):
    # Create the correlation with absolute values to measure collinearity
    corr_mat = abs(dfx.corr(numeric_only = False)) > corr    # type: ignore
    # Define the dictionary outside the function for uses
    global correl_dict
    # Create a dictionary with sorted values so that it is easy to see the next␣
↪steps
    correl_dict = dict(corr_mat.sum().sort_values(ascending = False))
    # Subtract each value by one for the self correlation on the diagonal
    correl_dict = {key: value - 1 for key, value in correl_dict.items() if␣
↪value != 1}
    # Print the dictionary
    print(correl_dict)
```

[182]:
```
# Create a function that takes a list of features and minimum number of␣
↪correlations
def corr_list(min_corr, corr, still_high_mcl = None):
    # Define the dictionary where all the information will be stored
    correlations2 = {}
```

```
    # Almost always the default value will be None
    if still_high_mcl is None:
        # # the default list is the keys from correl_dict whose values are at␣
↪least min_corr
        still_high_mcl = [key for key, value in correl_dict.items() if value >=␣
↪min_corr]
        # Drop all date time objects from the list by using the ending _d as an␣
↪identifier
        still_high_mcl = [element for element in still_high_mcl if not element.
↪endswith("_d")]
    # Store the correlation matrix in a variable for easy use
    still_corr = abs(X_scaled[still_high_mcl].corr(numeric_only = True))
    # Put the index and column names as the feature names
    still_corr.columns = still_high_mcl
    still_corr.index = still_high_mcl
    # Iterate through all the features in the list
    for shm in still_high_mcl:
        # create a list of all the features that have a correlation of over 7
        corrs2 = still_corr.index[still_corr[shm] > corr].tolist() #type: ignore
        # remove the self correlation
        corrs2.remove(shm)
        # input the feature and list into the dictionary
        correlations2.update({shm: corrs2})
    # print the result
    print(correlations2)
```

[183]:
```
# Find the number of features each feature is correlated with
corr_dict(X_scaled)
```

```
{'all_util': 5, 'il_util': 5, 'mths_since_rcnt_il': 5, 'open_il_24m': 4,
'inc_joint': 3, 'application_type': 3, 'verification_status_joint_Not Verified':
3, 'open_rv_24m': 3, 'open_il_6m': 3, 'dti_joint': 3, 'max_bal_bc': 2,
'open_rv_12m': 2, 'total_bal_il': 2, 'open_acc_6m': 2, 'open_il_12m': 1,
'total_rev_hi_lim': 1, 'mths_since_last_record': 1, 'pub_rec': 1, 'revol_bal':
1}
```

[184]:
```
# Get the list of features each feature is correlated with.
corr_list(corr = 0.7, min_corr = 1)
```

```
{'all_util': ['il_util', 'mths_since_rcnt_il', 'open_il_24m', 'open_il_6m',
'max_bal_bc'], 'il_util': ['all_util', 'mths_since_rcnt_il', 'open_il_24m',
'open_il_6m', 'total_bal_il'], 'mths_since_rcnt_il': ['all_util', 'il_util',
'open_il_24m', 'open_rv_24m', 'max_bal_bc'], 'open_il_24m': ['all_util',
'il_util', 'mths_since_rcnt_il', 'open_il_12m'], 'inc_joint':
['application_type', 'verification_status_joint_Not Verified', 'dti_joint'],
'application_type': ['inc_joint', 'verification_status_joint_Not Verified',
'dti_joint'], 'verification_status_joint_Not Verified': ['inc_joint',
'application_type', 'dti_joint'], 'open_rv_24m': ['mths_since_rcnt_il',
```

```
'open_rv_12m', 'open_acc_6m'], 'open_il_6m': ['all_util', 'il_util',
'total_bal_il'], 'dti_joint': ['inc_joint', 'application_type',
'verification_status_joint_Not Verified'], 'max_bal_bc': ['all_util',
'mths_since_rcnt_il'], 'open_rv_12m': ['open_rv_24m', 'open_acc_6m'],
'total_bal_il': ['il_util', 'open_il_6m'], 'open_acc_6m': ['open_rv_24m',
'open_rv_12m'], 'open_il_12m': ['open_il_24m'], 'total_rev_hi_lim':
['revol_bal'], 'mths_since_last_record': ['pub_rec'], 'pub_rec':
['mths_since_last_record'], 'revol_bal': ['total_rev_hi_lim']}
```

One set of features that is consistently multicollinear is the features that are measured in 6 months, 12 months, and 24 months. It isn't surprising that they are collinear since they are measuring the same thing but it is important to combine them so that I can preserve their information without confusing their impact. The simplest way to do this is to set the time frame equal and then compare.

[185]:
```
# Scale the number of 6 month installment accounts to 12
X_scaled['il_6'] =  X_scaled['open_il_6m'] * 2
# Scale the number of 24 month installment accounts to 12
X_scaled['il_24'] =  X_scaled['open_il_24m'] / 2
# Take the max out of all the scaled features to find if there is a spike in␣
 ↪install accounts.
X_scaled['open_install_acc'] = X_scaled[['il_6', 'open_il_12m', 'il_24']].
 ↪max(axis = 1)

# Scale the number of 24 month revolving accounts to 12
X_scaled['rv_24'] =  X_scaled['open_rv_24m'] / 2
# Take the max out of all the scaled features to find if there is a spike in␣
 ↪revolving accounts.
X_scaled['open_revolving'] = X_scaled[['open_rv_12m', 'rv_24']].max(axis = 1)

# Drop all the original columns now that they are useless.
X_scaled.drop(['il_6', 'open_il_12m', 'il_24', 'open_rv_12m', 'rv_24',␣
 ↪'open_il_6m', 'open_il_24m','open_rv_24m'], axis = 1, inplace = True)
```

[186]:
```
# Find the number of features each feature is correlated with again.
corr_dict(X_scaled)
```

```
{'mths_since_rcnt_il': 5, 'open_install_acc': 4, 'il_util': 4, 'all_util': 4,
'inc_joint': 3, 'dti_joint': 3, 'application_type': 3,
'verification_status_joint_Not Verified': 3, 'open_revolving': 2,
'total_bal_il': 2, 'max_bal_bc': 2, 'revol_bal': 1, 'total_rev_hi_lim': 1,
'open_acc_6m': 1, 'pub_rec': 1, 'mths_since_last_record': 1}
```

[187]:
```
# Get the list of features each feature is correlated with again.
corr_list(corr = 0.7, min_corr = 1)
```

```
{'mths_since_rcnt_il': ['open_install_acc', 'il_util', 'all_util',
'open_revolving', 'max_bal_bc'], 'open_install_acc': ['mths_since_rcnt_il',
'il_util', 'all_util', 'total_bal_il'], 'il_util': ['mths_since_rcnt_il',
'open_install_acc', 'all_util', 'total_bal_il'], 'all_util':
```

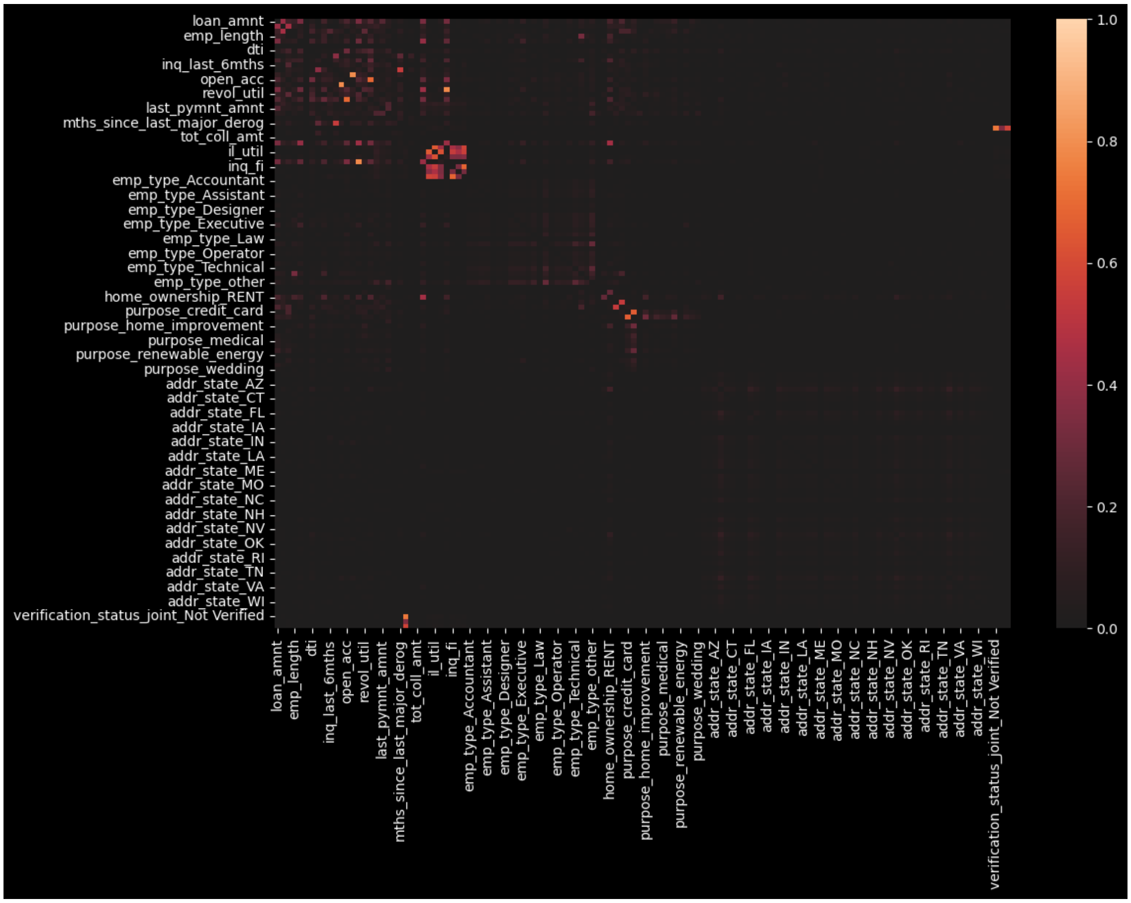
```
['mths_since_rcnt_il', 'open_install_acc', 'il_util', 'max_bal_bc'],
'inc_joint': ['dti_joint', 'application_type', 'verification_status_joint_Not
Verified'], 'dti_joint': ['inc_joint', 'application_type',
'verification_status_joint_Not Verified'], 'application_type': ['inc_joint',
'dti_joint', 'verification_status_joint_Not Verified'],
'verification_status_joint_Not Verified': ['inc_joint', 'dti_joint',
'application_type'], 'open_revolving': ['mths_since_rcnt_il', 'open_acc_6m'],
'total_bal_il': ['open_install_acc', 'il_util'], 'max_bal_bc':
['mths_since_rcnt_il', 'all_util'], 'revol_bal': ['total_rev_hi_lim'],
'total_rev_hi_lim': ['revol_bal'], 'open_acc_6m': ['open_revolving'], 'pub_rec':
['mths_since_last_record'], 'mths_since_last_record': ['pub_rec']}
```

Criteria for choosing collinear features - features that have default values of 0 are better than features with undefined values. - dollar amounts over number of accounts. They give more information and once they are properly scaled they will be more useful. - The more information the variable contains the better. - Synthesized features are better than original features.

il util max__bal__bc total__rev__hi__lim

[188]:
```python
# Drop the features that least fit the criteria
X_scaled.drop(['mths_since_rcnt_il', 'all_util', 'open_install_acc',␣
 ↪'open_revolving',
               'total_bal_il', 'inc_joint', 'application_type'], axis = 1,␣
 ↪inplace = True)
```

[189]:
```python
zero_diagonal_heatmap(X_scaled)
```



[190]:
```
corr_dict(X_scaled)
```

```
{'mths_since_last_record': 1, 'dti_joint': 1, 'verification_status_joint_Not
Verified': 1, 'revol_bal': 1, 'total_rev_hi_lim': 1, 'pub_rec': 1}
```

[191]:
```
corr_list(corr = 0.7, min_corr = 1)
```

```
{'mths_since_last_record': ['pub_rec'], 'dti_joint':
['verification_status_joint_Not Verified'], 'verification_status_joint_Not
Verified': ['dti_joint'], 'revol_bal': ['total_rev_hi_lim'], 'total_rev_hi_lim':
['revol_bal'], 'pub_rec': ['mths_since_last_record']}
```

Unsurprisingly two features that both rely on the borrowers having a joint account are collinear. I will drop the dummy variable as the other one contains more information

[192]:
```
# Drop the least relevant columns that are in corr_list
X_scaled.drop(['mths_since_last_record', 'total_rev_hi_lim',␣
↪'verification_status_joint_Not Verified'], axis = 1, inplace = True)
```

To make sure that I had fully eliminated the multicollinearity I calculated every remaining feature's

Variance Inflation Factor (VIF). The formula for VIF is:

$$VIF = \frac{1}{1 - R^2}$$

Where R^2 is the R-squared value that represents correlations.

[193]:
```
# Create vif function with a dataframe as the argument
def vif(dfx):
    #calculate the vif for the dataframe to see if it is non collinear
    vifs = [variance_inflation_factor(dfx.values, i) for i in range(dfx.
 ↪shape[1])]
    # Put those numbers into a dictionary
    vif_dict = {dfx.columns[i]:vifs[i] for i in range(dfx.shape[1])}
    # Create a filtered dictionary to find out how many features have a VIF␣
 ↪under 2.5
    filtered_dict = {k: v for k, v in vif_dict.items() if v < 2.5}
    # Print the amount of columns that aren't collinear out of all the columns
    print(f"{len(filtered_dict)} out of {len(dfx.columns)} features have a vif␣
 ↪< 2.5")
    # Create a sorted dictionary based on the values
    vif_dict = {k:v for k, v in sorted(vif_dict.items(), key=lambda item:␣
 ↪item[1], reverse = True)}
    # Print the result
    print(vif_dict)
```

[194]:
```
vif(X_scaled)
```

```
57 out of 123 features have a vif < 2.5
{'emp_type_other': 169.57974002764598, 'emp_type_Manager': 111.7147586447447,
'emp_type_Technical': 97.59897197456667, 'addr_state_CA': 51.094645200048134,
'emp_type_Unemployed': 46.39212956418784, 'emp_type_Healer': 45.254717422269884,
'addr_state_NY': 31.767745462942365, 'addr_state_TX': 30.633566724669535,
'emp_type_Director': 30.59330240518125, 'emp_type_Education': 27.89717101959111,
'addr_state_FL': 26.705856680717247, 'emp_type_Executive': 26.408281585647327,
'purpose_debt_consolidation': 24.96764806132425, 'emp_type_Vol':
24.23857849854677, 'emp_type_Operator': 23.461031656313015, 'emp_type_Service':
21.245410815327098, 'emp_type_Analyst': 20.537956437604667,
'purpose_credit_card': 18.93967828897951, 'emp_type_Assistant':
18.239376685202956, 'emp_type_Sales': 17.13106743520082, 'addr_state_IL':
16.424458850979022, 'addr_state_NJ': 15.502262195704335, 'addr_state_PA':
14.716247800285, 'addr_state_OH': 13.980629680318021, 'addr_state_GA':
13.74480709611746, 'addr_state_VA': 12.536294915046794, 'emp_type_Finance':
12.043774061171954, 'addr_state_NC': 11.88644182350205, 'addr_state_MI':
11.147880276767328, 'emp_type_Accountant': 10.636077148271605, 'addr_state_MD':
10.29940975567778, 'addr_state_MA': 10.110592376611661, 'addr_state_AZ':
10.032653547892291, 'emp_type_Admin': 9.900472150565802, 'addr_state_WA':
9.604463414936513, 'addr_state_CO': 9.335793678183599, 'addr_state_MN':
```

8.09825045258343, 'addr_state_MO': 7.3337478164256416, 'addr_state_IN': 7.156485816507435, 'addr_state_CT': 7.036377600696594, 'addr_state_TN': 6.755168526348449, 'addr_state_NV': 6.5577667265598, 'emp_type_Manlab': 6.5364938162900845, 'purpose_home_improvement': 6.521215768226548, 'addr_state_WI': 6.175510070761902, 'addr_state_AL': 6.011851495010638, 'emp_type_Clerk': 5.952461595533043, 'addr_state_OR': 5.87356020795903, 'addr_state_SC': 5.763179271570182, 'addr_state_LA': 5.739137666645026, 'purpose_other': 5.6022187340814265, 'addr_state_KY': 4.837516798054074, 'addr_state_OK': 4.629869513098388, 'addr_state_KS': 4.55823347482431, 'addr_state_AR': 3.9876202810025814, 'addr_state_UT': 3.8173769222531315, 'emp_type_Designer': 3.29235512104278, 'addr_state_NM': 3.224544717816936, 'addr_state_HI': 3.0593367943517693, 'addr_state_WV': 2.979129559660258, 'addr_state_NH': 2.9350223110414437, 'purpose_major_purchase': 2.8961488105820443, 'emp_type_Law': 2.8076997711159604, 'addr_state_RI': 2.755806044102352, 'il_util': 2.7308005905563095, 'addr_state_MS': 2.725067872453719, 'total_acc': 2.3412784187313673, 'emp_type_Clergy': 2.29732842794935, 'inq_last_12m': 2.272432327564242, 'open_acc': 2.236904562835468, 'purpose_small_business': 2.176687879808439, 'addr_state_MT': 2.155466265716131, 'addr_state_DE': 2.134154484319103, 'addr_state_DC': 2.1003725611804636, 'inq_fi': 2.027959693450723, 'open_acc_6m': 1.9692331466328516, 'purpose_medical': 1.9542082569980301, 'addr_state_WY': 1.916714690815398, 'grade': 1.8826323466447483, 'tot_cur_bal': 1.8817770223027797, 'addr_state_SD': 1.8212586202969663, 'dti_joint': 1.8138557594089295, 'addr_state_VT': 1.8123857317559084, 'loan_amnt': 1.770450023225754, 'verification_status_Verified': 1.6876383832952806, 'max_bal_bc': 1.6706128393200244, 'home_ownership_RENT': 1.6655753043101997, 'purpose_moving': 1.6129660556591545, 'term': 1.6119942311081126, 'mths_since_last_delinq': 1.6025300556111002, 'verification_status_joint_Verified': 1.5951578280856389, 'purpose_vacation': 1.5330136777572476, 'addr_state_NE': 1.5329177157986424, 'verification_status_Source Verified': 1.5316244200874691, 'mths_since_last_major_derog': 1.5014338644346426, 'revol_bal': 1.4842587244820489, 'purpose_house': 1.4200368732846809, 'annual_inc': 1.411788066763737, 'dti': 1.3580820245624448, 'revol_util': 1.3431387213056754, 'total_cu_tl': 1.313277509616113, 'earliest_cr_line': 1.2774740226005623, 'purpose_wedding': 1.2676206035373658, 'addr_state_ME': 1.2389826625507752, 'emp_length': 1.23015208079886, 'addr_state_ND': 1.2176282318766618, 'verification_status_joint_Source Verified': 1.21746162184719, 'last_credit_pull_d': 1.2155761978077833, 'delinq_2yrs': 1.203056931316467, 'inq_last_6mths': 1.1700827644410172, 'home_ownership_OWN': 1.1557264940598269, 'last_pymnt_amnt': 1.1515113802060482, 'initial_list_status': 1.128468138127505, 'pub_rec': 1.0729805349528614, 'purpose_renewable_energy': 1.0650906758641836, 'purpose_educational': 1.0559243025772849, 'acc_now_delinq': 1.0219935281968442, 'collections_12_mths_ex_med': 1.014027801963101, 'addr_state_IA': 1.0075443124569217, 'addr_state_ID': 1.005791675253549, 'home_ownership_OTHER': 1.0037427138737391, 'tot_coll_amt': 1.0008889405120973, 'pymnt_plan': 1.0001515259612508}

[197]:
```
corr_dict(X_scaled, corr = .5)
```

```
{'il_util': 4, 'inq_last_12m': 3, 'open_acc_6m': 2, 'inq_fi': 2,
'purpose_credit_card': 1, 'dti_joint': 1, 'mths_since_last_major_derog': 1,
'max_bal_bc': 1, 'total_acc': 1, 'verification_status_Source Verified': 1,
'verification_status_Verified': 1, 'verification_status_joint_Verified': 1,
'mths_since_last_delinq': 1, 'purpose_debt_consolidation': 1, 'open_acc': 1}
```

[198]:
```
corr_list(corr= .6, min_corr = 1)
```

```
{'il_util': ['open_acc_6m', 'max_bal_bc'], 'inq_last_12m': ['inq_fi'],
'open_acc_6m': ['il_util'], 'inq_fi': ['inq_last_12m'], 'purpose_credit_card':
['purpose_debt_consolidation'], 'dti_joint': [], 'mths_since_last_major_derog':
[], 'max_bal_bc': ['il_util'], 'total_acc': ['open_acc'],
'verification_status_Source Verified': [], 'verification_status_Verified': [],
'verification_status_joint_Verified': [], 'mths_since_last_delinq': [],
'purpose_debt_consolidation': ['purpose_credit_card'], 'open_acc':
['total_acc']}
```

[195]:
```
X_scaled.drop(['emp_type_other', 'addr_state_CA', ], axis = 1, inplace = True)
```

[196]:
```
vif(X_scaled)
```

```
115 out of 121 features have a vif < 2.5
{'purpose_debt_consolidation': 24.96761119362853, 'purpose_credit_card':
18.93961785627865, 'purpose_home_improvement': 6.5212061819225475,
'purpose_other': 5.602217767774178, 'purpose_major_purchase': 2.896130530451369,
'il_util': 2.730798750721492, 'total_acc': 2.3412250411725206, 'inq_last_12m':
2.2724305415892028, 'open_acc': 2.2366887915277776, 'purpose_small_business':
2.1766772917631436, 'inq_fi': 2.027958161959543, 'open_acc_6m':
1.969223020722815, 'purpose_medical': 1.9541917795725594, 'grade':
1.882609031023419, 'tot_cur_bal': 1.8817109187706282, 'dti_joint':
1.8138544467538595, 'loan_amnt': 1.7700889155867419,
'verification_status_Verified': 1.6876008414952248, 'max_bal_bc':
1.67059263851143, 'home_ownership_RENT': 1.6649091693054299, 'purpose_moving':
1.612957560969563, 'term': 1.6119723224069213, 'mths_since_last_delinq':
1.6024830947155813, 'verification_status_joint_Verified': 1.5951568502705917,
'purpose_vacation': 1.5330036609346471, 'verification_status_Source Verified':
1.531580136517418, 'mths_since_last_major_derog': 1.501287811129522,
'revol_bal': 1.4842361387733702, 'addr_state_TX': 1.4710102981603153,
'addr_state_NY': 1.4451569706399565, 'purpose_house': 1.4200324341093116,
'annual_inc': 1.411719771951544, 'addr_state_FL': 1.3856905162976025,
'emp_type_Manager': 1.3846289802441962, 'emp_type_Unemployed':
1.375953539616218, 'dti': 1.3579245825891364, 'revol_util': 1.3429068081647426,
'total_cu_tl': 1.3132247928287366, 'emp_type_Technical': 1.291253306726926,
'earliest_cr_line': 1.2774538519434466, 'purpose_wedding': 1.2676155303887011,
'addr_state_IL': 1.2367942449247422, 'emp_length': 1.229640725680606,
'verification_status_joint_Source Verified': 1.2174567422552078,
'last_credit_pull_d': 1.2153732889718927, 'addr_state_PA': 1.2150862304066916,
```

```
'addr_state_OH': 1.2136184480397716, 'addr_state_NJ': 1.211756939901659,
'delinq_2yrs': 1.2030519679157148, 'addr_state_GA': 1.2027119102739965,
'addr_state_NC': 1.1744377442178562, 'addr_state_VA': 1.173599554052189,
'inq_last_6mths': 1.170005112914522, 'addr_state_MI': 1.167534979071377,
'emp_type_Healer': 1.1595810943404903, 'home_ownership_OWN': 1.1557259923910626,
'last_pymnt_amnt': 1.1515071804270436, 'emp_type_Executive': 1.1435806737986498,
'addr_state_MD': 1.1398098047832472, 'addr_state_AZ': 1.1366589905091007,
'addr_state_MA': 1.133407108180576, 'initial_list_status': 1.128389047723271,
'addr_state_WA': 1.127056734443586, 'addr_state_CO': 1.126097731275573,
'emp_type_Director': 1.1185216937061957, 'addr_state_MN': 1.111092139024878,
'addr_state_IN': 1.1080028479183648, 'addr_state_MO': 1.104552350953598,
'emp_type_Education': 1.0999518529880863, 'addr_state_TN': 1.0958630462702927,
'addr_state_CT': 1.0905781695634875, 'addr_state_AL': 1.08680622935313,
'emp_type_Vol': 1.0856810680418014, 'emp_type_Operator': 1.0853554300692798,
'addr_state_NV': 1.0841331667014709, 'addr_state_WI': 1.0838599850181452,
'addr_state_SC': 1.0790555747547514, 'addr_state_LA': 1.0789413239840797,
'emp_type_Analyst': 1.0748841656080348, 'pub_rec': 1.0729507946244885,
'addr_state_OR': 1.0727078127290053, 'emp_type_Service': 1.0708955288083002,
'emp_type_Assistant': 1.06592784508023, 'addr_state_KY': 1.0656549519342704,
'purpose_renewable_energy': 1.065090388165688, 'addr_state_OK':
1.0620347311872813, 'emp_type_Sales': 1.0590093693419453, 'addr_state_KS':
1.0584012271526348, 'purpose_educational': 1.0559240389158238, 'addr_state_AR':
1.0519412080590325, 'addr_state_UT': 1.0447937645886771, 'emp_type_Accountant':
1.0364811525268898, 'addr_state_NM': 1.0364600347939292, 'emp_type_Finance':
1.036374254004344, 'addr_state_WV': 1.035090185088152, 'emp_type_Admin':
1.032202003968641７, 'addr_state_MS': 1.0316747889622462, 'addr_state_HI':
1.0312558285234903, 'addr_state_NH': 1.0299181430529991, 'addr_state_RI':
1.0263658039473804, 'emp_type_Manlab': 1.0222894653635657, 'acc_now_delinq':
1.0219905270476908, 'emp_type_Clerk': 1.0206809614570438, 'addr_state_MT':
1.0188570977775266, 'addr_state_DE': 1.0178363363690994, 'addr_state_DC':
1.0169735661498065, 'addr_state_WY': 1.0158492688032048, 'addr_state_SD':
1.0146824568977704, 'collections_12_mths_ex_med': 1.014024859903138,
'addr_state_VT': 1.0135005534468404, 'addr_state_NE': 1.0093473303132385,
'emp_type_Designer': 1.0091182896478026, 'emp_type_Law': 1.007220539009543,
'emp_type_Clergy': 1.005854682864526, 'addr_state_ME': 1.0052699934125424,
'addr_state_ND': 1.0041975392687317, 'home_ownership_OTHER': 1.003742575348957,
'addr_state_IA': 1.0012945682069592, 'tot_coll_amt': 1.0008888908688367,
'addr_state_ID': 1.000435302095682, 'pymnt_plan': 1.0001514705171868}
```

[203]:
```
X_scaled.drop(['purpose_debt_consolidation'], axis = 1, inplace = True)
```

[204]:
```
vif(X_scaled)
```

```
119 out of 120 features have a vif < 2.5
{'il_util': 2.7307888242024836, 'total_acc': 2.34122363770942, 'inq_last_12m':
2.2724287817188364, 'open_acc': 2.2362154778845387, 'inq_fi': 2.027957789717889,
'open_acc_6m': 1.9692149833036792, 'grade': 1.8825266359843544, 'tot_cur_bal':
1.8816856234174637, 'dti_joint': 1.8138513300300452, 'loan_amnt':
```

1.76378376631423, 'verification_status_Verified': 1.6875362019651317,
'max_bal_bc': 1.6705891495577427, 'home_ownership_RENT': 1.6649046553273368,
'term': 1.6113500489848105, 'mths_since_last_delinq': 1.6024581712519677,
'verification_status_joint_Verified': 1.5951548391405679,
'verification_status_Source Verified': 1.5314865482589934,
'mths_since_last_major_derog': 1.5012617294420394, 'revol_bal':
1.4840021620392303, 'addr_state_TX': 1.471009052033995, 'addr_state_NY':
1.445139520674233, 'annual_inc': 1.4115003680493299, 'addr_state_FL':
1.3856881112765589, 'emp_type_Manager': 1.3845719470330107,
'emp_type_Unemployed': 1.3758843463096149, 'dti': 1.3568266538343439,
'revol_util': 1.3404689847207572, 'total_cu_tl': 1.313216696550245,
'emp_type_Technical': 1.291252864896083, 'earliest_cr_line': 1.2774538426875484,
'addr_state_IL': 1.236793043311825, 'emp_length': 1.2295347307952664,
'verification_status_joint_Source Verified': 1.2174567012313438,
'addr_state_PA': 1.2150812972677942, 'last_credit_pull_d': 1.2140420349666088,
'addr_state_OH': 1.2136099740784634, 'addr_state_NJ': 1.2117561619082107,
'delinq_2yrs': 1.2030420768765697, 'addr_state_GA': 1.2027109402714755,
'addr_state_NC': 1.1744360565205412, 'addr_state_VA': 1.1735991837449828,
'inq_last_6mths': 1.1699938360933184, 'addr_state_MI': 1.1675327406173073,
'emp_type_Healer': 1.1595726960576076, 'home_ownership_OWN': 1.1556346606602725,
'last_pymnt_amnt': 1.1513341738552931, 'emp_type_Executive': 1.1435483886317626,
'addr_state_MD': 1.139809707927233, 'addr_state_AZ': 1.1366565574300957,
'addr_state_MA': 1.1333957445531833, 'initial_list_status': 1.1282615470377386,
'addr_state_WA': 1.1270557903662477, 'addr_state_CO': 1.1260946762503712,
'purpose_credit_card': 1.1221073903675312, 'emp_type_Director':
1.1184903635699266, 'addr_state_MN': 1.111089812879824, 'addr_state_IN':
1.1079998719019568, 'addr_state_MO': 1.1045512858739144, 'emp_type_Education':
1.099945224104737, 'addr_state_TN': 1.0958594251080822, 'addr_state_CT':
1.0905774354390236, 'addr_state_AL': 1.0868059692953485,
'purpose_home_improvement': 1.0859889095512643, 'emp_type_Vol':
1.0856756631964963, 'emp_type_Operator': 1.0853548688948504, 'addr_state_NV':
1.084133153852521, 'addr_state_WI': 1.0838585857943173, 'purpose_other':
1.0792270558935455, 'addr_state_SC': 1.0790555482931103, 'addr_state_LA':
1.0789353312786993, 'emp_type_Analyst': 1.0748773175088338, 'pub_rec':
1.072882602580039, 'addr_state_OR': 1.072704089026199, 'emp_type_Service':
1.0708780679243441, 'emp_type_Assistant': 1.065852142672351, 'addr_state_KY':
1.0656546322556582, 'addr_state_OK': 1.062033248172776, 'emp_type_Sales':
1.0590011240943042, 'addr_state_KS': 1.058401215690259, 'addr_state_AR':
1.051938265638769, 'addr_state_UT': 1.0447901011894005,
'purpose_small_business': 1.0400080197076909, 'emp_type_Accountant':
1.0364722234800476, 'addr_state_NM': 1.0364568579215143, 'emp_type_Finance':
1.0363311115149034, 'addr_state_WV': 1.0350897034065472, 'emp_type_Admin':
1.0321939421946484, 'addr_state_MS': 1.0316741156182487, 'addr_state_HI':
1.0312541185818314, 'addr_state_NH': 1.0299139919431572,
'purpose_major_purchase': 1.029421341791009, 'addr_state_RI': 1.02636078843678,
'emp_type_Manlab': 1.0222840437021758, 'acc_now_delinq': 1.0219903082614925,
'emp_type_Clerk': 1.0206739950427475, 'purpose_moving': 1.019002022618076,
'addr_state_MT': 1.01885669067497, 'purpose_medical': 1.0178710196559664,

```
'addr_state_DE': 1.0178345137750904, 'addr_state_DC': 1.0169695912928167,
'addr_state_WY': 1.0158399393098407, 'addr_state_SD': 1.0146792946358334,
'collections_12_mths_ex_med': 1.0140236676392183, 'purpose_vacation':
1.0138077193121813, 'addr_state_VT': 1.0135003335191375, 'purpose_wedding':
1.012731673605749, 'purpose_educational': 1.0115037506358027, 'purpose_house':
1.010585488111, 'addr_state_NE': 1.0093460943965555, 'emp_type_Designer':
1.0091124645236929, 'emp_type_Law': 1.0072052221037187, 'emp_type_Clergy':
1.005846092220493, 'addr_state_ME': 1.00526910748636, 'addr_state_ND':
1.0041965609613241, 'home_ownership_OTHER': 1.003740834803285,
'purpose_renewable_energy': 1.0016474231528574, 'addr_state_IA':
1.0012934740261756, 'tot_coll_amt': 1.0008888906939468, 'addr_state_ID':
1.0004330794741287, 'pymnt_plan': 1.000151470516797}
```

While I wanted all of the features to be under 2.5, The dataset is still large and running the vif function takes over 10 minutes to run. I will first perform feature selection and then I will return to vif.

[206]:
```
y_binary = np.where(loan['loan_status'].isin(['Fully Paid', 'Current',␣
↪'Issued']), 1, 0)
```

##

Lasso Regularization

Lasso Regularization is a technique for variable selection that uses regression to evaluate the effect that features have on a target variable. The idea is to add a penalty term that contains the coefficient or slope of the variables with respect to the target variable multiplied by parameter.

$$Loss(\beta_1, ...\beta_n) = SSD + \alpha|\beta|$$

Where β is the coefficient of the feature, SSD is the sum of squared distances of the point to the regression line and α is the penalty's parameter. If there is more than a single feature, then we can sum the coefficients:

$$Loss(\beta_1, ...\beta_n) = SSD + \alpha * \sum_{i=1}^{n} |\beta_i|$$

Lasso aims to reduce the loss in equation (3). If a feature is not important, then changing its slope will not move the regression line close enough to the data points to decrease the loss function with a non-zero value. If a feature is important, the regression line will move towards the data and minimize the SSD faster than it increases the penalty term. Since only features with non-zero coefficients are meaningful, I can discard all the features with a coefficient of zero.

First we will run a gridsearch to find the best value of α, and then we will use the coeffecients from that value of α to determine which features stay in the model.

[207]:
```
# Create Lasso model
lasso = Lasso(max_iter = 50000)

# Define hyperparameter grid with a value less than 00.5 since that was the be
params = {'alpha': np.linspace(.0001, .01, 20)}

# Perform grid search
```

```
grid_search = GridSearchCV(estimator=lasso, param_grid=params, cv=8)

# fit the gridsearch of parameters to the data
grid_search.fit(X_scaled, y_binary)

# Print best hyperparameters
print("Best hyperparameters: ", grid_search.best_params_)
```

```
Best hyperparameters:  {'alpha': 0.01}
```

[208]:
```
# Find the coefficients of lasso regularization
lasso = Lasso(alpha = .001)
# Fit the lasso regularization to the data
lasso.fit(X_scaled, y_binary)
# Create a dictionary of all the features and their corresponding lasso␣
↪coefficients
lasso_dict = {X_scaled.columns[i]:lasso.coef_[i] for i in range(len(X_scaled.
↪columns)) if list(lasso.coef_)[i] != 0}
# Create a list of all the features that don't have a lasso coefficient of zero
lasso_features = [X_scaled.columns[i] for i in range(len(X_scaled.columns)) if␣
↪list(lasso.coef_)[i] != 0]
# Print the features and coefficients
print(lasso_dict)
# Print the number of features are left from lasso
print(len(lasso_dict))
```

```
{'loan_amnt': -0.010391316872062236, 'term': 0.002948502470419479, 'grade':
0.03788254325234942, 'annual_inc': 0.0022825117307152722, 'pymnt_plan':
-0.00023065455231244378, 'inq_last_6mths': -0.016965471271241, 'pub_rec':
0.00328094929656237, 'revol_bal': 0.00017543217177538116, 'revol_util':
-0.005030065433174285, 'total_acc': -0.001725540484437378,
'initial_list_status': 0.010934774737318688, 'last_pymnt_amnt':
0.04433076049762753, 'last_credit_pull_d': 0.051610620607713674,
'collections_12_mths_ex_med': 0.0004269000786191071,
'mths_since_last_major_derog': -0.003163609410562672, 'dti_joint':
0.0007302915041974671, 'tot_cur_bal': 0.0021794962954110103, 'open_acc_6m':
0.0023301194338613833, 'il_util': 0.004372566069389636, 'max_bal_bc':
0.0015211874387965623, 'inq_last_12m': 0.001700906513944068,
'emp_type_Accountant': 0.0014560825330323078, 'emp_type_Admin':
0.0011180935599351067, 'emp_type_Analyst': 0.0020839310189119495,
'emp_type_Assistant': 0.001716231097294549, 'emp_type_Clergy':
0.0004086759716117618, 'emp_type_Clerk': 0.0003684051643701604,
'emp_type_Designer': 8.386470181023748e-05, 'emp_type_Director':
0.003250926563873108, 'emp_type_Education': 0.001905001710557324,
'emp_type_Executive': 0.0028161364298661838, 'emp_type_Healer':
0.001802276485713458, 'emp_type_Manager': 0.00534780536249633,
'emp_type_Operator': 0.00017582019867604627, 'emp_type_Technical':
0.0017201398884130043, 'emp_type_Vol': 0.0013373643692269049,
```

'home_ownership_OTHER': -9.787280568326191e-05, 'home_ownership_RENT':
-0.0034307500430190156, 'verification_status_Source Verified':
0.0006960764915860841, 'verification_status_Verified': -0.002840233859866113,
'purpose_credit_card': 0.0010696145226728286, 'purpose_major_purchase':
0.0006338286983715384, 'purpose_other': 0.00030442720791727033,
'purpose_small_business': -0.002592371167640805\3, 'purpose_wedding':
0.0008600910780396444, 'addr_state_AL': -2.717313790854032e-05, 'addr_state_CO':
0.00023737701126124214, 'addr_state_DC': 0.0003228253473749971, 'addr_state_FL':
-0.0008238996713227565, 'addr_state_IL': 0.0018622525159464468, 'addr_state_KS':
0.0003714578370217212, 'addr_state_ME': 0.0005063581439605373, 'addr_state_MS':
0.0004445402559083527, 'addr_state_ND': 0.00031819850705387936, 'addr_state_NE':
0.0007900323376744395, 'addr_state_NH': 1.8891369076422682e-05, 'addr_state_NV':
-0.0006201602888351354, 'addr_state_NY': -0.000804910934598037, 'addr_state_SC':
0.0004239016727849242, 'addr_state_TX': 0.0011805290982262464, 'addr_state_VA':
-0.00022906652147532304}
61

[211]:
```
df = X_scaled[lasso_dict.keys()]
```

[212]:
```
vif(df)
```

60 out of 61 features have a vif < 2.5
{'il_util': 2.503648536830864, 'open_acc_6m': 1.9535214353509527, 'tot_cur_bal':
1.770654329297814, 'grade': 1.7439133806542608, 'loan_amnt': 1.7138957829652597,
'max_bal_bc': 1.6615584854813974, 'inq_last_12m': 1.6414880361328827,
'verification_status_Verified': 1.625458267723455, 'term': 1.5872880178470525,
'verification_status_Source Verified': 1.523078231850744, 'revol_bal':
1.4016325218077512, 'annual_inc': 1.330297515878795, 'home_ownership_RENT':
1.3073547671221735, 'total_acc': 1.2448280761753376, 'revol_util':
1.2440126800569045, 'emp_type_Manager': 1.2293581030145195,
'last_credit_pull_d': 1.1823828739055637, 'emp_type_Technical':
1.1632605142157963, 'inq_last_6mths': 1.1554307323054438, 'last_pymnt_amnt':
1.140772579699193, 'initial_list_status': 1.1210513382502567,
'emp_type_Executive': 1.1074575667950384, 'emp_type_Healer': 1.0974140074627614,
'purpose_credit_card': 1.090279220777798, 'mths_since_last_major_derog':
1.0872113114911621, 'emp_type_Director': 1.0761671055142379,
'emp_type_Education': 1.0593520442957665, 'purpose_other': 1.0584937234060041,
'addr_state_NY': 1.0554819306031729, 'addr_state_TX': 1.0515488522453607,
'emp_type_Vol': 1.0511714498170022, 'emp_type_Operator': 1.0502161947572395,
'pub_rec': 1.0497176483600994, 'emp_type_Analyst': 1.0460144044523427,
'emp_type_Assistant': 1.0392069571630331, 'addr_state_FL': 1.0390726711554674,
'purpose_small_business': 1.0315205970379795, 'addr_state_IL':
1.0262364921930622, 'emp_type_Accountant': 1.0217594682580176,
'purpose_major_purchase': 1.0206314648766086, 'addr_state_VA':
1.020258725151585, 'emp_type_Admin': 1.0192132595134875, 'addr_state_CO':
1.0133330322712164, 'emp_type_Clerk': 1.0117102129881905, 'addr_state_AL':
1.0112905135137134, 'collections_12_mths_ex_med': 1.010775638772582,
'purpose_wedding': 1.0107579856757944, 'addr_state_NV': 1.0095052605905286,

```
'addr_state_SC': 1.0094403966131993, 'addr_state_KS': 1.0067548263507855,
'emp_type_Designer': 1.0054892655608172, 'addr_state_MS': 1.005021981570084,
'emp_type_Clergy': 1.0034703330142034, 'addr_state_DC': 1.0034678554709804,
'addr_state_NH': 1.0032999442711354, 'dti_joint': 1.0031683280406816,
'home_ownership_OTHER': 1.0030594050709054, 'addr_state_NE': 1.0015282820188407,
'addr_state_ME': 1.0014568397523846, 'addr_state_ND': 1.0009243078467163,
'pymnt_plan': 1.000111548764211}
```

[213]:
```
df.to_csv('./data/interim/wrangled', index = False)
```