In [300]:
```
import os

import pandas as pd

from matplotlib import pyplot as plt

import numpy as np


from sklearn.ensemble import GradientBoostingClassifier
```

In [301]:
```
# Question 3: Reading the LendingClub dataset

dataset = pd.read_csv('F:/Seagate_Sync/VOL/VOL00/VIVEK/Big Data Analytics Cert
ification/BDA 102/Assignment 7/LoanStats_2018Q1_Assg_7.csv', dtype = 'unicode'
, skiprows=1)
```

In [302]:
```
# Question 3: Reading the column names to verify columns: Amount requested, In
terest rates etc. columns

list(dataset.columns.values)
```

```
Out[302]: ['id',
           'member_id',
           'loan_amnt',
           'funded_amnt',
           'funded_amnt_inv',
           'term',
           'int_rate',
           'installment',
           'grade',
           'sub_grade',
           'emp_title',
           'emp_length',
           'home_ownership',
           'annual_inc',
           'verification_status',
           'issue_d',
           'loan_status',
           'pymnt_plan',
           'url',
           'desc',
           'purpose',
           'title',
           'zip_code',
           'addr_state',
           'dti',
           'delinq_2yrs',
           'earliest_cr_line',
           'inq_last_6mths',
           'mths_since_last_delinq',
           'mths_since_last_record',
           'open_acc',
           'pub_rec',
           'revol_bal',
           'revol_util',
           'total_acc',
           'initial_list_status',
           'out_prncp',
           'out_prncp_inv',
           'total_pymnt',
           'total_pymnt_inv',
           'total_rec_prncp',
           'total_rec_int',
           'total_rec_late_fee',
           'recoveries',
           'collection_recovery_fee',
           'last_pymnt_d',
           'last_pymnt_amnt',
           'next_pymnt_d',
           'last_credit_pull_d',
           'collections_12_mths_ex_med',
           'mths_since_last_major_derog',
           'policy_code',
           'application_type',
           'annual_inc_joint',
           'dti_joint',
           'verification_status_joint',
           'acc_now_delinq',
```

```
'tot_coll_amt',
'tot_cur_bal',
'open_acc_6m',
'open_act_il',
'open_il_12m',
'open_il_24m',
'mths_since_rcnt_il',
'total_bal_il',
'il_util',
'open_rv_12m',
'open_rv_24m',
'max_bal_bc',
'all_util',
'total_rev_hi_lim',
'inq_fi',
'total_cu_tl',
'inq_last_12m',
'acc_open_past_24mths',
'avg_cur_bal',
'bc_open_to_buy',
'bc_util',
'chargeoff_within_12_mths',
'delinq_amnt',
'mo_sin_old_il_acct',
'mo_sin_old_rev_tl_op',
'mo_sin_rcnt_rev_tl_op',
'mo_sin_rcnt_tl',
'mort_acc',
'mths_since_recent_bc',
'mths_since_recent_bc_dlq',
'mths_since_recent_inq',
'mths_since_recent_revol_delinq',
'num_accts_ever_120_pd',
'num_actv_bc_tl',
'num_actv_rev_tl',
'num_bc_sats',
'num_bc_tl',
'num_il_tl',
'num_op_rev_tl',
'num_rev_accts',
'num_rev_tl_bal_gt_0',
'num_sats',
'num_tl_120dpd_2m',
'num_tl_30dpd',
'num_tl_90g_dpd_24m',
'num_tl_op_past_12m',
'pct_tl_nvr_dlq',
'percent_bc_gt_75',
'pub_rec_bankruptcies',
'tax_liens',
'tot_hi_cred_lim',
'total_bal_ex_mort',
'total_bc_limit',
'total_il_high_credit_limit',
'revol_bal_joint',
'sec_app_earliest_cr_line',
'sec_app_inq_last_6mths',
```

```
 'sec_app_mort_acc',
 'sec_app_open_acc',
 'sec_app_revol_util',
 'sec_app_open_act_il',
 'sec_app_num_rev_accts',
 'sec_app_chargeoff_within_12_mths',
 'sec_app_collections_12_mths_ex_med',
 'sec_app_mths_since_last_major_derog',
 'hardship_flag',
 'hardship_type',
 'hardship_reason',
 'hardship_status',
 'deferral_term',
 'hardship_amount',
 'hardship_start_date',
 'hardship_end_date',
 'payment_plan_start_date',
 'hardship_length',
 'hardship_dpd',
 'hardship_loan_status',
 'orig_projected_additional_accrued_interest',
 'hardship_payoff_balance_amount',
 'hardship_last_payment_amount',
 'disbursement_method',
 'debt_settlement_flag',
 'debt_settlement_flag_date',
 'settlement_status',
 'settlement_date',
 'settlement_amount',
 'settlement_percentage',
 'settlement_term']
```

In [303]:
```
dataset.head(3)
```

Out[303]:

| | id | member_id | loan_amnt | funded_amnt | funded_amnt_inv | term | int_rate | installı |
|---|---|---|---|---|---|---|---|---|
| **0** | NaN | NaN | 20000 | 20000 | 20000 | 36 months | 10.41% | 649.21 |
| **1** | NaN | NaN | 11000 | 11000 | 11000 | 36 months | 7.34% | 341.37 |
| **2** | NaN | NaN | 12000 | 12000 | 12000 | 36 months | 6.07% | 365.45 |

3 rows × 145 columns

In [304]:
```
dataset['int_rate'].dtype
```

Out[304]:
```
dtype('O')
```

In [305]:
```
# Question 4: Cleaning the data and dropping first 2 columns: ID and Member_ID

dataset.drop(['id','member_id'],1, inplace=True)
```

In [306]:
```
# Question 4: Converting interest rate to type float

dataset.int_rate = pd.Series(dataset.int_rate).str.replace('%', '').astype(flo
at)
```

In [307]:
```
# Question 4: Verifying the conversion of int_rate variable.

dataset['int_rate'].dtype
```

Out[307]:
```
dtype('float64')
```

In [308]:
```
# Question 4: Value count for "url" variable

dataset.url.value_counts()
```

Out[308]:
```
Series([], Name: url, dtype: int64)
```

In [309]:
```
# Question 4: Value count for "desc" variable

dataset.desc.value_counts()
```

Out[309]:
```
Series([], Name: desc, dtype: int64)
```

In [310]:
```
# Question 4: Dropping the url and desc variables since they do not contain an
y values.

dataset.drop(['url', 'desc'],1, inplace=True)
```

In [311]:
```
# Question 4: Value count for emp_length variable

dataset.emp_length.value_counts()
```

Out[311]:
```
10+ years    35706
2 years      10191
3 years       9179
< 1 year      7339
1 year        7169
4 years       6918
5 years       6815
6 years       4716
7 years       4002
8 years       3278
9 years       3123
Name: emp_length, dtype: int64
```

In [312]:
```
# Question 4: Converting "revol_util" to float similar to int_rate variable

dataset.revol_util = pd.Series(dataset.revol_util).str.replace('%', '').astype
(float)
```

In [313]:
```
# Question 4: Cleaning and dropping few more post loan variables because of ze
ro values

dataset.drop(['out_prncp_inv','total_rec_late_fee',
         'recoveries','collection_recovery_fee',
         'total_pymnt_inv' ],1, inplace=True)
```

In [314]:
```
# Question 4:

dataset.iloc[:5,30:42]
```

Out[314]:

| | total_acc | initial_list_status | out_prncp | total_pymnt | total_rec_prncp | total_rec_int | las |
|---|---|---|---|---|---|---|---|
| **0** | 9 | w | 18560.45 | 1884.02 | 1439.55 | 444.47 | Ju |
| **1** | 33 | w | 10450.15 | 673.77 | 549.85 | 123.92 | Ju |
| **2** | 23 | w | 11388.96 | 722.81 | 611.04 | 111.77 | Ju |
| **3** | 29 | w | 32678.15 | 3436.3 | 2321.85 | 1114.45 | Ju |
| **4** | 16 | w | 19480.21 | 826.28 | 519.79 | 306.49 | Ju |

In [315]:
```
# Question 4: Value count for "collections_12_mths_ex_med" variable

dataset.collections_12_mths_ex_med.value_counts()
```

Out[315]:
```
0    106216
1      1525
2       102
3        13
4         6
8         1
5         1
Name: collections_12_mths_ex_med, dtype: int64
```

In [316]:
```
# Question 4: Value count for "acc_now_delinq" variable

dataset.acc_now_delinq.value_counts()
```

Out[316]:
```
0    107838
1        26
Name: acc_now_delinq, dtype: int64
```

In [317]:
```
# Question 4: Value count for "chargeoff_within_12_mths" variable

dataset.chargeoff_within_12_mths.value_counts()
```

Out[317]:
```
0    107194
1       618
2        37
3         9
4         3
7         1
9         1
6         1
Name: chargeoff_within_12_mths, dtype: int64
```

In [318]:
```
# Question 4: Value count for "num_tl_120dpd_2m" variable

dataset.num_tl_120dpd_2m.value_counts()
```

Out[318]:
```
0    104356
Name: num_tl_120dpd_2m, dtype: int64
```

In [319]:
```
# Question 4: Value count for "num_tl_30dpd" variable

dataset.num_tl_30dpd.value_counts()
```

Out[319]:
```
0    107841
1        23
Name: num_tl_30dpd, dtype: int64
```

In [320]:
```
# Question 4: Value count for "hardship_flag" variable

dataset.hardship_flag.value_counts()
```

Out[320]:
```
N    107864
Name: hardship_flag, dtype: int64
```

In [321]:
```
# Question 4: Dropping more variables based on their counts

dataset.drop(['hardship_flag','hardship_type',
         'hardship_reason','hardship_status',
         'deferral_term','hardship_amount','hardship_start_date','hardship_end
_date','payment_plan_start_date','hardship_length','hardship_dpd','hardship_lo
an_status','orig_projected_additional_accrued_interest','hardship_payoff_balan
ce_amount','hardship_last_payment_amount'],1, inplace=True)
```

In [322]:
```
# Question 4: Value count for "settlement_term" variable

dataset.settlement_term.value_counts()
```

Out[322]:
```
10    1
18    1
12    1
Name: settlement_term, dtype: int64
```

In [323]:
```
# Question 4: Dropping more variables based on their counts

dataset.drop(['debt_settlement_flag_date','settlement_status',
         'settlement_date','settlement_amount',
         'settlement_percentage','settlement_term' ],1, inplace=True)
```

In [324]:
```
#Questions 5: Checking "annual_inc" data type

dataset['annual_inc'].dtype
```

Out[324]:
```
dtype('O')
```

In [325]:
```
# Question 5: Converting "annual_inc" data type to float

dataset.annual_inc = pd.Series(dataset.annual_inc).str.replace('%', '').astype
(float)
```

In [326]:
```
# Question 5: Verifying "annual_inc" data type

dataset['annual_inc'].dtype
```

Out[326]:
```
dtype('float64')
```

In [327]:
```
# Question 5: Dividing "annual_inc" by 12

dataset['New_Col'] = (dataset['annual_inc']/12)
```

In [328]:
```
# Question 5: Checking "New_Col" data type

dataset['New_Col'].dtype
```

Out[328]:
```
dtype('float64')
```

In [329]:
```
# Question 5: Listing column names to verify "New_Col" is added as a new colum
n

list(dataset.columns.values)
```

```
Out[329]: ['loan_amnt',
           'funded_amnt',
           'funded_amnt_inv',
           'term',
           'int_rate',
           'installment',
           'grade',
           'sub_grade',
           'emp_title',
           'emp_length',
           'home_ownership',
           'annual_inc',
           'verification_status',
           'issue_d',
           'loan_status',
           'pymnt_plan',
           'purpose',
           'title',
           'zip_code',
           'addr_state',
           'dti',
           'delinq_2yrs',
           'earliest_cr_line',
           'inq_last_6mths',
           'mths_since_last_delinq',
           'mths_since_last_record',
           'open_acc',
           'pub_rec',
           'revol_bal',
           'revol_util',
           'total_acc',
           'initial_list_status',
           'out_prncp',
           'total_pymnt',
           'total_rec_prncp',
           'total_rec_int',
           'last_pymnt_d',
           'last_pymnt_amnt',
           'next_pymnt_d',
           'last_credit_pull_d',
           'collections_12_mths_ex_med',
           'mths_since_last_major_derog',
           'policy_code',
           'application_type',
           'annual_inc_joint',
           'dti_joint',
           'verification_status_joint',
           'acc_now_delinq',
           'tot_coll_amt',
           'tot_cur_bal',
           'open_acc_6m',
           'open_act_il',
           'open_il_12m',
           'open_il_24m',
           'mths_since_rcnt_il',
           'total_bal_il',
           'il_util',
```

```
'open_rv_12m',
'open_rv_24m',
'max_bal_bc',
'all_util',
'total_rev_hi_lim',
'inq_fi',
'total_cu_tl',
'inq_last_12m',
'acc_open_past_24mths',
'avg_cur_bal',
'bc_open_to_buy',
'bc_util',
'chargeoff_within_12_mths',
'delinq_amnt',
'mo_sin_old_il_acct',
'mo_sin_old_rev_tl_op',
'mo_sin_rcnt_rev_tl_op',
'mo_sin_rcnt_tl',
'mort_acc',
'mths_since_recent_bc',
'mths_since_recent_bc_dlq',
'mths_since_recent_inq',
'mths_since_recent_revol_delinq',
'num_accts_ever_120_pd',
'num_actv_bc_tl',
'num_actv_rev_tl',
'num_bc_sats',
'num_bc_tl',
'num_il_tl',
'num_op_rev_tl',
'num_rev_accts',
'num_rev_tl_bal_gt_0',
'num_sats',
'num_tl_120dpd_2m',
'num_tl_30dpd',
'num_tl_90g_dpd_24m',
'num_tl_op_past_12m',
'pct_tl_nvr_dlq',
'percent_bc_gt_75',
'pub_rec_bankruptcies',
'tax_liens',
'tot_hi_cred_lim',
'total_bal_ex_mort',
'total_bc_limit',
'total_il_high_credit_limit',
'revol_bal_joint',
'sec_app_earliest_cr_line',
'sec_app_inq_last_6mths',
'sec_app_mort_acc',
'sec_app_open_acc',
'sec_app_revol_util',
'sec_app_open_act_il',
'sec_app_num_rev_accts',
'sec_app_chargeoff_within_12_mths',
'sec_app_collections_12_mths_ex_med',
'sec_app_mths_since_last_major_derog',
'disbursement_method',
```

```
 'debt_settlement_flag',
 'New_Col']
```

In [330]:
```
# Question 5: Checking "revol_bal" data type

dataset['revol_bal'].dtype
```

Out[330]:
```
dtype('O')
```

In [331]:
```
# Question 5: Changing "revol_bal" data type to float

dataset.revol_bal = pd.Series(dataset.revol_bal).astype(float)
```

In [332]:
```
# Question 5: Verifying "revol_bal" data type

dataset['revol_bal'].dtype
```

Out[332]:
```
dtype('float64')
```

In [333]:
```
# Question 5: Getting rid of zeros from "revol_bal" by copying itself to the r
ow with values greater than 0

dataset = dataset[dataset['revol_bal'] > 0]
```

In [334]:
```
# Question 5: Doing a value count of "revol_bal" variable.

dataset.revol_bal.value_counts()
```

```
Out[334]: 8.0         22
          5.0         20
          4.0         18
          5124.0      18
          10.0        18
          12.0        17
          3954.0      15
          25.0        15
          3994.0      15
          3327.0      14
          50.0        14
          2648.0      14
          2.0         14
          3885.0      14
          4029.0      14
          6702.0      13
          22.0        13
          11371.0     13
          7512.0      13
          4239.0      13
          3984.0      13
          5198.0      13
          5554.0      13
          3132.0      13
          5238.0      13
          2878.0      13
          12142.0     13
          6224.0      12
          5401.0      12
          4607.0      12
                      ..
          31063.0      1
          31909.0      1
          57422.0      1
          43722.0      1
          47808.0      1
          69071.0      1
          19986.0      1
          44745.0      1
          26749.0      1
          105863.0     1
          14034.0      1
          29856.0      1
          20429.0      1
          42448.0      1
          44729.0      1
          14803.0      1
          26017.0      1
          29054.0      1
          48597.0      1
          52925.0      1
          40574.0      1
          36547.0      1
          62022.0      1
          21788.0      1
          22978.0      1
          66333.0      1
```

```
206.0        1
1360.0       1
36545.0      1
43541.0      1
Name: revol_bal, Length: 37682, dtype: int64
```

In [335]:
```
# Question 5: Divinding "New_Col" with "revol_bal" to get "debt_to_income_rati
o"

dataset['debt_to_income_ratio'] = (dataset['New_Col']/dataset['revol_bal'])
```

In [336]:
```
# Question 5: Listing column names to verify "debt_to_income_ratio" is added a
s a new column

list(dataset.columns.values)
```

```
Out[336]: ['loan_amnt',
           'funded_amnt',
           'funded_amnt_inv',
           'term',
           'int_rate',
           'installment',
           'grade',
           'sub_grade',
           'emp_title',
           'emp_length',
           'home_ownership',
           'annual_inc',
           'verification_status',
           'issue_d',
           'loan_status',
           'pymnt_plan',
           'purpose',
           'title',
           'zip_code',
           'addr_state',
           'dti',
           'delinq_2yrs',
           'earliest_cr_line',
           'inq_last_6mths',
           'mths_since_last_delinq',
           'mths_since_last_record',
           'open_acc',
           'pub_rec',
           'revol_bal',
           'revol_util',
           'total_acc',
           'initial_list_status',
           'out_prncp',
           'total_pymnt',
           'total_rec_prncp',
           'total_rec_int',
           'last_pymnt_d',
           'last_pymnt_amnt',
           'next_pymnt_d',
           'last_credit_pull_d',
           'collections_12_mths_ex_med',
           'mths_since_last_major_derog',
           'policy_code',
           'application_type',
           'annual_inc_joint',
           'dti_joint',
           'verification_status_joint',
           'acc_now_delinq',
           'tot_coll_amt',
           'tot_cur_bal',
           'open_acc_6m',
           'open_act_il',
           'open_il_12m',
           'open_il_24m',
           'mths_since_rcnt_il',
           'total_bal_il',
           'il_util',
```

```
'open_rv_12m',
'open_rv_24m',
'max_bal_bc',
'all_util',
'total_rev_hi_lim',
'inq_fi',
'total_cu_tl',
'inq_last_12m',
'acc_open_past_24mths',
'avg_cur_bal',
'bc_open_to_buy',
'bc_util',
'chargeoff_within_12_mths',
'delinq_amnt',
'mo_sin_old_il_acct',
'mo_sin_old_rev_tl_op',
'mo_sin_rcnt_rev_tl_op',
'mo_sin_rcnt_tl',
'mort_acc',
'mths_since_recent_bc',
'mths_since_recent_bc_dlq',
'mths_since_recent_inq',
'mths_since_recent_revol_delinq',
'num_accts_ever_120_pd',
'num_actv_bc_tl',
'num_actv_rev_tl',
'num_bc_sats',
'num_bc_tl',
'num_il_tl',
'num_op_rev_tl',
'num_rev_accts',
'num_rev_tl_bal_gt_0',
'num_sats',
'num_tl_120dpd_2m',
'num_tl_30dpd',
'num_tl_90g_dpd_24m',
'num_tl_op_past_12m',
'pct_tl_nvr_dlq',
'percent_bc_gt_75',
'pub_rec_bankruptcies',
'tax_liens',
'tot_hi_cred_lim',
'total_bal_ex_mort',
'total_bc_limit',
'total_il_high_credit_limit',
'revol_bal_joint',
'sec_app_earliest_cr_line',
'sec_app_inq_last_6mths',
'sec_app_mort_acc',
'sec_app_open_acc',
'sec_app_revol_util',
'sec_app_open_act_il',
'sec_app_num_rev_accts',
'sec_app_chargeoff_within_12_mths',
'sec_app_collections_12_mths_ex_med',
'sec_app_mths_since_last_major_derog',
'disbursement_method',
```

```
'debt_settlement_flag',
'New_Col',
'debt_to_income_ratio']
```

In [337]:
```
# Question 5: Dropping more columns that will not be used in analysis

dataset.drop(['sub_grade','emp_title','issue_d','pymnt_plan','zip_code','mths_
since_last_delinq','mths_since_last_record',
            'initial_list_status','out_prncp',
 'total_pymnt',
 'total_rec_prncp',
 'total_rec_int',
 'last_pymnt_d',
 'last_pymnt_amnt',
 'next_pymnt_d',
 'collections_12_mths_ex_med',
 'mths_since_last_major_derog',
 'policy_code',
 'application_type',
 'annual_inc_joint',
 'dti_joint',
 'verification_status_joint',
 'acc_now_delinq',
 'tot_coll_amt',
 'tot_cur_bal',
 'open_acc_6m',
 'open_act_il',
 'open_il_12m',
 'open_il_24m',
 'mths_since_rcnt_il',
 'total_bal_il',
 'il_util',
 'open_rv_12m',
 'open_rv_24m',
 'max_bal_bc',
 'all_util',
 'total_rev_hi_lim',
 'inq_fi',
 'total_cu_tl',
 'inq_last_12m',
 'acc_open_past_24mths',
 'avg_cur_bal',
 'bc_open_to_buy',
 'bc_util',
 'chargeoff_within_12_mths',
 'mo_sin_old_il_acct',
 'mo_sin_old_rev_tl_op',
 'mo_sin_rcnt_rev_tl_op',
 'mo_sin_rcnt_tl',
 'mths_since_recent_bc',
 'mths_since_recent_bc_dlq',
 'mths_since_recent_inq',
 'mths_since_recent_revol_delinq',
 'num_accts_ever_120_pd',
 'num_actv_bc_tl',
 'num_actv_rev_tl',
 'num_bc_sats',
 'num_bc_tl',
 'num_il_tl',
 'num_op_rev_tl',
```

```
'num_rev_accts',
'num_rev_tl_bal_gt_0',
'num_sats',
'num_tl_120dpd_2m',
'num_tl_30dpd',
'num_tl_90g_dpd_24m',
'num_tl_op_past_12m',
'pct_tl_nvr_dlq',
'percent_bc_gt_75',
'tot_hi_cred_lim',
'total_bal_ex_mort',
'total_bc_limit',
'total_il_high_credit_limit',
'revol_bal_joint',
'sec_app_earliest_cr_line',
'sec_app_inq_last_6mths',
'sec_app_mort_acc',
'sec_app_open_acc',
'sec_app_revol_util',
'sec_app_open_act_il',
'sec_app_num_rev_accts',
'sec_app_chargeoff_within_12_mths',
'sec_app_collections_12_mths_ex_med',
'sec_app_mths_since_last_major_derog',
'disbursement_method',
'debt_settlement_flag',
'New_Col', ],1, inplace=True)
```

In [338]:
```
# Question 6: Counting the values of "debt_to_income_ratio" variable

dataset.debt_to_income_ratio.value_counts()
```

```
Out[338]: 0.000000     248
          1.082251      11
          0.621891      11
          0.511247       9
          0.868056       9
          1.700680       9
          0.900901       8
          2.252252       8
          0.548246       8
          0.530110       8
          0.367107       8
          0.414594       8
          0.750751       8
          4.166667       7
          0.514403       7
          1.282051       7
          0.946970       7
          0.586854       7
          1.096491       7
          1.666667       7
          0.578704       7
          0.841751       7
          1.488095       7
          0.718391       7
          2.525253       7
          0.372856       7
          0.825083       7
          0.608273       7
          83.333333      7
          0.793651       7
                       ...
          0.029875       1
          0.456268       1
          0.353425       1
          0.322741       1
          0.647228       1
          0.408426       1
          11.239563      1
          1.070890       1
          0.443646       1
          2.790115       1
          0.629479       1
          0.125978       1
          0.630597       1
          0.349443       1
          0.364917       1
          1.966703       1
          0.172578       1
          1.239926       1
          0.162549       1
          0.252474       1
          0.311236       1
          0.111522       1
          0.225139       1
          0.421784       1
          0.157729       1
          0.341155       1
```

```
0.661454       1
0.948702       1
0.411388       1
0.176814       1
Name: debt_to_income_ratio, Length: 97004, dtype: int64
```

In [339]:
```
len(dataset.debt_to_income_ratio)
```

Out[339]:
```
106610
```

In [340]:
```
# Question 6: Saving changes to "cleaned" CSV file

dataset.to_csv("F:/Seagate_Sync/VOL/VOL00/VIVEK/Big Data Analytics Certificati
on/BDA 102/Assignment 7/cleaned_LoanStats.csv", index=False)
```

In [341]:
```
# Question 6: loading the saved output

dataset = pd.read_csv('F:/Seagate_Sync/VOL/VOL00/VIVEK/Big Data Analytics Cert
ification/BDA 102/Assignment 7/cleaned_LoanStats.csv')
print(dataset.shape)
dataset.head()
```

```
(106610, 30)
```

Out[341]:

| | loan_amnt | funded_amnt | funded_amnt_inv | term | int_rate | installment | grade | emp_ |
|---|---|---|---|---|---|---|---|---|
| **0** | 20000 | 20000 | 20000 | 36 months | 10.41 | 649.21 | B | NaN |
| **1** | 11000 | 11000 | 11000 | 36 months | 7.34 | 341.37 | A | 10+ y |
| **2** | 12000 | 12000 | 12000 | 36 months | 6.07 | 365.45 | A | 10+ y |
| **3** | 35000 | 35000 | 35000 | 36 months | 16.01 | 1230.67 | C | 3 yea |
| **4** | 20000 | 20000 | 20000 | 60 months | 9.92 | 424.16 | B | 10+ y |

5 rows × 30 columns

In [342]:
```
# Question 6: Calculating null values in each column

null_counts = dataset.isnull().sum()
print("Number of null values in each column:\n{}".format(null_counts))
```

```
Number of null values in each column:
loan_amnt                  0
funded_amnt                0
funded_amnt_inv            0
term                       0
int_rate                   0
installment                0
grade                      0
emp_length              9270
home_ownership             0
annual_inc                 0
verification_status        0
loan_status                0
purpose                    0
title                      0
addr_state                 0
dti                      260
delinq_2yrs                0
earliest_cr_line           0
inq_last_6mths             0
open_acc                   0
pub_rec                    0
revol_bal                  0
revol_util                16
total_acc                  0
last_credit_pull_d         2
delinq_amnt                0
mort_acc                   0
pub_rec_bankruptcies       0
tax_liens                  0
debt_to_income_ratio       0
dtype: int64
```

In [343]:
```
# Question 6: Deleting "emp_length" column since it has plenty of null values
 as well as dropping and null values from any other rows

dataset = dataset.drop("emp_length",axis=1)
dataset = dataset.dropna()
```

In [344]:
```
# Question 6: List the columns we have left

list(dataset.columns.values)
```

Out[344]:
```
['loan_amnt',
 'funded_amnt',
 'funded_amnt_inv',
 'term',
 'int_rate',
 'installment',
 'grade',
 'home_ownership',
 'annual_inc',
 'verification_status',
 'loan_status',
 'purpose',
 'title',
 'addr_state',
 'dti',
 'delinq_2yrs',
 'earliest_cr_line',
 'inq_last_6mths',
 'open_acc',
 'pub_rec',
 'revol_bal',
 'revol_util',
 'total_acc',
 'last_credit_pull_d',
 'delinq_amnt',
 'mort_acc',
 'pub_rec_bankruptcies',
 'tax_liens',
 'debt_to_income_ratio']
```

In [345]:
```
# Question 6: Investigating categorical columns and printing data types and th
eir values

print("Data types and their frequency\n{}".format(dataset.dtypes.value_counts
()))
```

```
Data types and their frequency
int64      12
object     10
float64     7
dtype: int64
```

In [346]:
```
# Question 6: Getting a snap shot of values in each column with data type obje
ct

object_columns_df = dataset.select_dtypes(include=['object'])

print(object_columns_df.iloc[0])
```

```
term                           36 months
grade                                  B
home_ownership                  MORTGAGE
verification_status             Verified
loan_status                      Current
purpose                debt_consolidation
title                  Debt consolidation
addr_state                             FL
earliest_cr_line                Oct-1987
last_credit_pull_d              Mar-2018
Name: 0, dtype: object
```

In [347]:
```
# Question 6: Dropping more columns

dataset.drop(['last_credit_pull_d','addr_state','title','earliest_cr_line' ],1
, inplace=True)
```

In [348]:
```
# Question 6: Dropping "purpose" column

dataset.drop(['purpose' ],1, inplace=True)
```

In [349]:
```
dataset.drop(['term','grade','home_ownership','verification_status','loan_stat
us' ],1, inplace=True)
```

In [350]:
```
# Question 6: List the columns we have left

list(dataset.columns.values)
```

Out[350]:
```
['loan_amnt',
 'funded_amnt',
 'funded_amnt_inv',
 'int_rate',
 'installment',
 'annual_inc',
 'dti',
 'delinq_2yrs',
 'inq_last_6mths',
 'open_acc',
 'pub_rec',
 'revol_bal',
 'revol_util',
 'total_acc',
 'delinq_amnt',
 'mort_acc',
 'pub_rec_bankruptcies',
 'tax_liens',
 'debt_to_income_ratio']
```

In [351]:
```
# Question 6: Saving the changes to cleaned CSV file

dataset.to_csv("F:/Seagate_Sync/VOL/VOL00/VIVEK/Big Data Analytics Certificati
on/BDA 102/Assignment 7/cleaned_LoanStats.csv", index=False)
```

In [352]:
```
# Question 6:

from  sklearn import ensemble
from sklearn import datasets
from sklearn.utils import shuffle
from sklearn.metrics import mean_squared_error
from matplotlib import pyplot as plt
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble.partial_dependence import plot_partial_dependence
```

In [353]:
```
# Question 6: Assigning int_rate to y

y = dataset.int_rate.values
```

In [354]:
```
# Question 6:

X, y = shuffle(dataset.values, y, random_state=30)

X = X.astype(np.float32)
```

In [355]:
```
# Question 6: Train and Test split

offset = int(X.shape[0] * 0.75)

X_train, y_train = X[:offset], y[:offset]

X_test, y_test = X[offset:], y[offset:]
```

In [356]:
```
# Question 6: Fitting the regressor with 1000 trees, printing mean squared err
or and feature importances

params = {'n_estimators': 1000, 'max_depth': 4, 'min_samples_split': 2,

          'learning_rate': 0.01, 'loss': 'ls'}

clf = ensemble.GradientBoostingRegressor(**params)

clf.fit(X_train, y_train)

mse = mean_squared_error(y_test, clf.predict(X_test))

print("MSE: %.4f" % mse)

clf.feature_importances_
```

```
MSE: 0.0000
```

Out[356]:
```
array([0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0.])
```

In [357]:
```
# Question 6: Printing accuracy score

clf.score(X_test, y_test)
```

Out[357]: 0.9999999895258826

In [358]:
```
# Question 7: Feature ranking and importances

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make_classification

from sklearn.ensemble import GradientBoostingClassifier

importances = clf.feature_importances_

std = np.std([clf.feature_importances_ for tree in clf.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Print the feature ranking

print("Feature ranking:")

for f in range(X.shape[1]):

    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f
]]))
```

```
Feature ranking:
1. feature 3 (1.000000)
2. feature 18 (0.000000)
3. feature 8 (0.000000)
4. feature 1 (0.000000)
5. feature 2 (0.000000)
6. feature 4 (0.000000)
7. feature 5 (0.000000)
8. feature 6 (0.000000)
9. feature 7 (0.000000)
10. feature 9 (0.000000)
11. feature 17 (0.000000)
12. feature 10 (0.000000)
13. feature 11 (0.000000)
14. feature 12 (0.000000)
15. feature 13 (0.000000)
16. feature 14 (0.000000)
17. feature 15 (0.000000)
18. feature 16 (0.000000)
19. feature 0 (0.000000)
```

In [ ]:
```
# Question 7: Plot partial dependence plots

from sklearn.ensemble.partial_dependence import partial_dependence, plot_partial_dependence

params = {'n_estimators': 1000, 'max_depth': 4, 'min_samples_split': 2,

          'learning_rate': 0.01, 'loss': 'ls'}

clf = GradientBoostingRegressor(**params)

clf.fit(X_train, y_train)

features = [0, 1]

# 2, 3, 4, 5, 6, 7]

# 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]

names = ['Loan Amount', 'Interest Rate']

#'funded_amnt_inv', 'int_rate', 'installment', 'annual_inc', 'dti', 'delinq_2yrs']

# ['inq_last_6mths',
#        'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc', 'delinq_amnt', 'mort_acc', 'pub_rec_bankruptcies', 'tax_liens',
#        'debt_to_income_ratio']

fig, axs = plot_partial_dependence(clf, features, X_train, feature_names=names, n_jobs=2, grid_resolution=50) # number of values to plot on x axis

plt.subplots_adjust(bottom=0.1, right=1.1, top=1.4)

print('Custom 3d plot via ``partial_dependence``')

fig = plt.figure()
```