In [1]:

```
import numpy as np
import pandas as pd
```

Import data

In [2]:

```
data = pd.read_csv('XYZCorp_LendingData.txt', sep = '\t', encoding = 'IS(
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\IPython\core\in
teractiveshell.py:3058: DtypeWarning: Columns (17,45,53) h
ave mixed types. Specify dtype option on import or set low
_memory=False.
  interactivity=interactivity, compiler=compiler, result=r
esult)
```

View Data

In [3]:

```
#Transposing as there are many columns and fewer rows to display
data.head().transpose()
```

Out[3]:

	0	1	2	3	4
id	1077501	1077430	1077175	1076863	1075358
member_id	1296599	1314167	1313524	1277178	1311748
loan_amnt	5000	2500	2400	10000	3000
funded_amnt	5000	2500	2400	10000	3000
funded_amnt_inv	4975	2500	2400	10000	3000
...	...	...	...	...	...
total_rev_hi_lim	NaN	NaN	NaN	NaN	NaN
inq_fi	NaN	NaN	NaN	NaN	NaN
total_cu_tl	NaN	NaN	NaN	NaN	NaN
inq_last_12m	NaN	NaN	NaN	NaN	NaN
default_ind	0	1	0	0	0

73 rows × 5 columns

In [4]:

```
data.shape
```

Out[4]:

```
(855969, 73)
```

In [5]:

```
#Display all rows
pd.set_option('display.max_rows', 73)
pd.set_option('display.width', 1000)
```

In [6]:

```
data.head().transpose()
```

Out[6]:

	0	1	2	
id	1077501	1077430	1077175	
member_id	1296599	1314167	1313524	
loan_amnt	5000	2500	2400	
funded_amnt	5000	2500	2400	
funded_amnt_inv	4975	2500	2400	
term	36 months	60 months	36 months	
int_rate	10.65	15.27	15.96	
installment	162.87	59.83	84.33	
grade	B	C	C	
sub_grade	B2	C4	C5	
emp_title	NaN	Ryder	NaN	R
emp_length	10+ years	< 1 year	10+ years	
home_ownership	RENT	RENT	RENT	
annual_inc	24000	30000	12252	
verification_status	Verified	Source Verified	Not Verified	
issue_d	Dec-2011	Dec-2011	Dec-2011	
pymnt_plan	n	n	n	
desc	Borrower added on 12/22/11 > I need to upgra...	Borrower added on 12/22/11 > I plan to use t...	NaN	p
purpose	credit_card	car	small_business	
title	Computer	bike	real estate business	
zip_code	860xx	309xx	606xx	

	0	1	2
addr_state	AZ	GA	IL
dti	27.65	1	8.72
delinq_2yrs	0	0	0
earliest_cr_line	Jan-1985	Apr-1999	Nov-2001
inq_last_6mths	1	5	2
mths_since_last_delinq	NaN	NaN	NaN
mths_since_last_record	NaN	NaN	NaN
open_acc	3	3	2
pub_rec	0	0	0
revol_bal	13648	1687	2956
revol_util	83.7	9.4	98.5
total_acc	9	4	10
initial_list_status	f	f	f
out_prncp	0	0	0
out_prncp_inv	0	0	0
total_pymnt	5861.07	1008.71	3003.65
total_pymnt_inv	5831.78	1008.71	3003.65
total_rec_prncp	5000	456.46	2400
total_rec_int	861.07	435.17	603.65
total_rec_late_fee	0	0	0
recoveries	0	117.08	0
collection_recovery_fee	0	1.11	0
last_pymnt_d	Jan-2015	Apr-2013	Jun-2014
last_pymnt_amnt	171.62	119.66	649.91
next_pymnt_d	NaN	NaN	NaN
last_credit_pull_d	Jan-2016	Sep-2013	Jan-2016
collections_12_mths_ex_med	0	0	0
mths_since_last_major_derog	NaN	NaN	NaN
policy_code	1	1	1
application_type	INDIVIDUAL	INDIVIDUAL	INDIVIDUAL

	0	1	2
annual_inc_joint	NaN	NaN	NaN
dti_joint	NaN	NaN	NaN
verification_status_joint	NaN	NaN	NaN
acc_now_delinq	0	0	0
tot_coll_amt	NaN	NaN	NaN
tot_cur_bal	NaN	NaN	NaN
open_acc_6m	NaN	NaN	NaN
open_il_6m	NaN	NaN	NaN
open_il_12m	NaN	NaN	NaN
open_il_24m	NaN	NaN	NaN
mths_since_rcnt_il	NaN	NaN	NaN
total_bal_il	NaN	NaN	NaN
il_util	NaN	NaN	NaN
open_rv_12m	NaN	NaN	NaN
open_rv_24m	NaN	NaN	NaN
max_bal_bc	NaN	NaN	NaN
all_util	NaN	NaN	NaN
total_rev_hi_lim	NaN	NaN	NaN
inq_fi	NaN	NaN	NaN
total_cu_tl	NaN	NaN	NaN
inq_last_12m	NaN	NaN	NaN
default_ind	0	1	0

In [7]:

```
#Data types of Columns
data.dtypes
```

Out[7]:

```
id                            int64
member_id                     int64
loan_amnt                   float64
funded_amnt                 float64
funded_amnt_inv             float64
term                         object
int_rate                    float64
installment                 float64
grade                        object
sub_grade                    object
emp_title                    object
emp_length                   object
home_ownership               object
annual_inc                  float64
verification_status          object
issue_d                      object
pymnt_plan                   object
desc                         object
purpose                      object
title                        object
zip_code                     object
addr_state                   object
dti                         float64
delinq_2yrs                 float64
earliest_cr_line             object
inq_last_6mths              float64
mths_since_last_delinq      float64
mths_since_last_record      float64
open_acc                    float64
pub_rec                     float64
revol_bal                   float64
revol_util                  float64
total_acc                   float64
initial_list_status          object
out_prncp                   float64
out_prncp_inv               float64
total_pymnt                 float64
total_pymnt_inv             float64
total_rec_prncp             float64
total_rec_int               float64
```

```
total_rec_late_fee            float64
recoveries                    float64
collection_recovery_fee       float64
last_pymnt_d                   object
last_pymnt_amnt               float64
next_pymnt_d                   object
last_credit_pull_d             object
collections_12_mths_ex_med    float64
mths_since_last_major_derog   float64
policy_code                   float64
application_type               object
annual_inc_joint              float64
dti_joint                     float64
verification_status_joint      object
acc_now_delinq                float64
tot_coll_amt                  float64
tot_cur_bal                   float64
open_acc_6m                   float64
open_il_6m                    float64
open_il_12m                   float64
open_il_24m                   float64
mths_since_rcnt_il            float64
total_bal_il                  float64
il_util                       float64
open_rv_12m                   float64
open_rv_24m                   float64
max_bal_bc                    float64
all_util                      float64
total_rev_hi_lim              float64
inq_fi                        float64
total_cu_tl                   float64
inq_last_12m                  float64
default_ind                     int64
dtype: object
```

Understanding Data

In [8]:

```
data['emp_length'].unique()
```

Out[8]:

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

In [9]:

```
data['pymnt_plan'].unique()
```

Out[9]:

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

Checking Null Values

In [10]:

```
data.isna().sum()
```

Out[10]:

```
id                            0
member_id                     0
loan_amnt                     0
funded_amnt                   0
funded_amnt_inv               0
term                          0
int_rate                      0
installment                   0
grade                         0
sub_grade                     0
emp_title                 49443
emp_length                43061
home_ownership                0
annual_inc                    0
verification_status           0
issue_d                       0
pymnt_plan                    0
desc                     734157
purpose                       0
title                        33
zip_code                      0
addr_state                    0
dti                           0
delinq_2yrs                   0
earliest_cr_line              0
inq_last_6mths                0
mths_since_last_delinq   439812
mths_since_last_record   724785
open_acc                      0
pub_rec                       0
revol_bal                     0
revol_util                  446
total_acc                     0
initial_list_status           0
out_prncp                     0
out_prncp_inv                 0
total_pymnt                   0
total_pymnt_inv               0
total_rec_prncp               0
total_rec_int                 0
total_rec_late_fee            0
```

```
recoveries                         0
collection_recovery_fee            0
last_pymnt_d                    8862
last_pymnt_amnt                    0
next_pymnt_d                  252971
last_credit_pull_d                50
collections_12_mths_ex_med        56
mths_since_last_major_derog   642830
policy_code                        0
application_type                   0
annual_inc_joint              855527
dti_joint                     855529
verification_status_joint     855527
acc_now_delinq                     0
tot_coll_amt                   67313
tot_cur_bal                    67313
open_acc_6m                   842681
open_il_6m                    842681
open_il_12m                   842681
open_il_24m                   842681
mths_since_rcnt_il            843035
total_bal_il                  842681
il_util                       844360
open_rv_12m                   842681
open_rv_24m                   842681
max_bal_bc                    842681
all_util                      842681
total_rev_hi_lim               67313
inq_fi                        842681
total_cu_tl                   842681
inq_last_12m                  842681
default_ind                        0
dtype: int64
```

In [11]:

```
#Getting column names
data.columns
```

Out[11]:

```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'fun
ded_amnt_inv', 'term', 'int_rate', 'installment', 'grade',
'sub_grade', 'emp_title', 'emp_length', 'home_ownership',
'annual_inc', 'verification_status', 'issue_d', 'pymnt_pla
n', 'desc', 'purpose', 'title', 'zip_code', 'addr_state',
'dti', 'delinq_2yrs', 'earliest_cr_line', 'inq_last_6mth
s', 'mths_since_last_delinq', 'mths_since_last_record', 'o
pen_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_ac
c', 'initial_list_status', 'out_prncp', 'out_prncp_inv',
'total_pymnt', 'total_pymnt_inv', 'total_rec_prncp', 'tota
l_rec_int', 'total_rec_late_fee', 'recoveries', 'collectio
n_recovery_fee', 'last_pymnt_d', 'last_pymnt_amnt', 'next_
pymnt_d', 'last_credit_pull_d', 'collections_12_mths_ex_me
d', 'mths_since_last_major_derog', 'policy_code', 'applica
tion_type', 'annual_inc_joint', 'dti_joint', 'verification
_status_joint', 'acc_now_delinq', 'tot_coll_amt', 'tot_cur
_bal', 'open_acc_6m', 'open_il_6m', 'open_il_12m', 'open_i
l_24m',
       'mths_since_rcnt_il', 'total_bal_il', 'il_util', 'o
pen_rv_12m', 'open_rv_24m', 'max_bal_bc', 'all_util', 'tot
al_rev_hi_lim', 'inq_fi', 'total_cu_tl', 'inq_last_12m',
'default_ind'],
      dtype='object')
```

Setting Index

In [12]:

```
data['id'].nunique()
```

Out[12]:

```
855969
```

In [13]:

```
data['member_id'].nunique()
```

Out[13]:

855969

Since, member_id indicates that the customer has been given a loan setting it as index and dropping id

In [14]:

```
data.set_index('member_id',inplace = True)
data[:3].head()
```

Out[14]:

member_id	id	loan_amnt	funded_amnt	funded_amnt_inv	term
1296599	1077501	5000.0	5000.0	4975.0	36 months
1314167	1077430	2500.0	2500.0	2500.0	60 months
1313524	1077175	2400.0	2400.0	2400.0	36 months

3 rows × 72 columns

In [15]:

```
data.drop('id', inplace = True, axis = 1)
```

In [16]:

```
#Checking dimensions
data.shape
```

Out[16]:

(855969, 71)

Getting Summary - To fill Null values according to Mean or Median or Mode

In [17]:

```
data.describe().transpose()
```

Out[17]:

	count	mean	std
loan_amnt	855969.0	14745.571335	8425.340005
funded_amnt	855969.0	14732.378305	8419.471653
funded_amnt_inv	855969.0	14700.061226	8425.805478
int_rate	855969.0	13.192320	4.368365
installment	855969.0	436.238072	243.726876
annual_inc	855969.0	75071.185963	64264.469813
dti	855969.0	18.122165	17.423629
delinq_2yrs	855969.0	0.311621	0.857189
inq_last_6mths	855969.0	0.680915	0.964033
mths_since_last_delinq	416157.0	34.149943	21.868500
mths_since_last_record	131184.0	70.463959	27.827120
open_acc	855969.0	11.542447	5.308094
pub_rec	855969.0	0.194537	0.581585
revol_bal	855969.0	16910.526992	22223.741577
revol_util	855523.0	55.019405	23.811585
total_acc	855969.0	25.269269	11.818841
out_prncp	855969.0	8284.830230	8461.946669
out_prncp_inv	855969.0	8281.449347	8458.496422
total_pymnt	855969.0	7653.296336	7909.383591
total_pymnt_inv	855969.0	7622.220520	7885.156400
total_rec_prncp	855969.0	5850.841088	6676.410888
total_rec_int	855969.0	1755.046221	2081.692980
total_rec_late_fee	855969.0	0.319530	3.609399
recoveries	855969.0	47.089499	413.136043
collection_recovery_fee	855969.0	4.951227	62.478569

	count	mean	std
last_pymnt_amnt	855969.0	2225.985295	4864.966593
collections_12_mths_ex_med	855913.0	0.014230	0.133712
mths_since_last_major_derog	213139.0	44.101450	22.164907
policy_code	855969.0	1.000000	0.000000
annual_inc_joint	442.0	107412.163982	47987.608637
dti_joint	440.0	18.318477	7.221855
acc_now_delinq	855969.0	0.004944	0.077333
tot_coll_amt	788656.0	225.412882	10489.445929
tot_cur_bal	788656.0	139766.247529	153938.532621
open_acc_6m	13288.0	1.072998	1.206939
open_il_6m	13288.0	2.945665	3.080330
open_il_12m	13288.0	0.749323	0.986169
open_il_24m	13288.0	1.666767	1.685394
mths_since_rcnt_il	12934.0	20.833153	26.739237
total_bal_il	13288.0	36511.541391	42492.757796
il_util	11609.0	71.486993	23.015293
open_rv_12m	13288.0	1.354305	1.483710
open_rv_24m	13288.0	2.945515	2.595313
max_bal_bc	13288.0	5840.443332	5108.500262
all_util	13288.0	61.024526	20.018117
total_rev_hi_lim	788656.0	32163.574526	37699.637883
inq_fi	13288.0	0.947772	1.441667
total_cu_tl	13288.0	1.524232	2.697601
inq_last_12m	13288.0	1.841963	2.975049
default_ind	855969.0	0.054286	0.226581

In [18]:

```
data['mths_since_last_delinq'].fillna(data['mths_since_last_delinq'].mea
```

In [19]:

```
data['mths_since_last_delinq'].isna().sum()
```

Out[19]:

```
0
```

In [20]:

```
data['mths_since_last_record'].fillna(data['mths_since_last_record'].mea
```

In [21]:

```
data['mths_since_last_record'].isna().sum()
```

Out[21]:

```
0
```

In [22]:

```
data['mths_since_last_major_derog'].fillna(data['mths_since_last_major_de
```

In [23]:

```
data['mths_since_last_major_derog'].isna().sum()
```

Out[23]:

```
0
```

In [24]:

```
#Since dtu_join is Null only where the Loan/Application Type is individu
data['dti_joint'].fillna(0, inplace = True)
```

In [25]:

```
data['dti_joint'].isna().sum()
```

Out[25]:

```
0
```

In [26]:

```
data.shape
```

Out[26]:

```
(855969, 71)
```

In [27]:

```
#Removing all columns (unnecessary one's) where the null values are more
data.drop(data.columns[data.isna().sum() == 842681].tolist(), axis = 1, 
```

In [28]:

```
data.shape
```

Out[28]:

```
(855969, 59)
```

In [29]:

```
data.isna().sum()
```

Out[29]:

```
loan_amnt                       0
funded_amnt                     0
funded_amnt_inv                 0
term                            0
int_rate                        0
installment                     0
grade                           0
sub_grade                       0
emp_title                   49443
emp_length                  43061
home_ownership                  0
annual_inc                      0
verification_status             0
issue_d                         0
pymnt_plan                      0
desc                       734157
purpose                         0
title                          33
zip_code                        0
addr_state                      0
dti                             0
delinq_2yrs                     0
earliest_cr_line                0
inq_last_6mths                  0
mths_since_last_delinq          0
mths_since_last_record          0
open_acc                        0
pub_rec                         0
revol_bal                       0
revol_util                    446
total_acc                       0
initial_list_status             0
out_prncp                       0
out_prncp_inv                   0
total_pymnt                     0
total_pymnt_inv                 0
total_rec_prncp                 0
total_rec_int                   0
total_rec_late_fee              0
recoveries                      0
collection_recovery_fee         0
```

```
last_pymnt_d                    8862
last_pymnt_amnt                    0
next_pymnt_d                  252971
last_credit_pull_d                50
collections_12_mths_ex_med        56
mths_since_last_major_derog        0
policy_code                        0
application_type                   0
annual_inc_joint              855527
dti_joint                          0
verification_status_joint     855527
acc_now_delinq                     0
tot_coll_amt                   67313
tot_cur_bal                    67313
mths_since_rcnt_il            843035
il_util                       844360
total_rev_hi_lim               67313
default_ind                        0
dtype: int64
```

In [30]:

```
data.drop(['mths_since_rcnt_il'], inplace = True, axis = 1)
data.shape
```

Out[30]:

```
(855969, 58)
```

In [31]:

```
#Finding the count in each class
data.groupby('il_util').agg(np.size).transpose()
```

Out[31]:

il_util	0.0	0.5	0.7	1.1	1.2	1.8	1.9	2.1	2.4
loan_amnt	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
funded_amnt	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
funded_amnt_inv	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
term	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
int_rate	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
installment	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
grade	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
sub_grade	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
emp_title	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
emp_length	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
home_ownership	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
annual_inc	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
verification_status	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
issue_d	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pymnt_plan	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
desc	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
purpose	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
title	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
zip_code	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
addr_state	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
dti	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
delinq_2yrs	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
earliest_cr_line	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
inq_last_6mths	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
mths_since_last_delinq	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

il_util	0.0	0.5	0.7	1.1	1.2	1.8	1.9	2.1	2.4
mths_since_last_record	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
open_acc	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
pub_rec	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
revol_bal	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
revol_util	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
total_acc	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
initial_list_status	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
out_prncp	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
out_prncp_inv	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
total_pymnt	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
total_pymnt_inv	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
total_rec_prncp	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
total_rec_int	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
total_rec_late_fee	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
recoveries	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
collection_recovery_fee	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
last_pymnt_d	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
last_pymnt_amnt	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
next_pymnt_d	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
last_credit_pull_d	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
collections_12_mths_ex_med	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
mths_since_last_major_derog	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
policy_code	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
application_type	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
annual_inc_joint	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
dti_joint	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
verification_status_joint	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
acc_now_delinq	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
tot_coll_amt	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
tot_cur_bal	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

il_util	0.0	0.5	0.7	1.1	1.2	1.8	1.9	2.1	2.4
total_rev_hi_lim	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
default_ind	71.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

57 rows × 1195 columns

In [32]:

```
data['il_util'].describe()
```

Out[32]:

```
count    11609.000000
mean        71.486993
std         23.015293
min          0.000000
25%         58.500000
50%         75.000000
75%         87.500000
max        223.300000
Name: il_util, dtype: float64
```

In [33]:

```
#Simply exploring with Median
data['il_util'].fillna(data['il_util'].median(), inplace = True)
```

In [34]:

```
data['il_util'].isna().sum()
```

Out[34]:

```
0
```

Converting Categorical into Factor

In [35]:

```
#default_ind column is treated as numeric, converting it into Factor
data['default_ind'] = pd.factorize(data['default_ind'])[0]
```

In [36]:

```
data.columns[data.isna().any()]
```

Out[36]:

```
Index(['emp_title', 'emp_length', 'desc', 'title', 'revol_
util', 'last_pymnt_d', 'next_pymnt_d', 'last_credit_pull_
d', 'collections_12_mths_ex_med', 'annual_inc_joint', 'ver
ification_status_joint', 'tot_coll_amt', 'tot_cur_bal', 't
otal_rev_hi_lim'], dtype='object')
```

In [37]:

```
data.iloc[0, data.columns.get_loc('next_pymnt_d')]
```

Out[37]:

```
nan
```

In [38]:

```
cols = ['last_pymnt_d', 'next_pymnt_d', 'last_credit_pull_d']

for i in cols:
    data[i] = pd.to_datetime(data[i])
    data[i] = data[i].fillna(data.iloc[0, data.columns.get_loc(i)])
```

In [39]:

```
data.columns[data.isna().any()]
```

Out[39]:

```
Index(['emp_title', 'emp_length', 'desc', 'title', 'revol_
util', 'next_pymnt_d', 'collections_12_mths_ex_med', 'annu
al_inc_joint', 'verification_status_joint', 'tot_coll_am
t', 'tot_cur_bal', 'total_rev_hi_lim'], dtype='object')
```

In [40]:

```
data.loc[data['next_pymnt_d'].first_valid_index()]['next_pymnt_d']
```

Out[40]:

```
Timestamp('2016-02-01 00:00:00')
```

In [41]:

```
data['next_pymnt_d'].fillna(data.loc[data['next_pymnt_d'].first_valid_in
```

In [42]:

```
data.columns[data.isna().any()]
```

Out[42]:

```
Index(['emp_title', 'emp_length', 'desc', 'title', 'revol_
util', 'collections_12_mths_ex_med', 'annual_inc_joint',
'verification_status_joint', 'tot_coll_amt', 'tot_cur_ba
l', 'total_rev_hi_lim'], dtype='object')
```

In [43]:

```
data[data.columns[data.isna().any()]].head()
```

Out[43]:

member_id	emp_title	emp_length	desc	title	revol_util	cc
1296599	NaN	10+ years	Borrower added on 12/22/11 > I need to upgra...	Computer	83.7	
1314167	Ryder	< 1 year	Borrower added on 12/22/11 > I plan to use t...	bike	9.4	
1313524	NaN	10+ years	NaN	real estate business	98.5	
1277178	AIR RESOURCES BOARD	10+ years	Borrower added on 12/21/11 > to pay for prop...	personel	21.0	
1311748	University Medical Group	1 year	Borrower added on 12/21/11 > I plan on combi...	Personal	53.9	

In [44]:

```
data.loc[data['verification_status_joint'].first_valid_index()]['verifica
```

Out[44]:

```
'Not Verified'
```

In [45]:

```
cols = ['revol_util', 'collections_12_mths_ex_med', 'annual_inc_joint',

for i in cols:
    data[i].fillna(data[i].mean(), inplace = True)
```

In [46]:

```
data.columns[data.isna().any()]
```

Out[46]:

```
Index(['emp_title', 'emp_length', 'desc', 'title', 'verifi
cation_status_joint'], dtype='object')
```

In [47]:

```
data['verification_status_joint'].fillna(data['verification_status_joint
```

Write Cleansed Data as CSV

Used by Tableau

In [48]:

```
#data.to_csv('Cleansed Data.csv')
```

Dropping Un-necessary Columns

In [49]:

```
data.drop(['zip_code', 'addr_state', 'desc', 'title', 'earliest_cr_line'
```

In [50]:

```
len(data.columns)
```

Out[50]:

```
49
```

Feature Engineering

In [51]:

```
data['emp_length'].head()
```

Out[51]:

```
member_id
1296599    10+ years
1314167     < 1 year
1313524    10+ years
1277178    10+ years
1311748       1 year
Name: emp_length, dtype: object
```

In [52]:

```
data['emp_length'].dtypes
```

Out[52]:

```
dtype('O')
```

In [53]:

```
# Cleaning Employee Length Info
data['emp_length'].replace(regex = True, inplace = True, to_replace = r'
data['emp_length'].replace(regex = True, inplace = True, to_replace = r'
data['emp_length'].replace(regex = True, inplace = True, to_replace = r'
data['emp_length'].replace(regex = True, inplace = True, to_replace = r'
data['emp_length'].replace(regex = True, inplace = True, to_replace = r'

data['emp_length'] = data['emp_length'].astype(str)
data['emp_length'].head()
```

Out[53]:

```
member_id
1296599    10
1314167     0
1313524    10
1277178    10
1311748     1
Name: emp_length, dtype: object
```

In [54]:

```
#Checking for Missing values
data['emp_length'].isna().sum()
```

Out[54]:

```
0
```

In [55]:

```
data['emp_length'].unique()
```

Out[55]:

```
array(['10', '0', '1', '3', '8', '9', '4', '5', '6', '2',
'7', 'nan'],
      dtype=object)
```

In [56]:

```
data[data['emp_length'] == 'nan']['int_rate'].mean()
```

Out[56]:

```
13.236918557393858
```

In [57]:

```
data[data['emp_length'] == '9']['int_rate'].mean()
```

Out[57]:

```
13.235036160419744
```

In [58]:

```
#Treating 'nan' as Null values
data['emp_length'].replace(regex = True, inplace = True, to_replace = 'na
```

In [59]:

```
data['emp_length'].isna().sum()
```

Out[59]:

```
43061
```

In [60]:

```
data['emp_length'].fillna('9', inplace = True)
```

In [61]:

```
data['emp_length'].unique()
```

Out[61]:

```
array(['10', '0', '1', '3', '8', '9', '4', '5', '6', '2',
'7'],
      dtype=object)
```

In [62]:

```
data['grade'].unique()
```

Out[62]:

```
array(['B', 'C', 'A', 'E', 'F', 'D', 'G'], dtype=object)
```

In [63]:

```
data['home_ownership'].unique()
```

Out[63]:

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

In [64]:

```
data['verification_status'].unique()
```

Out[64]:

```
array(['Verified', 'Source Verified', 'Not Verified'], dty
pe=object)
```

In [65]:

```
data['verification_status_joint'].unique()
```

Out[65]:

```
array(['Not Verified', 'Source Verified', 'Verified'], dty
pe=object)
```

In [66]:

```
data['application_type'].unique()
```

Out[66]:

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

In [67]:

```
data['initial_list_status'].unique()
```

Out[67]:

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

In [68]:

```
cols = ['grade', 'verification_status', 'home_ownership', 'pymnt_plan',

for i in cols:
    data[i] = pd.factorize(data[i])[0]
```

In [69]:

```
data['grade'].unique()
```

Out[69]:

```
array([0, 1, 2, 3, 4, 5, 6], dtype=int64)
```

In [70]:

```
data['home_ownership'].unique()
```

Out[70]:

```
array([0, 1, 2, 3, 4, 5], dtype=int64)
```

In [71]:

```
data['verification_status'].unique()
```

Out[71]:

```
array([0, 1, 2], dtype=int64)
```

In [72]:

```
data['verification_status_joint'].unique()
```

Out[72]:

```
array([0, 1, 2], dtype=int64)
```

In [73]:

```
data['application_type'].unique()
```

Out[73]:

```
array([0, 1], dtype=int64)
```

In [74]:

```
data['initial_list_status'].unique()
```

Out[74]:

```
array([0, 1], dtype=int64)
```

One hot coding

In [75]:

```
from sklearn.preprocessing import LabelBinarizer
```

In [76]:

```
lb = LabelBinarizer()
#One hot coding
x = lb.fit_transform(data['emp_length'].values)
```

In [77]:

```
#Creating Dataframe based on One Hot Coding values and columns
dfOneHot = pd.DataFrame(x, columns = ["Emp_Length_"+str(int(i)) for i in
dfOneHot
```

Out[77]:

	Emp_Length_0	Emp_Length_1	Emp_Length_2	Emp_Length_3
0	0	0	1	0
1	1	0	0	0
2	0	0	1	0
3	0	0	1	0
4	0	1	0	0
...	...	...	...	...
855964	0	0	0	0
855965	0	0	1	0
855966	0	0	0	0
855967	0	1	0	0
855968	0	0	1	0

855969 rows × 11 columns

In [78]:

```
len(data.columns)
```

Out[78]:

49

In [79]:

```
#Pre-preparing for merging based on index
dfOneHot.set_index(data.index, inplace = True)
```

In [80]:

```
#Merged both the data
data = pd.concat([data, dfOneHot], axis = 1)
print(len(data.columns))
data.columns
```

```
60
```

Out[80]:

```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'ter
m', 'int_rate', 'installment', 'grade', 'emp_length', 'hom
e_ownership', 'annual_inc', 'verification_status', 'issue_
d', 'pymnt_plan', 'dti', 'delinq_2yrs', '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', 'tota
l_pymnt', 'total_pymnt_inv', 'total_rec_prncp', 'total_rec
_int', 'total_rec_late_fee', 'recoveries', 'collection_rec
overy_fee', 'last_pymnt_d', 'last_pymnt_amnt', 'next_pymnt
_d', 'last_credit_pull_d', 'collections_12_mths_ex_med',
'mths_since_last_major_derog', 'application_type', 'annual
_inc_joint', 'dti_joint', 'verification_status_joint', 'ac
c_now_delinq', 'tot_coll_amt', 'tot_cur_bal', 'il_util',
'total_rev_hi_lim', 'default_ind', 'Emp_Length_0', 'Emp_Le
ngth_1', 'Emp_Length_2', 'Emp_Length_3', 'Emp_Length_4',
'Emp_Length_5', 'Emp_Length_6', 'Emp_Length_7', 'Emp_Lengt
h_8',
       'Emp_Length_9', 'Emp_Length_10'],
      dtype='object')
```

In [81]:

```
#Dropping redundant column
data.drop('emp_length', axis = 1, inplace = True)
len(data.columns)
```

Out[81]:

```
59
```

In [82]:

```
x = lb.fit_transform(data['term'].values)
```

In [83]:

```
x.shape
```

Out[83]:

```
(855969, 1)
```

In [84]:

```
data['term'] = x
data['term'].head()
```

Out[84]:

```
member_id
1296599    0
1314167    1
1313524    0
1277178    0
1311748    1
Name: term, dtype: int32
```

In [85]:

```
data['term'].unique()
```

Out[85]:

```
array([0, 1], dtype=int64)
```

EDA

In [86]:

```
#Displaying only mean and median to understand data (find outliers)
data.describe().transpose()[['mean', '50%']]
```

Out[86]:

	mean	50%
loan_amnt	14745.571335	13000.000000
funded_amnt	14732.378305	13000.000000
funded_amnt_inv	14700.061226	13000.000000
term	0.298782	0.000000
int_rate	13.192320	12.990000
installment	436.238072	382.550000
grade	1.759513	1.000000
home_ownership	1.101646	2.000000
annual_inc	75071.185963	65000.000000
verification_status	0.973939	1.000000
pymnt_plan	0.000006	0.000000
dti	18.122165	17.610000
delinq_2yrs	0.311621	0.000000
inq_last_6mths	0.680915	0.000000
mths_since_last_delinq	34.149943	34.149943
mths_since_last_record	70.463959	70.463959
open_acc	11.542447	11.000000
pub_rec	0.194537	0.000000
revol_bal	16910.526992	11903.000000
revol_util	55.019405	55.900000
total_acc	25.269269	24.000000
initial_list_status	0.482978	0.000000
out_prncp	8284.830230	6290.250000
out_prncp_inv	8281.449347	6287.650000
total_pymnt	7653.296336	4976.160000

	mean	50%
total_pymnt_inv	7622.220520	4948.250000
total_rec_prncp	5850.841088	3286.890000
total_rec_int	1755.046221	1076.910000
total_rec_late_fee	0.319530	0.000000
recoveries	47.089499	0.000000
collection_recovery_fee	4.951227	0.000000
last_pymnt_amnt	2225.985295	468.820000
collections_12_mths_ex_med	0.014230	0.000000
mths_since_last_major_derog	44.101450	44.101450
application_type	0.000516	0.000000
annual_inc_joint	107412.163982	107412.163982
dti_joint	0.009416	0.000000
verification_status_joint	0.000384	0.000000
acc_now_delinq	0.004944	0.000000
tot_coll_amt	225.412882	0.000000
tot_cur_bal	139766.247529	100561.000000
il_util	74.952355	75.000000
total_rev_hi_lim	32163.574526	25850.000000
default_ind	0.054286	0.000000
Emp_Length_0	0.078971	0.000000
Emp_Length_1	0.064085	0.000000
Emp_Length_2	0.329556	0.000000
Emp_Length_3	0.088772	0.000000
Emp_Length_4	0.078732	0.000000
Emp_Length_5	0.059165	0.000000
Emp_Length_6	0.062867	0.000000
Emp_Length_7	0.048420	0.000000
Emp_Length_8	0.050474	0.000000
Emp_Length_9	0.049559	0.000000
Emp_Length_10	0.089399	0.000000

In [87]:

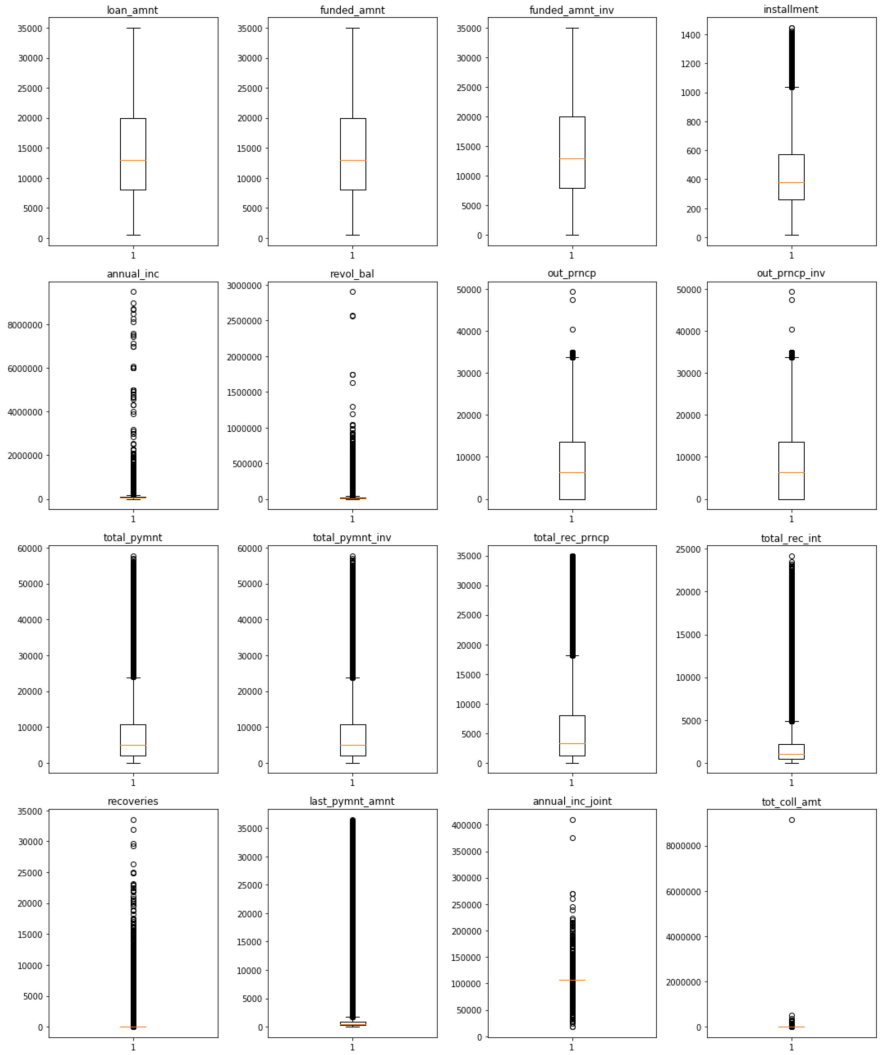
```
b = data.columns[data.isna().any()].tolist()

c = ['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'installment', 'annua

for i in b:
    if i in c:
        c.remove(i)
```

In [88]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [89]:

```
k = 0

fig, axes = plt.subplots(4, 4, figsize = (15, 18))
try:
    for i in range(4):
        for j in range(4):
            axes[i, j].boxplot(data[c[k]])
            axes[i,j].set_title(c[k])
            k += 1
    plt.tight_layout()
except IndexError:
    pass
```



In [90]:

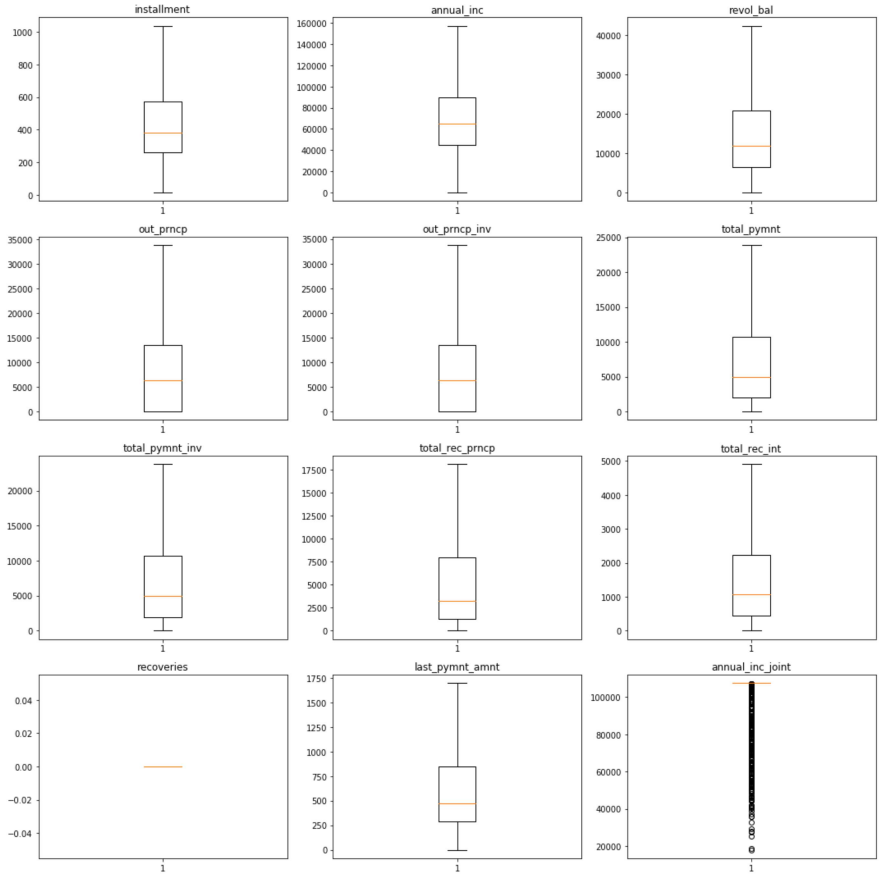
```
for i in ['loan_amnt', 'funded_amnt', 'funded_amnt_inv']:
    c.remove(i)
```

In [91]:

```
for i in c:
    q3 = np.percentile(data[i], [75 ,25])[0]
    q1 = np.percentile(data[i], [75 ,25])[1]
    iqr = q3 - q1
    up_bound = q3 + 1.5 * iqr
    data.loc[data[i] > up_bound, i] = up_bound
```

In [92]:

```
k = 0

fig, axes = plt.subplots(4, 3, figsize = (15, 15))
try:
    for i in range(4):
        for j in range(3):
            axes[i, j].boxplot(data[c[k]])
            axes[i,j].set_title(c[k])
            k += 1
    plt.tight_layout()
except IndexError:
    pass
```



In [93]:

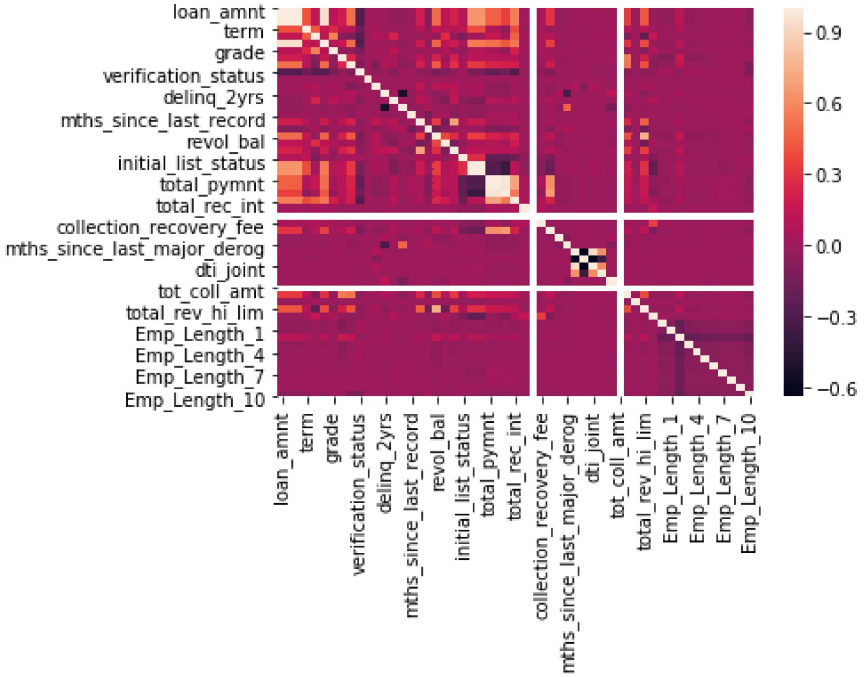
```
import seaborn as sns
```

In [94]:

```
## Write Cleansed Data as CSVplt.figure(figsize = (15, 8))
sns.heatmap(data.corr())
```

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1df6d1147c8>
```



In [95]:

```
data.groupby('default_ind').mean().transpose()
```

Out[95]:

default_ind	0	1
loan_amnt	14755.476206	14573.018486
funded_amnt	14744.071602	14528.669271
funded_amnt_inv	14716.632534	14411.372330
term	0.295341	0.358728
int_rate	13.030057	16.019092
installment	432.817888	437.510303
grade	1.726316	2.337853
home_ownership	1.109298	0.968343
annual_inc	72060.719042	63290.641673
verification_status	0.981453	0.843050
pymnt_plan	0.000006	0.000000
dti	18.103677	18.444234
delinq_2yrs	0.313508	0.278757
inq_last_6mths	0.663729	0.980309
mths_since_last_delinq	34.146719	34.206119
mths_since_last_record	70.406743	71.460714
open_acc	11.570042	11.061721
pub_rec	0.197269	0.146943
revol_bal	15166.546307	14062.145501
revol_util	54.765744	59.438433
total_acc	25.328980	24.229044
initial_list_status	0.494808	0.276885
out_prncp	8740.452369	304.129686
out_prncp_inv	8736.912514	304.040055
total_pymnt	7336.371434	6291.133953

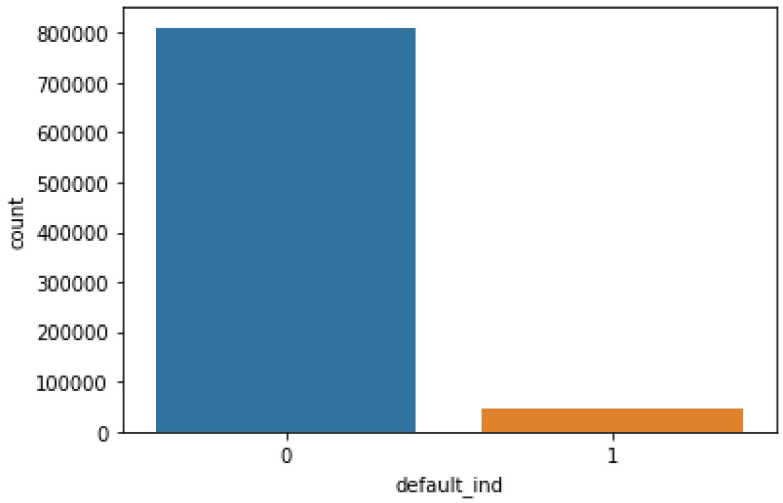
default_ind	0	1
total_pymnt_inv	7305.866474	6231.776690
total_rec_prncp	5539.186453	3311.502213
total_rec_int	1542.675234	1897.775708
total_rec_late_fee	0.197805	2.440093
recoveries	0.000000	0.000000
collection_recovery_fee	0.000000	91.206597
last_pymnt_amnt	684.338990	433.855842
collections_12_mths_ex_med	0.014572	0.008287
mths_since_last_major_derog	44.117552	43.820936
application_type	0.000546	0.000000
annual_inc_joint	107402.494836	107412.163982
dti_joint	0.009957	0.000000
verification_status_joint	0.000406	0.000000
acc_now_delinq	0.005002	0.003938
tot_coll_amt	0.000000	0.000000
tot_cur_bal	134134.435134	117223.854939
il_util	74.949620	75.000000
total_rev_hi_lim	29560.646609	26646.404728
Emp_Length_0	0.078635	0.084834
Emp_Length_1	0.063985	0.065832
Emp_Length_2	0.331787	0.290701
Emp_Length_3	0.088779	0.088644
Emp_Length_4	0.078757	0.078292
Emp_Length_5	0.059051	0.061140
Emp_Length_6	0.062424	0.070588
Emp_Length_7	0.047792	0.059354
Emp_Length_8	0.050069	0.057525
Emp_Length_9	0.049653	0.047926
Emp_Length_10	0.089068	0.095164

In [96]:

```
sns.countplot(x = 'default_ind', data = data)
```

Out[96]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1df6d23d708>
```



Percentage of Data distributed (Also Tableau represented)

In [97]:

```
print('Loan Defaulter %: ', (len(data[data['default_ind'] == 0]) / data.s
```

```
Loan Defaulter %:  94.57141555360066
```

In [98]:

```
print('Loan Non-Defaulter %: ', (len(data[data['default_ind'] == 1]) / da
```

```
Loan Non-Defaulter %:  5.428584446399344
```

NOTE: The above percentages show that the classes are imbalanced

Write Cleansed Data as CSV

In [99]:

```
#data.to_csv('Cleansed Data Final.csv')
```

Seperating Dependent and Independent Variable

In [100]:

```
#Sorting for training and test data-set
data['issue_d'] = pd.to_datetime(data['issue_d'])
data = data.sort_values('issue_d')
```

In [101]:

```
X = data.drop('default_ind', axis = 1)
y = data.loc[:,'default_ind']
```

In [102]:

```
X.shape
```

Out[102]:

```
(855969, 58)
```

In [103]:

```
y.shape
```

Out[103]:

```
(855969,)
```

In [104]:

```
X.head()
```

Out[104]:

member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	i
86999	7500.0	7500.0	800.0	0	13.75	
90962	5000.0	5000.0	4150.0	0	7.43	
98991	5750.0	5750.0	3675.0	0	7.43	
112227	5000.0	5000.0	3975.0	0	7.43	
109346	1200.0	1200.0	0.0	0	11.54	

5 rows × 58 columns

In [105]:

```
y.head()
```

Out[105]:

```
member_id
86999     0
90962     0
98991     0
112227    0
109346    0
Name: default_ind, dtype: int64
```

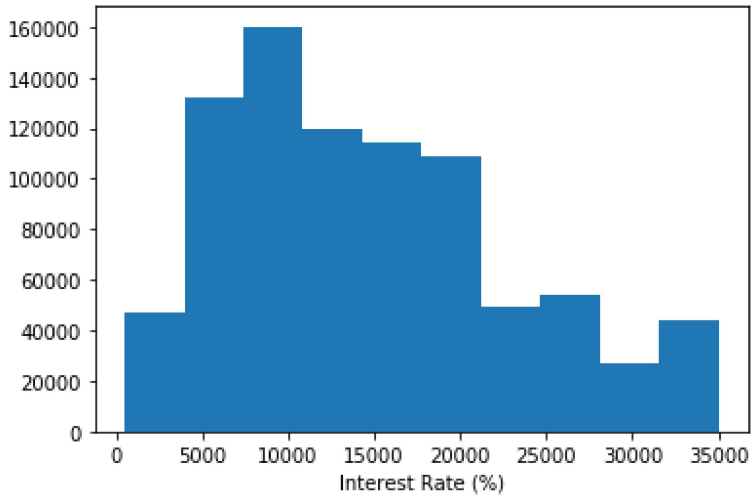
Visualization

In [106]:

```
plt.rcParams['agg.path.chunksize'] = 10000
plt.hist(X['loan_amnt'], histtype = 'bar')
plt.xlabel('Interest Rate (%)')
```

Out[106]:

```
Text(0.5, 0, 'Interest Rate (%)')
```

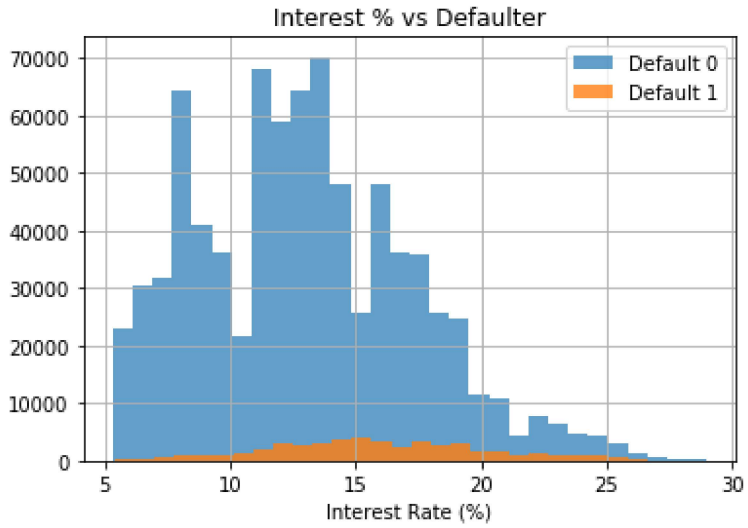


In [107]:

```
X[data['default_ind'] == 0]['int_rate'].hist(alpha = 0.7, label = 'Defau
X[data['default_ind'] == 1]['int_rate'].hist(alpha = 0.8, label = 'Defau
plt.legend()
plt.xlabel('Interest Rate')
plt.title('Interest % vs Defaulter')
plt.xlabel('Interest Rate (%)')
```

Out[107]:

```
Text(0.5, 0, 'Interest Rate (%)')
```

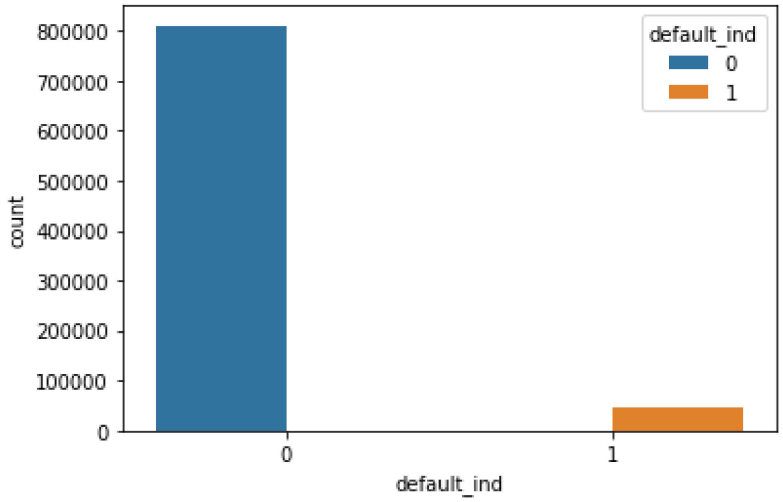


In [108]:

```
sns.countplot('default_ind', hue='default_ind', data=data)
```

Out[108]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1df60febc08>
```



Splitting Data in Training and Test Data-set

In [109]:

```
sum(X['issue_d'] <= 'May-2015')
```

Out[109]:

```
598978
```

In [110]:

```
X.columns
```

Out[110]:

```
Index(['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'ter
m', 'int_rate', 'installment', 'grade', 'home_ownership',
'annual_inc', 'verification_status', 'issue_d', 'pymnt_pla
n', 'dti', 'delinq_2yrs', 'inq_last_6mths', 'mths_since_la
st_delinq', 'mths_since_last_record', 'open_acc', 'pub_re
c', 'revol_bal', 'revol_util', 'total_acc', 'initial_list_
status', 'out_prncp', 'out_prncp_inv', 'total_pymnt', 'tot
al_pymnt_inv', 'total_rec_prncp', 'total_rec_int', 'total_
rec_late_fee', 'recoveries', 'collection_recovery_fee', 'l
ast_pymnt_d', 'last_pymnt_amnt', 'next_pymnt_d', 'last_cre
dit_pull_d', 'collections_12_mths_ex_med', 'mths_since_las
t_major_derog', 'application_type', 'annual_inc_joint', 'd
ti_joint', 'verification_status_joint', 'acc_now_delinq',
'tot_coll_amt', 'tot_cur_bal', 'il_util', 'total_rev_hi_li
m', 'Emp_Length_0', 'Emp_Length_1', 'Emp_Length_2', 'Emp_L
ength_3', 'Emp_Length_4', 'Emp_Length_5', 'Emp_Length_6',
'Emp_Length_7', 'Emp_Length_8', 'Emp_Length_9', 'Emp_Lengt
h_10'], dtype='object')
```

In [111]:

```
X.iloc[598977, 10]
```

Out[111]:

```
Timestamp('2015-05-01 00:00:00')
```

Converting Train Dates columns to Float

In [112]:

```
cols = data.columns[data.dtypes == 'datetime64[ns]']

for i in cols:
    data[i] = (data[i] - data[i].min())/np.timedelta64(1, 'M')
    data[i] = data[i].round()
    data[i] = pd.to_numeric(data[i], downcast = 'integer')
```

In [113]:

```
data.columns[data.dtypes == 'datetime64[ns]']
```

Out[113]:

```
Index([], dtype='object')
```

NOTE: Converting Datetime of columns to numbers of month since the first occured month in the specific column, as the format for all is of the form mm-yyyy

In [114]:

```
X = data.drop(['default_ind', 'issue_d'], axis = 1)
y = data.loc[:, 'default_ind']
```

In [115]:

```
train_X, train_y = X.iloc[:598978, :], y.iloc[:598978,]
```

In [116]:

```
train_X.head()
```

Out[116]:

member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	i
86999	7500.0	7500.0	800.0	0	13.75	
90962	5000.0	5000.0	4150.0	0	7.43	
98991	5750.0	5750.0	3675.0	0	7.43	
112227	5000.0	5000.0	3975.0	0	7.43	
109346	1200.0	1200.0	0.0	0	11.54	

5 rows × 57 columns

In [117]:

```
train_y.head()
```

Out[117]:

```
member_id
86999     0
90962     0
98991     0
112227    0
109346    0
Name: default_ind, dtype: int64
```

In [118]:

```
test_X, test_y = X.iloc[598978:, :], y.iloc[598978:,]
```

In [119]:

```
test_X.head()
```

Out[119]:

member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	i
55457247	35000.0	35000.0	35000.0	0	9.17	
55086031	11500.0	11500.0	11475.0	1	9.17	
55287617	12000.0	12000.0	12000.0	1	15.61	
55396770	35000.0	35000.0	35000.0	0	12.29	
53224173	27000.0	27000.0	27000.0	1	8.18	

5 rows × 57 columns

In [120]:

```
test_y.head()
```

Out[120]:

```
member_id
55457247    0
55086031    0
55287617    0
55396770    0
53224173    0
Name: default_ind, dtype: int64
```

In [121]:

```
data.groupby('default_ind').agg(len)
```

Out[121]:

default_ind	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate
0	809502.0	809502.0	809502.0	809502	809502.(
1	46467.0	46467.0	46467.0	46467	46467.(

2 rows × 58 columns

Multi - collinearity

In [122]:

```
from statsmodels.stats.outliers_influence import variance_inflation_facto
```

In [123]:

```
def mul_col_vif(train_X):
    vif = pd.DataFrame()
    vif['Features'] = train_X.columns
    vif['VIF'] = [variance_inflation_factor(train_X.values, i) for i in r
    print(vif.isnull().sum())
    vif.dropna(inplace = True)
    print()
    print(vif.isnull().sum())
    return vif
```

In [124]:

```
train_X = train_X.drop('funded_amnt_inv', axis = 1)
u = mul_col_vif(train_X)
u
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\statsmodels\reg
ression\linear_model.py:1636: RuntimeWarning: invalid valu
e encountered in double_scalars
  return 1 - self.ssr/self.centered_tss
C:\Users\sragh\Anaconda3\lib\site-packages\statsmodels\reg
ression\linear_model.py:1636: RuntimeWarning: divide by ze
ro encountered in double_scalars
  return 1 - self.ssr/self.centered_tss
C:\Users\sragh\Anaconda3\lib\site-packages\statsmodels\sta
ts\outliers_influence.py:185: RuntimeWarning: divide by ze
ro encountered in double_scalars
  vif = 1. / (1. - r_squared_i)


Features    0
VIF         5
dtype: int64

Features    0
VIF         0
dtype: int64
```

Out[124]:

	Features	VIF
0	loan_amnt	4.779445e+02
1	funded_amnt	5.380959e+02
2	term	7.129421e+00
3	int_rate	3.505830e+00
4	installment	5.440023e+01
5	grade	1.506469e+00
6	home_ownership	1.506408e+00
7	annual_inc	2.118139e+00
8	verification_status	1.270065e+00

	Features	VIF
9	pymnt_plan	1.000078e+00
10	dti	1.480688e+00
11	delinq_2yrs	1.479348e+00
12	inq_last_6mths	1.159649e+00
13	mths_since_last_delinq	1.622743e+00
14	mths_since_last_record	1.040418e+00
15	open_acc	2.297002e+00
16	pub_rec	1.086682e+00
17	revol_bal	5.166118e+00
18	revol_util	2.372973e+00
19	total_acc	2.210554e+00
20	initial_list_status	1.180528e+00
21	out_prncp	3.304797e+05
22	out_prncp_inv	3.304756e+05
23	total_pymnt	2.483388e+02
24	total_pymnt_inv	1.870057e+02
25	total_rec_prncp	4.624615e+01
26	total_rec_int	6.254558e+00
27	total_rec_late_fee	1.023570e+00
29	collection_recovery_fee	1.280009e+00
30	last_pymnt_d	2.861023e+00
31	last_pymnt_amnt	2.254546e+00
32	next_pymnt_d	1.046260e+00
33	last_credit_pull_d	2.096900e+00
34	collections_12_mths_ex_med	1.013276e+00
35	mths_since_last_major_derog	1.317092e+00
37	annual_inc_joint	1.381050e-04
40	acc_now_delinq	1.029685e+00
42	tot_cur_bal	1.957492e+00
43	il_util	0.000000e+00

	Features	VIF
44	total_rev_hi_lim	4.567487e+00
45	Emp_Length_0	inf
46	Emp_Length_1	inf
47	Emp_Length_2	inf
48	Emp_Length_3	inf
49	Emp_Length_4	inf
50	Emp_Length_5	inf
51	Emp_Length_6	inf
52	Emp_Length_7	inf
53	Emp_Length_8	inf
54	Emp_Length_9	inf
55	Emp_Length_10	inf

In [125]:

```
u = u.loc[:44,]
u
```

Out[125]:

	Features	VIF
0	loan_amnt	477.944453
1	funded_amnt	538.095857
2	term	7.129421
3	int_rate	3.505830
4	installment	54.400232
5	grade	1.506469
6	home_ownership	1.506408
7	annual_inc	2.118139
8	verification_status	1.270065
9	pymnt_plan	1.000078
10	dti	1.480688
11	delinq_2yrs	1.479348
12	inq_last_6mths	1.159649
13	mths_since_last_delinq	1.622743
14	mths_since_last_record	1.040418
15	open_acc	2.297002
16	pub_rec	1.086682
17	revol_bal	5.166118
18	revol_util	2.372973
19	total_acc	2.210554
20	initial_list_status	1.180528
21	out_prncp	330479.718175
22	out_prncp_inv	330475.628644
23	total_pymnt	248.338848
24	total_pymnt_inv	187.005702

	Features	VIF
25	total_rec_prncp	46.246151
26	total_rec_int	6.254558
27	total_rec_late_fee	1.023570
29	collection_recovery_fee	1.280009
30	last_pymnt_d	2.861023
31	last_pymnt_amnt	2.254546
32	next_pymnt_d	1.046260
33	last_credit_pull_d	2.096900
34	collections_12_mths_ex_med	1.013276
35	mths_since_last_major_derog	1.317092
37	annual_inc_joint	0.000138
40	acc_now_delinq	1.029685
42	tot_cur_bal	1.957492
43	il_util	0.000000
44	total_rev_hi_lim	4.567487

In [126]:

```
train_X = train_X.drop(['total_pymnt_inv', 'out_prncp', 'installment', '
u = mul_col_vif(train_X)
u
```

```
Features    0
VIF         5
dtype: int64

Features    0
VIF         0
dtype: int64
```

Out[126]:

	Features	VIF
0	loan_amnt	7.084397
1	term	1.831624
2	int_rate	2.988666
3	grade	1.497796
4	home_ownership	1.505989
5	annual_inc	2.117849
6	verification_status	1.268947
7	pymnt_plan	1.000072
8	dti	1.479461
9	delinq_2yrs	1.479034
10	inq_last_6mths	1.158992
11	mths_since_last_delinq	1.621547
12	mths_since_last_record	1.030663
13	open_acc	2.292600
14	pub_rec	1.086114
15	revol_bal	5.148203
16	revol_util	2.365650
17	total_acc	2.209847
18	initial_list_status	1.166323

	Features	VIF
19	out_prncp_inv	4.888058
20	total_pymnt	58.739114
21	total_rec_prncp	45.702836
22	total_rec_int	6.184007
23	total_rec_late_fee	1.023054
25	collection_recovery_fee	1.275879
26	last_pymnt_d	2.717365
27	last_pymnt_amnt	2.213324
28	next_pymnt_d	1.045355
29	last_credit_pull_d	2.093049
30	collections_12_mths_ex_med	1.013149
31	mths_since_last_major_derog	1.316890
33	annual_inc_joint	0.000068
36	acc_now_delinq	1.029605
38	tot_cur_bal	1.951749
39	il_util	0.000000
40	total_rev_hi_lim	4.547077
41	Emp_Length_0	inf
42	Emp_Length_1	inf
43	Emp_Length_2	inf
44	Emp_Length_3	inf
45	Emp_Length_4	inf
46	Emp_Length_5	inf
47	Emp_Length_6	inf
48	Emp_Length_7	inf
49	Emp_Length_8	inf
50	Emp_Length_9	inf
51	Emp_Length_10	inf

In [127]:

```
u = u.loc[:40,]
u
```

Out[127]:

	Features	VIF
0	loan_amnt	7.084397
1	term	1.831624
2	int_rate	2.988666
3	grade	1.497796
4	home_ownership	1.505989
5	annual_inc	2.117849
6	verification_status	1.268947
7	pymnt_plan	1.000072
8	dti	1.479461
9	delinq_2yrs	1.479034
10	inq_last_6mths	1.158992
11	mths_since_last_delinq	1.621547
12	mths_since_last_record	1.030663
13	open_acc	2.292600
14	pub_rec	1.086114
15	revol_bal	5.148203
16	revol_util	2.365650
17	total_acc	2.209847
18	initial_list_status	1.166323
19	out_prncp_inv	4.888058
20	total_pymnt	58.739114
21	total_rec_prncp	45.702836
22	total_rec_int	6.184007
23	total_rec_late_fee	1.023054

	Features	VIF
25	collection_recovery_fee	1.275879
26	last_pymnt_d	2.717365
27	last_pymnt_amnt	2.213324
28	next_pymnt_d	1.045355
29	last_credit_pull_d	2.093049
30	collections_12_mths_ex_med	1.013149
31	mths_since_last_major_derog	1.316890
33	annual_inc_joint	0.000068
36	acc_now_delinq	1.029605
38	tot_cur_bal	1.951749
39	il_util	0.000000
40	total_rev_hi_lim	4.547077

In [128]:

```
train_X = train_X.drop(['total_pymnt', 'revol_bal'], axis = 1)
u = mul_col_vif(train_X)
u
```

```
Features    0
VIF         5
dtype: int64

Features    0
VIF         0
dtype: int64
```

Out[128]:

	Features	VIF
0	loan_amnt	6.423957
1	term	1.806919
2	int_rate	2.917264
3	grade	1.494207
4	home_ownership	1.499237
5	annual_inc	2.063978
6	verification_status	1.268118
7	pymnt_plan	1.000070
8	dti	1.447968
9	delinq_2yrs	1.477104
10	inq_last_6mths	1.158121
11	mths_since_last_delinq	1.621370
12	mths_since_last_record	1.030060
13	open_acc	2.290219
14	pub_rec	1.085669
15	revol_util	1.355213
16	total_acc	2.209467
17	initial_list_status	1.163435
18	out_prncp_inv	4.760367

	Features	VIF
19	total_rec_prncp	5.533065
20	total_rec_int	4.003920
21	total_rec_late_fee	1.022713
23	collection_recovery_fee	1.154522
24	last_pymnt_d	2.706887
25	last_pymnt_amnt	2.196076
26	next_pymnt_d	1.045310
27	last_credit_pull_d	2.088070
28	collections_12_mths_ex_med	1.013134
29	mths_since_last_major_derog	1.316833
31	annual_inc_joint	0.000218
34	acc_now_delinq	1.029533
36	tot_cur_bal	1.948813
37	il_util	0.000000
38	total_rev_hi_lim	1.823569
39	Emp_Length_0	inf
40	Emp_Length_1	inf
41	Emp_Length_2	inf
42	Emp_Length_3	inf
43	Emp_Length_4	inf
44	Emp_Length_5	inf
45	Emp_Length_6	inf
46	Emp_Length_7	inf
47	Emp_Length_8	inf
48	Emp_Length_9	inf
49	Emp_Length_10	inf

In [129]:

```
u = u.loc[:38,]
u
```

Out[129]:

	Features	VIF
0	loan_amnt	6.423957
1	term	1.806919
2	int_rate	2.917264
3	grade	1.494207
4	home_ownership	1.499237
5	annual_inc	2.063978
6	verification_status	1.268118
7	pymnt_plan	1.000070
8	dti	1.447968
9	delinq_2yrs	1.477104
10	inq_last_6mths	1.158121
11	mths_since_last_delinq	1.621370
12	mths_since_last_record	1.030060
13	open_acc	2.290219
14	pub_rec	1.085669
15	revol_util	1.355213
16	total_acc	2.209467
17	initial_list_status	1.163435
18	out_prncp_inv	4.760367
19	total_rec_prncp	5.533065
20	total_rec_int	4.003920
21	total_rec_late_fee	1.022713
23	collection_recovery_fee	1.154522
24	last_pymnt_d	2.706887
25	last_pymnt_amnt	2.196076

	Features	VIF
26	next_pymnt_d	1.045310
27	last_credit_pull_d	2.088070
28	collections_12_mths_ex_med	1.013134
29	mths_since_last_major_derog	1.316833
31	annual_inc_joint	0.000218
34	acc_now_delinq	1.029533
36	tot_cur_bal	1.948813
37	il_util	0.000000
38	total_rev_hi_lim	1.823569

In [130]:

```
#Removing the columns from test data also
test_X = test_X.drop(['funded_amnt_inv', 'total_pymnt_inv', 'out_prncp',
```

Standard Scaling

In [131]:

```
#Works on Numeric Data only
cols_ind = [0, 1, 2, 8, 12, 19, 23, 24, 25, 26, 27, 28, 29, 31, 33, 39,
```

In [132]:

```
from sklearn.preprocessing import StandardScaler
```

In [133]:

```
sc = StandardScaler()
```

In [134]:

```
data[data.columns[cols_ind]] = sc.fit_transform(data[data.columns[cols_i
```

In [135]:

```
data[data.columns[cols_ind]].head()
```

Out[135]:

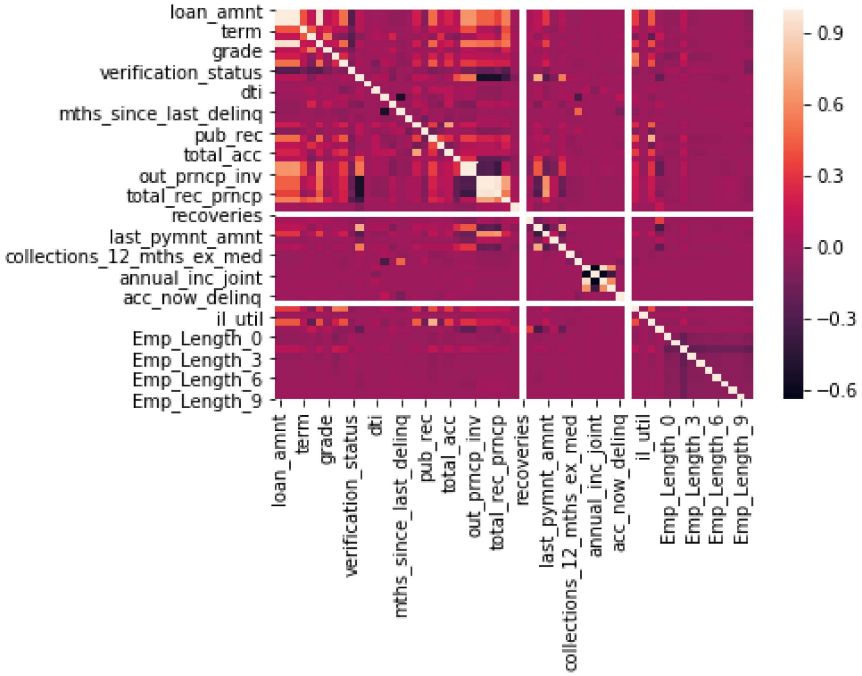
member_id	loan_amnt	funded_amnt	funded_amnt_inv	annual_inc	
86999	-0.859974	-0.859007	-1.649702	-1.431087	-0.21
90962	-1.156698	-1.155938	-1.252114	2.263186	-1.04
98991	-1.067681	-1.066858	-1.308488	1.541648	-1.02
112227	-1.156698	-1.155938	-1.272883	-0.911580	-0.89
109346	-1.607719	-1.607273	-1.744649	-1.488810	-0.92

In [136]:

```
## Write Cleansed Data as CSVplt.figure(figsize = (15, 8))
sns.heatmap(data.corr())
```

Out[136]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1df3784e048>
```



In [137]:

```
data.corr()
```

Out[137]:

	loan_amnt	funded_amnt	funded_amnt_
loan_amnt	1.000000	0.999265	0.9976
funded_amnt	0.999265	1.000000	0.9984
funded_amnt_inv	0.997601	0.998485	1.0000
term	0.411192	0.410209	0.4105
int_rate	0.142966	0.143116	0.1433
installment	0.949648	0.950678	0.9488
grade	0.113410	0.113336	0.1129
home_ownership	0.196468	0.196327	0.1963
annual_inc	0.507935	0.507690	0.5068
verification_status	-0.281637	-0.281071	-0.2815
issue_d	0.097422	0.102059	0.1146
pymnt_plan	0.000761	0.000766	0.0007
dti	0.020189	0.020583	0.0214
delinq_2yrs	-0.000739	-0.000424	0.0000
inq_last_6mths	-0.028928	-0.029279	-0.0304
mths_since_last_delinq	-0.027138	-0.027184	-0.0262
mths_since_last_record	-0.004809	-0.004913	-0.0007
open_acc	0.198926	0.199432	0.2000
pub_rec	-0.082199	-0.081858	-0.0810
revol_bal	0.496878	0.496763	0.4959
revol_util	0.119591	0.120104	0.1208
total_acc	0.222084	0.222039	0.2222
initial_list_status	0.085440	0.087014	0.0904
out_prncp	0.639669	0.641592	0.6439
out_prncp_inv	0.639672	0.641597	0.6439

	loan_amnt	funded_amnt	funded_amnt_
total_pymnt	0.456422	0.454924	0.451
total_pymnt_inv	0.457149	0.455826	0.455
total_rec_prncp	0.352026	0.350479	0.346
total_rec_int	0.564012	0.563010	0.560
total_rec_late_fee	0.025966	0.025579	0.023
recoveries	NaN	NaN	N
collection_recovery_fee	0.055135	0.054898	0.053
last_pymnt_d	0.103031	0.107143	0.122
last_pymnt_amnt	0.306566	0.305895	0.304
next_pymnt_d	-0.020473	-0.020782	-0.021
last_credit_pull_d	0.068954	0.072247	0.084
collections_12_mths_ex_med	-0.017041	-0.016886	-0.016
mths_since_last_major_derog	-0.018077	-0.018086	-0.018
application_type	0.011044	0.011087	0.011
annual_inc_joint	0.001913	0.001892	0.001
dti_joint	0.010362	0.010402	0.010
verification_status_joint	0.010219	0.010248	0.010
acc_now_delinq	0.003572	0.003675	0.003
tot_coll_amt	NaN	NaN	N
tot_cur_bal	0.326983	0.327124	0.326
il_util	-0.014582	-0.014620	-0.014
total_rev_hi_lim	0.429491	0.429559	0.428
default_ind	-0.004907	-0.005797	-0.008
Emp_Length_0	-0.032658	-0.032775	-0.033
Emp_Length_1	-0.031032	-0.031078	-0.031
Emp_Length_2	0.111586	0.111763	0.112
Emp_Length_3	-0.027464	-0.027499	-0.027
Emp_Length_4	-0.021470	-0.021628	-0.021
Emp_Length_5	-0.015009	-0.015110	-0.015
Emp_Length_6	-0.013041	-0.013131	-0.013

	loan_amnt	funded_amnt	funded_amnt_
Emp_Length_7	-0.005879	-0.005990	-0.0059
Emp_Length_8	0.000239	0.000322	0.0005
Emp_Length_9	0.005548	0.005599	0.0057
Emp_Length_10	-0.055159	-0.054977	-0.0543

59 rows × 59 columns

In [138]:

```
data[['Emp_Length_0', 'default_ind']].corr()
```

Out[138]:

	Emp_Length_0	default_ind
Emp_Length_0	1.000000	0.005209
default_ind	0.005209	1.000000

Multi - collinearity

Model Building

Logistic Regression

In [139]:

```
from sklearn.linear_model import LogisticRegression
```

In [140]:

```
log_reg_mod = LogisticRegression()
```

In [141]:

```
log_reg_mod.fit(train_X, train_y)
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\sklearn\linear_
model\logistic.py:432: FutureWarning: Default solver will
be changed to 'lbfgs' in 0.22. Specify a solver to silence
this warning.
  FutureWarning)
```

Out[141]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, f
it_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max
_iter=100,
                   multi_class='warn', n_jobs=None, penalt
y='l2',
                   random_state=None, solver='warn', tol=
0.0001, verbose=0,
                   warm_start=False)
```

In [142]:

```
pred_y = log_reg_mod.predict(test_X)
```

In [143]:

```
from sklearn.metrics import classification_report
```

In [144]:

```
print('Classification Report')
print(classification_report(test_y, pred_y))
```

```
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    256680
           1       0.78      0.79      0.79       311

    accuracy                           1.00    256991
   macro avg       0.89      0.90      0.89    256991
weighted avg       1.00      1.00      1.00    256991
```

In [145]:

```
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

In [146]:

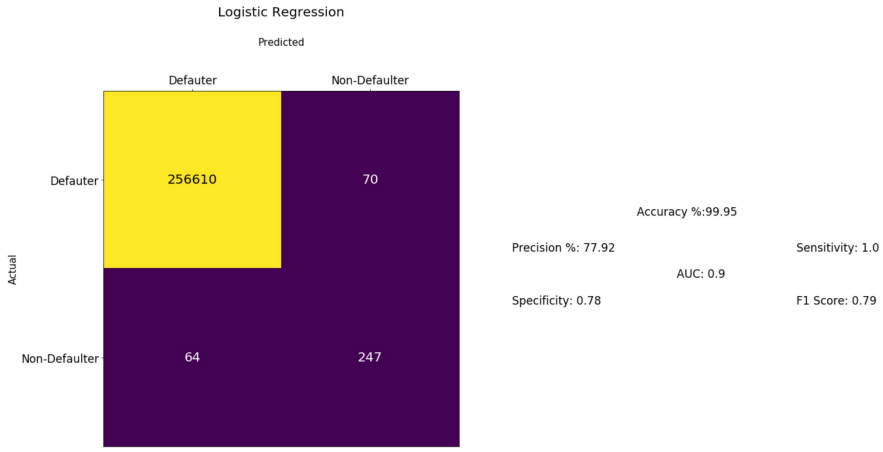
```
def plot_ConfusionMatrix_metrics(conf_mat, test_y, pred_y, figsize = Non

    if figsize is None:
        figsize = (len(conf_mat)*5, len(conf_mat)*5)

    fig, ax = plt.subplots(figsize=figsize)

    matshow = ax.matshow(conf_mat)

    for i in range(conf_mat.shape[0]):
        for j in range(conf_mat.shape[1]):
            cell_text = ''
            cell_text += format(conf_mat[i, j], '.0f')
            ax.text(x=j,
                        y=i,
                        s=cell_text,
                        va='center',
                        ha='center',
                        fontsize = 20,
                        color="white" if [i, j] != [0, 0]
                        else "black")

    if hide_spines:
        ax.spines['right'].set_visible(False)
        ax.spines['top'].set_visible(False)
        ax.spines['left'].set_visible(False)
        ax.spines['bottom'].set_visible(False)

    ax.xaxis.set_ticks_position('top')

    ax.set_xticklabels(class_names, fontsize = 17)
    ax.set_yticklabels(class_names, fontsize = 17)
    ax.xaxis.set_label_coords(0.5, 1.15)

    plt.title(title, fontsize = 20, y = 1.2)
    plt.xlabel('Predicted', fontsize = 15)
    plt.ylabel('Actual', fontsize = 15)

    ax.text(2.5, 0.2, 'Accuracy %:' + str(round(accuracy_score(test_y, pr
    ax.text(1.8, 0.4, 'Precision %: ' + str(round(precision_score(test_y
    ax.text(3.4, 0.4, 'Sensitivity: ' + str(round(conf_mat[0][0] / (conf
    ax.text(1.8, 0.7, 'Specificity: ' + str(round(conf_mat[1][1] / (conf
    ax.text(3.4, 0.7, 'F1 Score: ' + str(round(f1_score(test_y, pred_y),
    ax.text(2.725, 0.55, 'AUC: ' + str(round(roc_auc_score(test_y, pred_y
```

```
    plt.tight_layout()
```

In [147]:

```
plot_ConfusionMatrix_metrics(confusion_matrix(test_y, pred_y), test_y, pr
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launc
her.py:46: UserWarning: Tight layout not applied. The left
and right margins cannot be made large enough to accommoda
te all axes decorations.
```



In [148]:

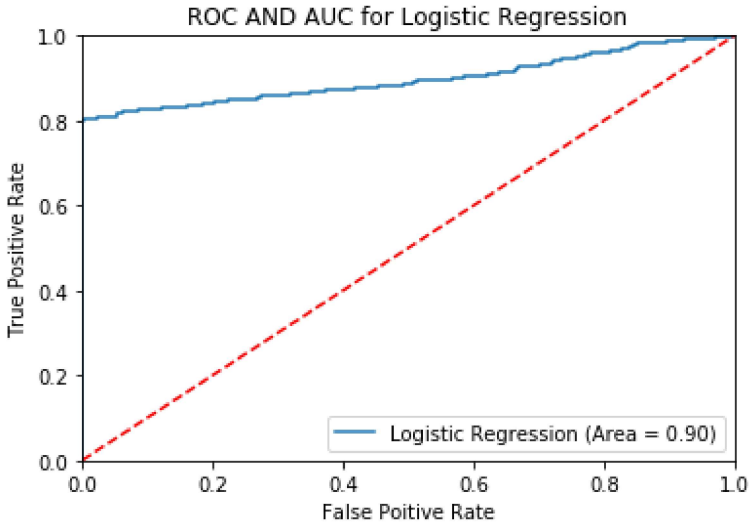
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

In [149]:

```
def roc_auc_plot(test_y, pred_y, title):
    roc_auc = roc_auc_score(test_y, pred_y)
    fpr, tpr, thres = roc_curve(test_y, log_reg_mod.predict_proba(test_X

    #AUC AND FPR.TPR AND THRESHOLDS
    plt.figure()
    plt.plot(fpr,tpr,label='%s (Area = %0.2f)' % (title, roc_auc))
    plt.plot([0,1],[0,1],'r--')
    plt.xlim([0.0,1.0])
    plt.ylim([0.0,1.0])
    plt.xlabel('False Poitive Rate')
    plt.ylabel('True Positive Rate')
    plt.title(f'ROC AND AUC for {title}')
    plt.legend(loc="lower right")
```

In [150]:

```
roc_auc_plot(test_y, pred_y, 'Logistic Regression')
```



In [151]:

```
print(f'\033[1m Majority Class Precision:\033[0m {classification_report(t
print(f'\033[1m Minority Class Recall:\033[0m {classification_report(tes
```

Majority Class Precision: 1.00
Minority Class Recall: 0.79

Decision Tree

In [152]:

```
from sklearn.tree import DecisionTreeClassifier
```

Via GINI Index

Lower the better (less impure)

In [153]:

```
dec_tre_gini = DecisionTreeClassifier(criterion = 'gini')
```

In [154]:

```
dec_tre_gini.fit(train_X, train_y)
```

Out[154]:

```
DecisionTreeClassifier(class_weight=None, criterion='gin
i', max_depth=None,
                       max_features=None, max_leaf_nodes=N
one,
                       min_impurity_decrease=0.0, min_impu
rity_split=None,
                       min_samples_leaf=1, min_samples_spl
it=2,
                       min_weight_fraction_leaf=0.0, preso
rt=False,
                       random_state=None, splitter='best')
```

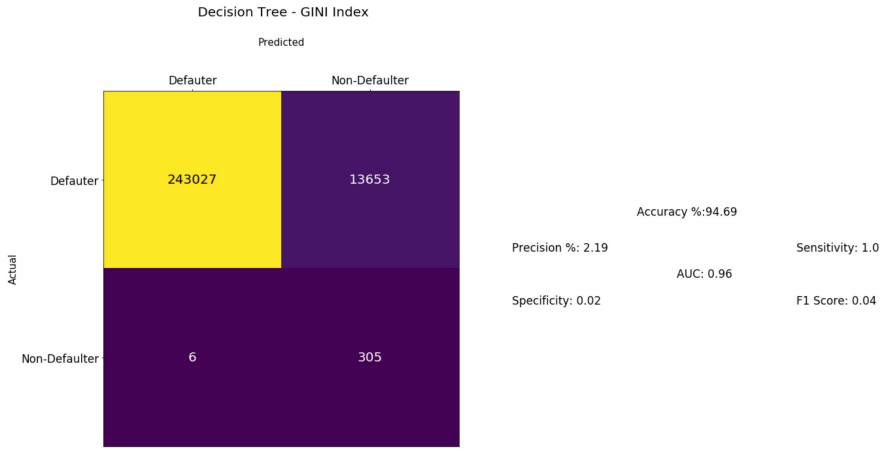
In [155]:

```
pred_y = dec_tre_gini.predict(test_X)
```

In [156]:

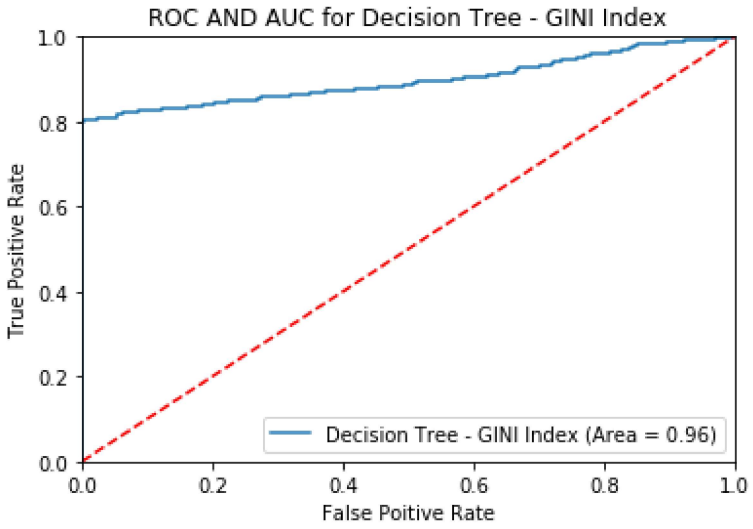
```
plot_ConfusionMatrix_metrics(confusion_matrix(test_y, pred_y), test_y, pr
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launc
her.py:46: UserWarning: Tight layout not applied. The left
and right margins cannot be made large enough to accommoda
te all axes decorations.
```



In [157]:

```
roc_auc_plot(test_y, pred_y, 'Decision Tree - GINI Index')
```



In [158]:

```
print(f'\033[1m Majority Class Precision:\033[0m {classification_report(t
print(f'\033[1m Minority Class Recall:\033[0m {classification_report(tes
```

Majority Class Precision: 1.00
Minority Class Recall: 0.98

Via Entropy

In [159]:

```
dec_tre_ent = DecisionTreeClassifier(criterion = 'entropy')
```

In [160]:

```
dec_tre_ent.fit(train_X, train_y)
```

Out[160]:

```
DecisionTreeClassifier(class_weight=None, criterion='entro
py', max_depth=None,
                       max_features=None, max_leaf_nodes=N
one,
                       min_impurity_decrease=0.0, min_impu
rity_split=None,
                       min_samples_leaf=1, min_samples_spl
it=2,
                       min_weight_fraction_leaf=0.0, preso
rt=False,
                       random_state=None, splitter='best')
```

In [161]:

```
pred_y = dec_tre_ent.predict(test_X)
```

In [162]:

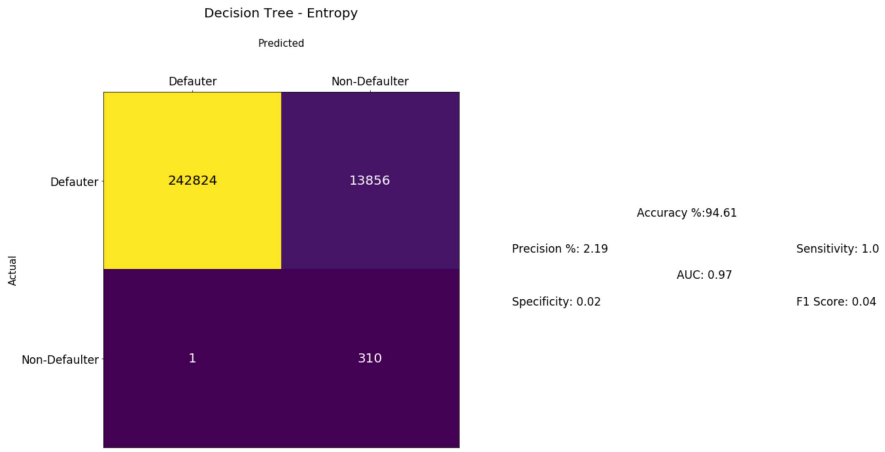
```
print('Classification Report')
print(classification_report(test_y, pred_y))
```

```
Classification Report
              precision    recall  f1-score   support

           0       1.00      0.95      0.97    256680
           1       0.02      1.00      0.04       311

    accuracy                           0.95    256991
   macro avg       0.51      0.97      0.51    256991
weighted avg       1.00      0.95      0.97    256991
```

In [163]:

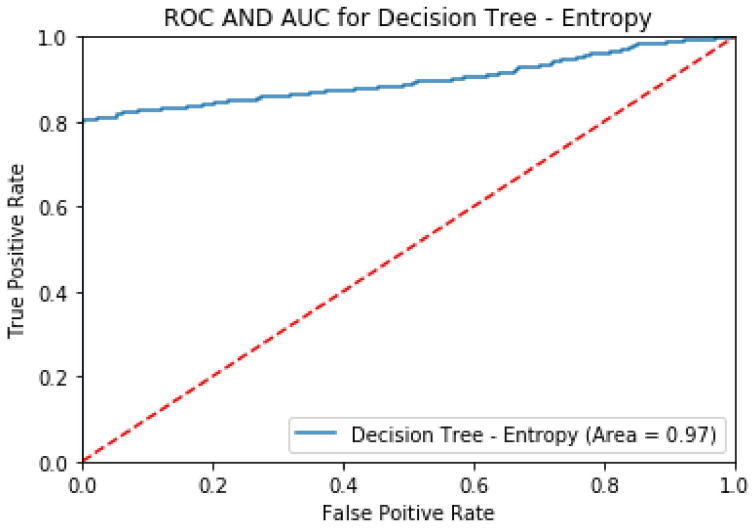
```
plot_ConfusionMatrix_metrics(confusion_matrix(test_y, pred_y), test_y, pr
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launc
her.py:46: UserWarning: Tight layout not applied. The left
and right margins cannot be made large enough to accommoda
te all axes decorations.
```



In [164]:

```
roc_auc_plot(test_y, pred_y, 'Decision Tree - Entropy')
```



In [165]:

```
print(f'\033[1m Majority Class Precision:\033[0m {classification_report(
print(f'\033[1m Minority Class Recall:\033[0m {classification_report(tes
```

Majority Class Precision: 1.00
Minority Class Recall: 1.00

Bagging - Boot-strap AGGregation

In [166]:

```
from sklearn.ensemble import BaggingClassifier
```

In [167]:

```
bag_mod = BaggingClassifier()
```

In [168]:

```
bag_mod.fit(train_X, train_y)
```

Out[168]:

```
BaggingClassifier(base_estimator=None, bootstrap=True, boo
tstrap_features=False,
                  max_features=1.0, max_samples=1.0, n_est
imators=10,
                  n_jobs=None, oob_score=False, random_sta
te=None, verbose=0,
                  warm_start=False)
```

In [169]:

```
pred_y = bag_mod.predict(test_X)
```

In [170]:

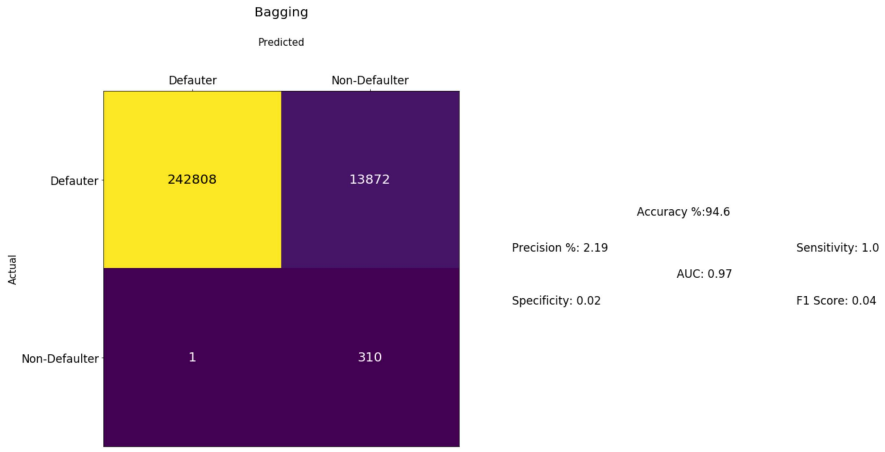
```
print('Classification Report')
print(classification_report(test_y, pred_y))
```

```
Classification Report
              precision    recall  f1-score   support

           0       1.00      0.95      0.97    256680
           1       0.02      1.00      0.04       311

    accuracy                           0.95    256991
   macro avg       0.51      0.97      0.51    256991
weighted avg       1.00      0.95      0.97    256991
```

In [171]:

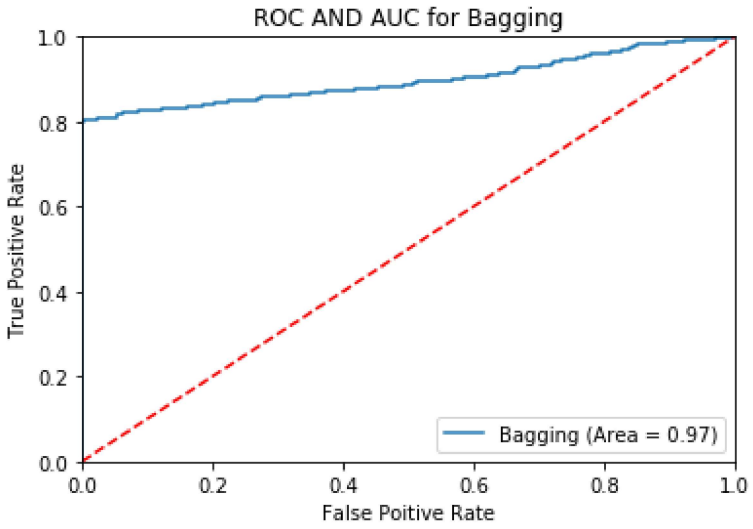
```
plot_ConfusionMatrix_metrics(confusion_matrix(test_y, pred_y), test_y, pr
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launc
her.py:46: UserWarning: Tight layout not applied. The left
and right margins cannot be made large enough to accommoda
te all axes decorations.
```



In [172]:

```
roc_auc_plot(test_y, pred_y, 'Bagging')
```



In [173]:

```
print(f'\033[1m Majority Class Precision:\033[0m {classification_report(
print(f'\033[1m Minority Class Recall:\033[0m {classification_report(tes
```

Majority Class Precision: 1.00
Minority Class Recall: 1.00

KNN

In [174]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [175]:

```
knn_mod = KNeighborsClassifier()
```

In [176]:

```
knn_mod.fit(train_X, train_y)
```

Out[176]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metri
c='minkowski',
                     metric_params=None, n_jobs=None, n_ne
ighbors=5, p=2,
                     weights='uniform')
```

In [177]:

```
pred_y = knn_mod.predict(test_X)
```

In [178]:

```
print(f'Accuracy %: {accuracy_score(test_y, pred_y) * 100}')
```

```
Accuracy %: 99.91011358374418
```

In [179]:

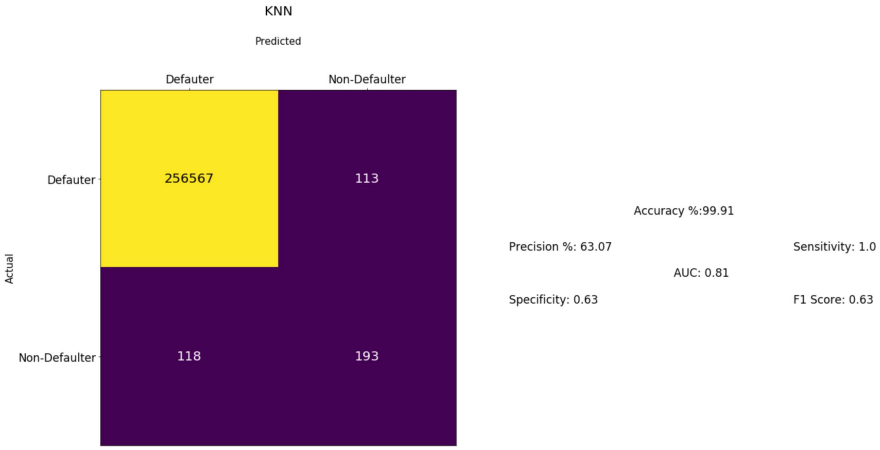
```
print('Classification Report')
print(classification_report(test_y, pred_y))
```

```
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    256680
           1       0.63      0.62      0.63       311

    accuracy                           1.00    256991
   macro avg       0.82      0.81      0.81    256991
weighted avg       1.00      1.00      1.00    256991
```

In [180]:

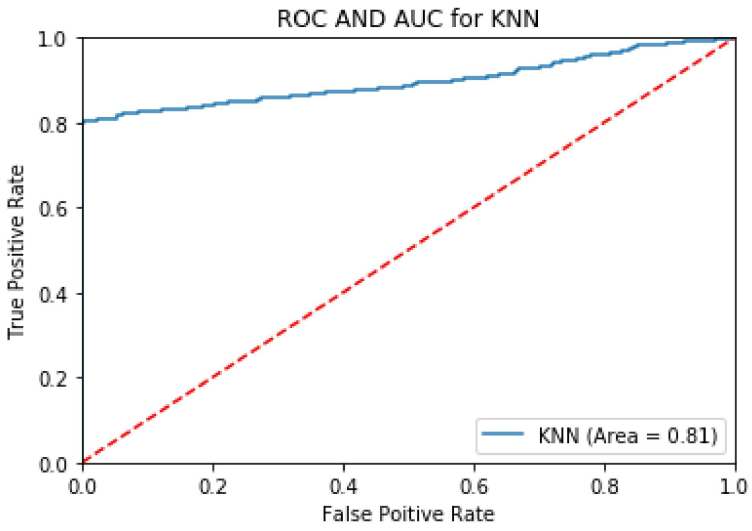
```
plot_ConfusionMatrix_metrics(confusion_matrix(test_y, pred_y), test_y, pr
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launc
her.py:46: UserWarning: Tight layout not applied. The left
and right margins cannot be made large enough to accommoda
te all axes decorations.
```



In [181]:

```
roc_auc_plot(test_y, pred_y, 'KNN')
```



In [182]:

```
print(f'\033[1m Majority Class Precision:\033[0m {classification_report(
print(f'\033[1m Minority Class Recall:\033[0m {classification_report(tes
```

Majority Class Precision: 1.00
Minority Class Recall: 0.62

Random Forest

In [183]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [184]:

```
ran_for_mod = RandomForestClassifier()
```

In [185]:

```
ran_for_mod.fit(train_X, train_y)
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\sklearn\ensembl
e\forest.py:245: FutureWarning: The default value of n_est
imators will change from 10 in version 0.20 to 100 in 0.2
2.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[185]:

```
RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='gini',
                       max_depth=None, max_features='aut
o', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impu
rity_split=None,
                       min_samples_leaf=1, min_samples_spl
it=2,
                       min_weight_fraction_leaf=0.0, n_est
imators=10,
                       n_jobs=None, oob_score=False, rando
m_state=None,
                       verbose=0, warm_start=False)
```

In [186]:

```
pred_y = ran_for_mod.predict(test_X)
```

In [187]:

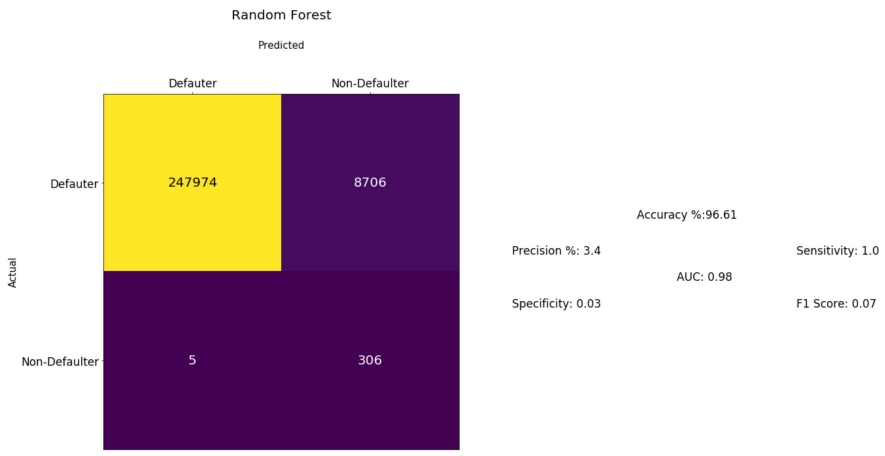
```
print('Classification Report')
print(classification_report(test_y, pred_y))
```

```
Classification Report
              precision    recall  f1-score   support

           0       1.00      0.97      0.98    256680
           1       0.03      0.98      0.07       311

    accuracy                           0.97    256991
   macro avg       0.52      0.98      0.52    256991
weighted avg       1.00      0.97      0.98    256991
```

In [188]:

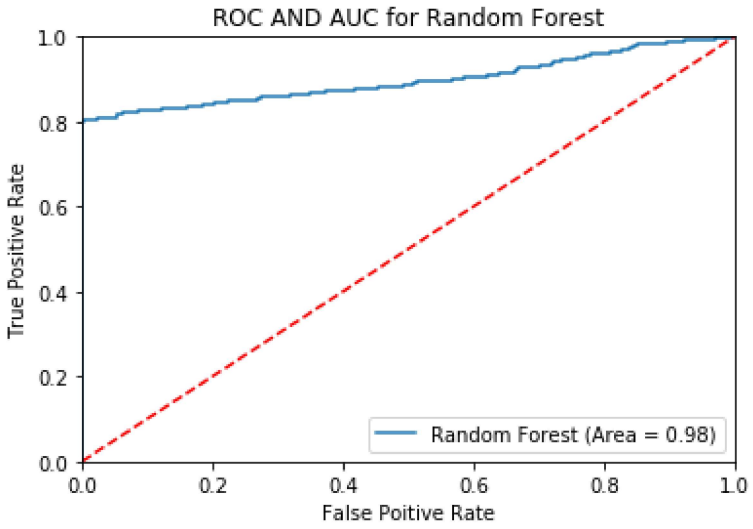
```
plot_ConfusionMatrix_metrics(confusion_matrix(test_y, pred_y), test_y, pr
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launc
her.py:46: UserWarning: Tight layout not applied. The left
and right margins cannot be made large enough to accommoda
te all axes decorations.
```



In [189]:

```
roc_auc_plot(test_y, pred_y, 'Random Forest')
```



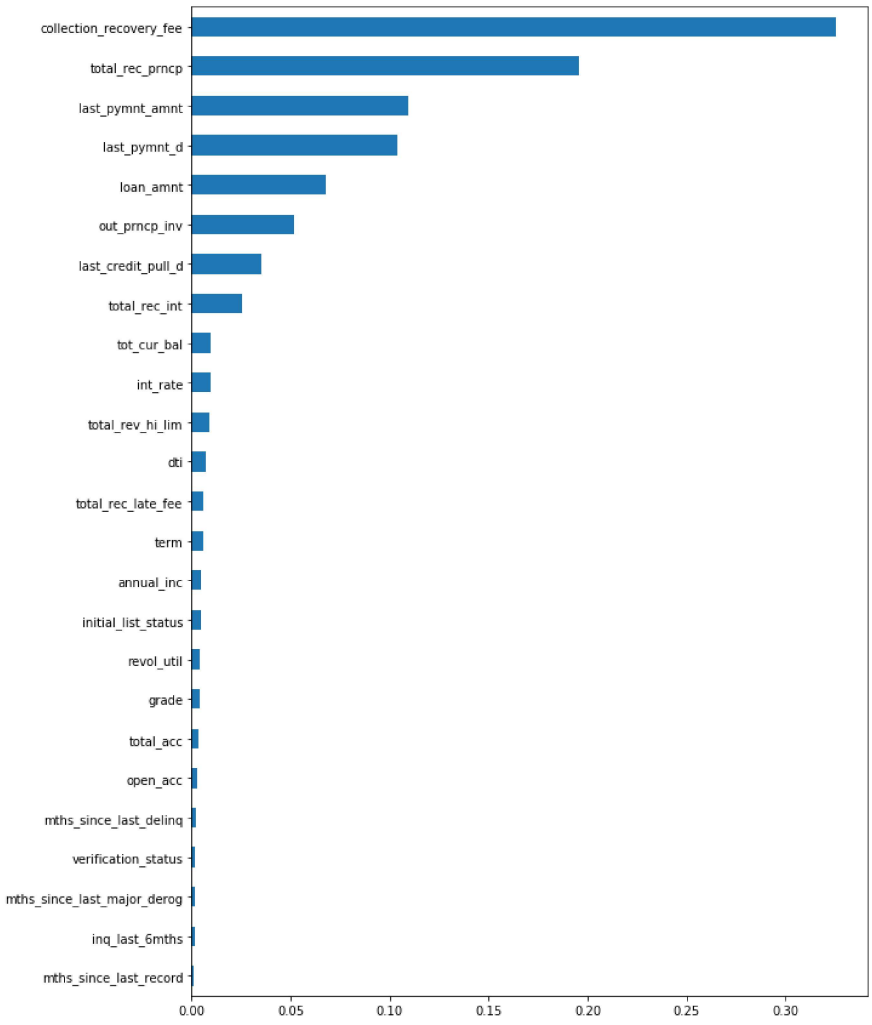
In [190]:

```
print(f'\033[1m Majority Class Precision:\033[0m {classification_report(
print(f'\033[1m Minority Class Recall:\033[0m {classification_report(tes
```

Majority Class Precision: 1.00
Minority Class Recall: 0.98

In [192]:

```
feat_importances = pd.Series(ran_for_mod.feature_importances_, index=tra
feat_importances.nlargest(25).plot(kind='barh', figsize = (10, 15)).inve
```



Fine Tuning

Since the data is imbalanced, oversampling using SMOTE and then running the models

In [193]:

```
from imblearn.over_sampling import SMOTE
```

```
Using TensorFlow backend.
```

In [194]:

```
print("Before OverSampling, counts of label '1': {}".format(sum(train_y =
print("Before OverSampling, counts of label '0': {} \n".format(sum(train_

sm = SMOTE(random_state=12, ratio = 1.0)
train_X_res, train_y_res = sm.fit_sample(train_X, train_y)

print('After OverSampling, the shape of train_X: {}'.format(train_X_res.s
print('After OverSampling, the shape of train_y: {} \n'.format(train_y_re

print("After OverSampling, counts of label '1': {}".format(sum(train_y_re
print("After OverSampling, counts of label '0': {}".format(sum(train_y_re
```

```
Before OverSampling, counts of label '1': 46156
Before OverSampling, counts of label '0': 552822

After OverSampling, the shape of train_X: (1105644, 50)
After OverSampling, the shape of train_y: (1105644,)

After OverSampling, counts of label '1': 552822
After OverSampling, counts of label '0': 552822
```

Model Building - Part 02

Logistic Regression

In [195]:

```
log_reg_mod.fit(train_X_res, train_y_res)
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\sklearn\linear_
model\logistic.py:432: FutureWarning: Default solver will
be changed to 'lbfgs' in 0.22. Specify a solver to silence
this warning.
  FutureWarning)
```

Out[195]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, f
it_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max
_iter=100,
                   multi_class='warn', n_jobs=None, penalt
y='l2',
                   random_state=None, solver='warn', tol=
0.0001, verbose=0,
                   warm_start=False)
```

In [196]:

```
pred_y = log_reg_mod.predict(test_X)
```

In [197]:

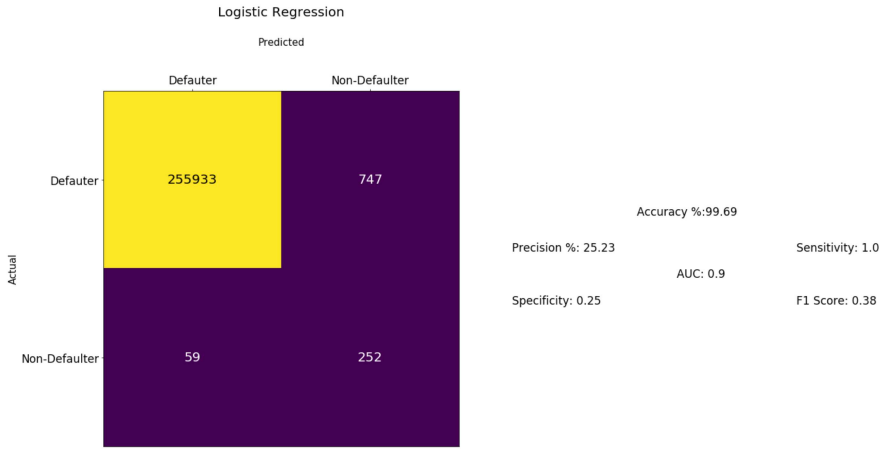
```
print('Classification Report')
print(classification_report(test_y, pred_y))
```

```
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    256680
           1       0.25      0.81      0.38       311

    accuracy                           1.00    256991
   macro avg       0.63      0.90      0.69    256991
weighted avg       1.00      1.00      1.00    256991
```

In [198]:

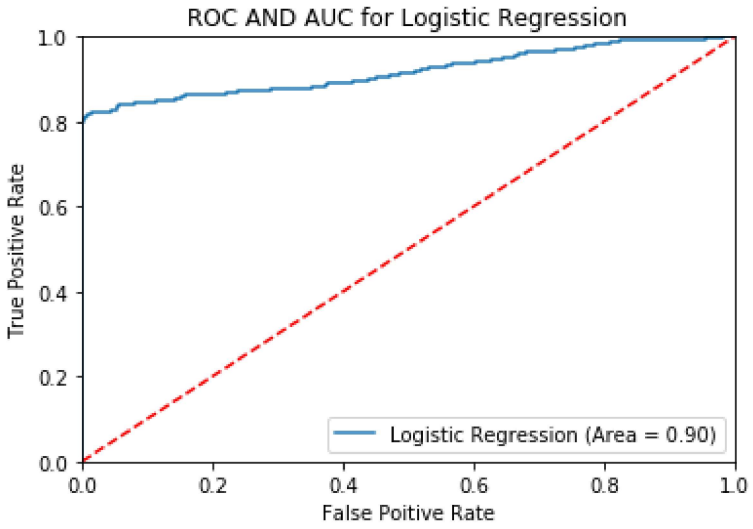
```
plot_ConfusionMatrix_metrics(confusion_matrix(test_y, pred_y), test_y, pr
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launc
her.py:46: UserWarning: Tight layout not applied. The left
and right margins cannot be made large enough to accommoda
te all axes decorations.
```



In [199]:

```
roc_auc_plot(test_y, pred_y, 'Logistic Regression')
```



In [200]:

```
print(f'\033[1m Majority Class Precision:\033[0m {classification_report(
print(f'\033[1m Minority Class Recall:\033[0m {classification_report(tes
```

Majority Class Precision: 1.00
Minority Class Recall: 0.81

Decision Tree

Via GINI Index

Lower the better (less impure)

In [201]:

```
dec_tre_gini.fit(train_X_res, train_y_res)
```

Out[201]:

```
DecisionTreeClassifier(class_weight=None, criterion='gin
i', max_depth=None,
                       max_features=None, max_leaf_nodes=N
one,
                       min_impurity_decrease=0.0, min_impu
rity_split=None,
                       min_samples_leaf=1, min_samples_spl
it=2,
                       min_weight_fraction_leaf=0.0, preso
rt=False,
                       random_state=None, splitter='best')
```

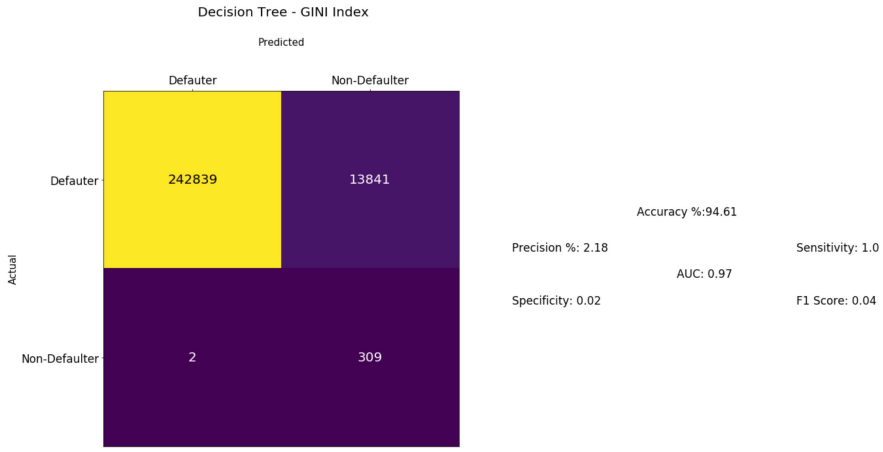
In [202]:

```
pred_y = dec_tre_gini.predict(test_X)
```

In [203]:

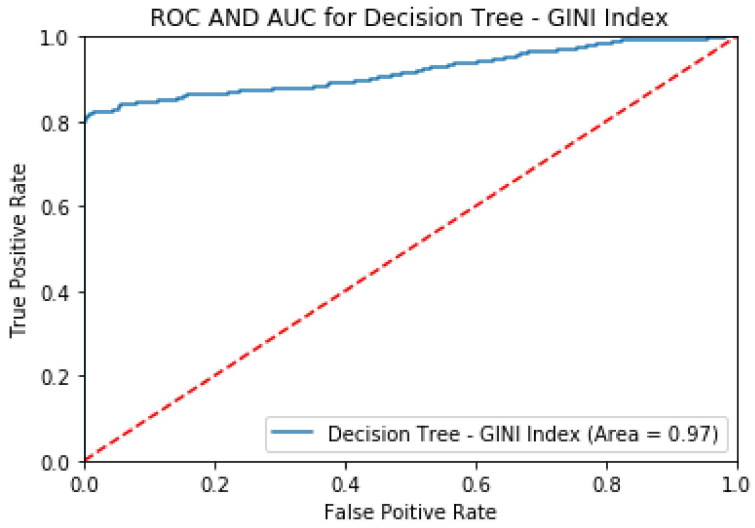
```
plot_ConfusionMatrix_metrics(confusion_matrix(test_y, pred_y), test_y, pr
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launc
her.py:46: UserWarning: Tight layout not applied. The left
and right margins cannot be made large enough to accommoda
te all axes decorations.
```



In [204]:

```
roc_auc_plot(test_y, pred_y, 'Decision Tree - GINI Index')
```



In [205]:

```
print(f'\033[1m Majority Class Precision:\033[0m {classification_report(
print(f'\033[1m Minority Class Recall:\033[0m {classification_report(tes
```

Majority Class Precision: 1.00
Minority Class Recall: 0.99

Via Entropy

In [206]:

```
dec_tre_ent.fit(train_X_res, train_y_res)
```

Out[206]:

```
DecisionTreeClassifier(class_weight=None, criterion='entro
py', max_depth=None,
                       max_features=None, max_leaf_nodes=N
one,
                       min_impurity_decrease=0.0, min_impu
rity_split=None,
                       min_samples_leaf=1, min_samples_spl
it=2,
                       min_weight_fraction_leaf=0.0, preso
rt=False,
                       random_state=None, splitter='best')
```

In [207]:

```
pred_y = dec_tre_ent.predict(test_X)
```

In [208]:

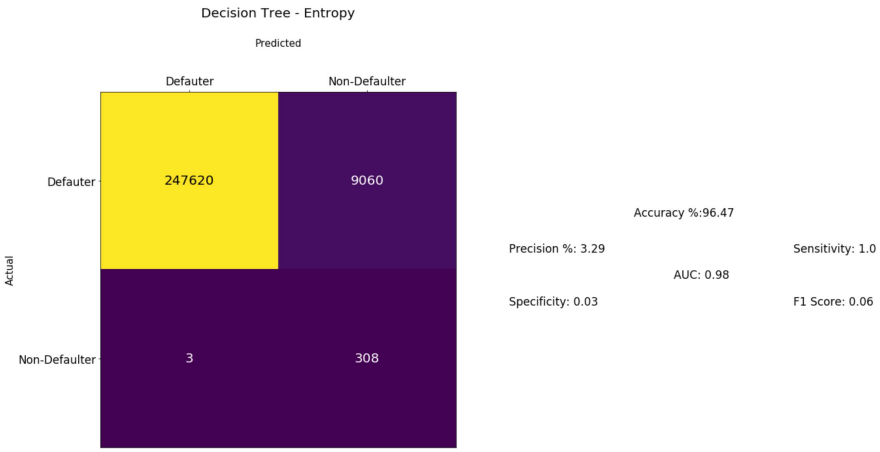
```
print('Classification Report')
print(classification_report(test_y, pred_y))
```

```
Classification Report
              precision    recall  f1-score   support

           0       1.00      0.96      0.98    256680
           1       0.03      0.99      0.06       311

    accuracy                           0.96    256991
   macro avg       0.52      0.98      0.52    256991
weighted avg       1.00      0.96      0.98    256991
```

In [209]:

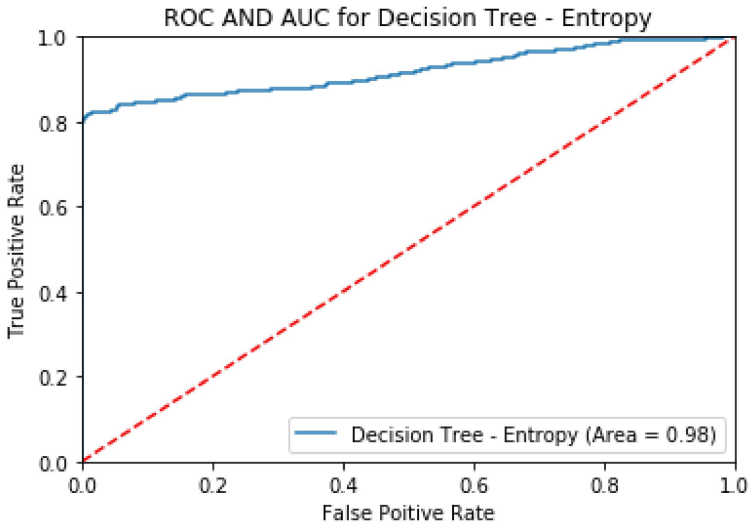
```
plot_ConfusionMatrix_metrics(confusion_matrix(test_y, pred_y), test_y, pr
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launc
her.py:46: UserWarning: Tight layout not applied. The left
and right margins cannot be made large enough to accommoda
te all axes decorations.
```



In [210]:

```
roc_auc_plot(test_y, pred_y, 'Decision Tree - Entropy')
```



In [211]:

```
print(f'\033[1m Majority Class Precision:\033[0m {classification_report(
print(f'\033[1m Minority Class Recall:\033[0m {classification_report(tes
```

Majority Class Precision: 1.00
Minority Class Recall: 0.99

Bagging - Boot-strap AGGregation

In [212]:

```
bag_mod = BaggingClassifier()
```

In [213]:

```
bag_mod.fit(train_X_res, train_y_res)
```

Out[213]:

```
BaggingClassifier(base_estimator=None, bootstrap=True, boo
tstrap_features=False,
                  max_features=1.0, max_samples=1.0, n_est
imators=10,
                  n_jobs=None, oob_score=False, random_sta
te=None, verbose=0,
                  warm_start=False)
```

In [214]:

```
pred_y = bag_mod.predict(test_X)
```

In [215]:

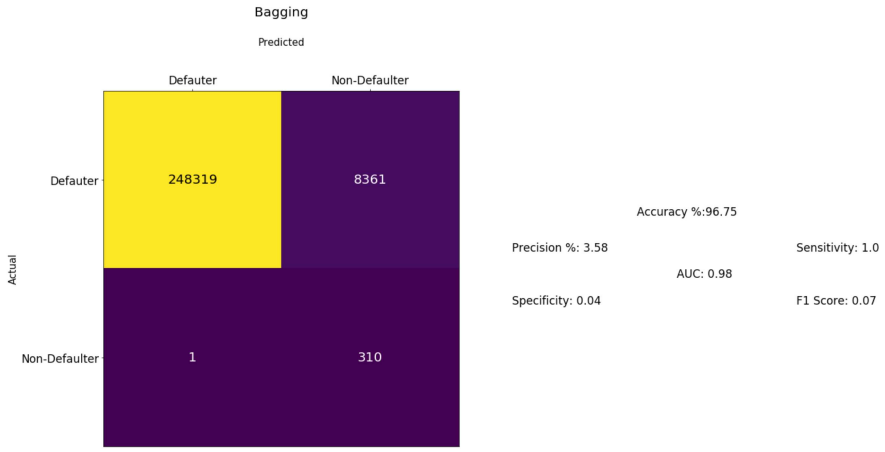
```
print('Classification Report')
print(classification_report(test_y, pred_y))
```

```
Classification Report
              precision    recall  f1-score   support

           0       1.00      0.97      0.98    256680
           1       0.04      1.00      0.07       311

    accuracy                           0.97    256991
   macro avg       0.52      0.98      0.53    256991
weighted avg       1.00      0.97      0.98    256991
```

In [216]:

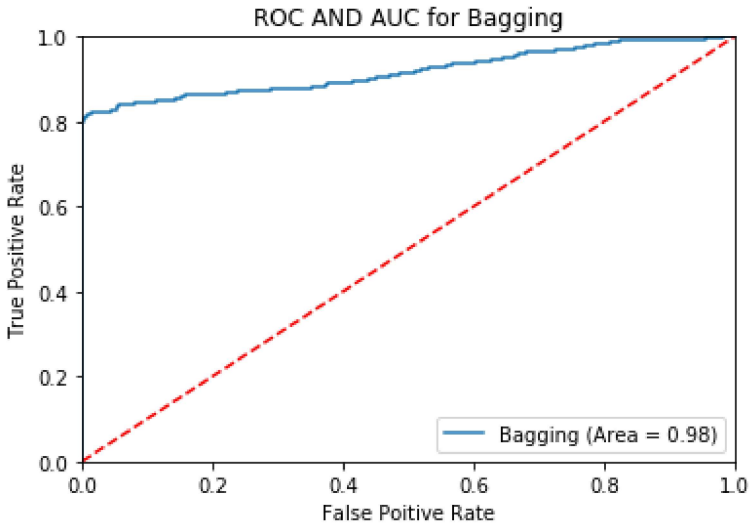
```
plot_ConfusionMatrix_metrics(confusion_matrix(test_y, pred_y), test_y, pr
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launc
her.py:46: UserWarning: Tight layout not applied. The left
and right margins cannot be made large enough to accommoda
te all axes decorations.
```



In [217]:

```
roc_auc_plot(test_y, pred_y, 'Bagging')
```



In [218]:

```
print(f'\033[1m Majority Class Precision:\033[0m {classification_report(
print(f'\033[1m Minority Class Recall:\033[0m {classification_report(tes
```

Majority Class Precision: 1.00
Minority Class Recall: 1.00

KNN

In [219]:

```
knn_mod = KNeighborsClassifier()
```

In [220]:

```
knn_mod.fit(train_X_res, train_y_res)
```

Out[220]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metri
c='minkowski',
                     metric_params=None, n_jobs=None, n_ne
ighbors=5, p=2,
                     weights='uniform')
```

In [221]:

```
pred_y = knn_mod.predict(test_X)
```

In [222]:

```
print(f'Accuracy %: {accuracy_score(test_y, pred_y) * 100}')
```

```
Accuracy %: 96.8944437742956
```

In [223]:

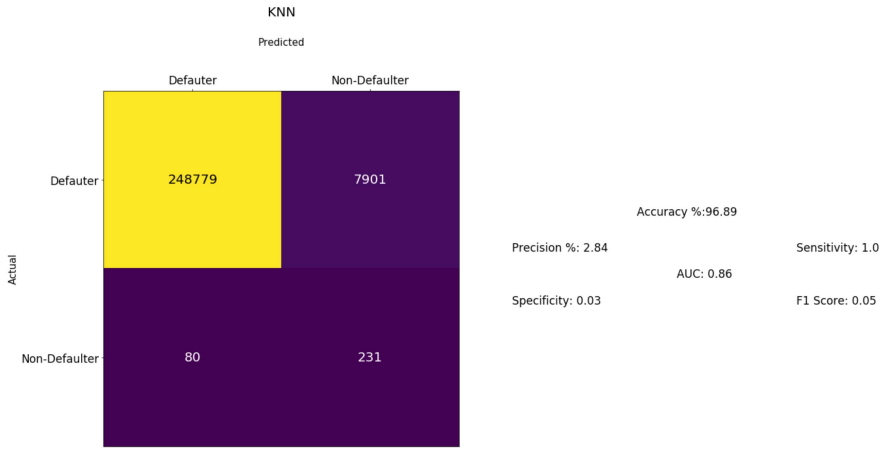
```
print('Classification Report')
print(classification_report(test_y, pred_y))
```

```
Classification Report
              precision    recall  f1-score   support

           0       1.00      0.97      0.98    256680
           1       0.03      0.74      0.05       311

    accuracy                           0.97    256991
   macro avg       0.51      0.86      0.52    256991
weighted avg       1.00      0.97      0.98    256991
```

In [224]:

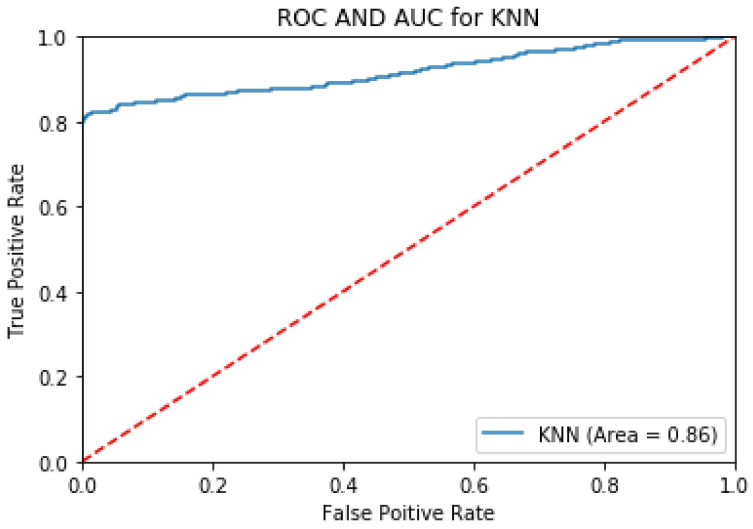
```
plot_ConfusionMatrix_metrics(confusion_matrix(test_y, pred_y), test_y, pr
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launc
her.py:46: UserWarning: Tight layout not applied. The left
and right margins cannot be made large enough to accommoda
te all axes decorations.
```



In [225]:

```
roc_auc_plot(test_y, pred_y, 'KNN')
```



In [226]:

```
print(f'\033[1m Majority Class Precision:\033[0m {classification_report(
print(f'\033[1m Minority Class Recall:\033[0m {classification_report(tes
```

Majority Class Precision: 1.00
Minority Class Recall: 0.74

Random Forest

In [227]:

```
ran_for_mod = RandomForestClassifier()
```

In [228]:

```
ran_for_mod.fit(train_X_res, train_y_res)
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\sklearn\ensembl
e\forest.py:245: FutureWarning: The default value of n_est
imators will change from 10 in version 0.20 to 100 in 0.2
2.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[228]:

```
RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='gini',
                       max_depth=None, max_features='aut
o', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impu
rity_split=None,
                       min_samples_leaf=1, min_samples_spl
it=2,
                       min_weight_fraction_leaf=0.0, n_est
imators=10,
                       n_jobs=None, oob_score=False, rando
m_state=None,
                       verbose=0, warm_start=False)
```

In [229]:

```
pred_y = ran_for_mod.predict(test_X)
```

In [230]:

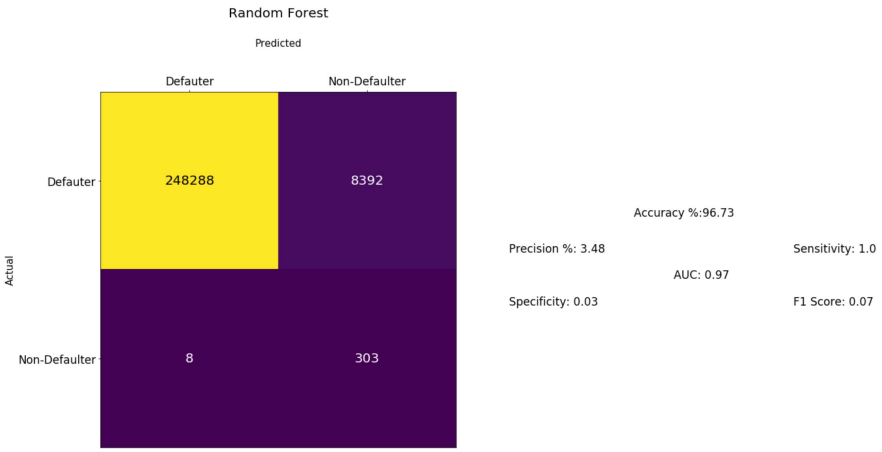
```
print('Classification Report')
print(classification_report(test_y, pred_y))
```

```
Classification Report
              precision    recall  f1-score   support

           0       1.00      0.97      0.98    256680
           1       0.03      0.97      0.07       311

    accuracy                           0.97    256991
   macro avg       0.52      0.97      0.53    256991
weighted avg       1.00      0.97      0.98    256991
```

In [231]:

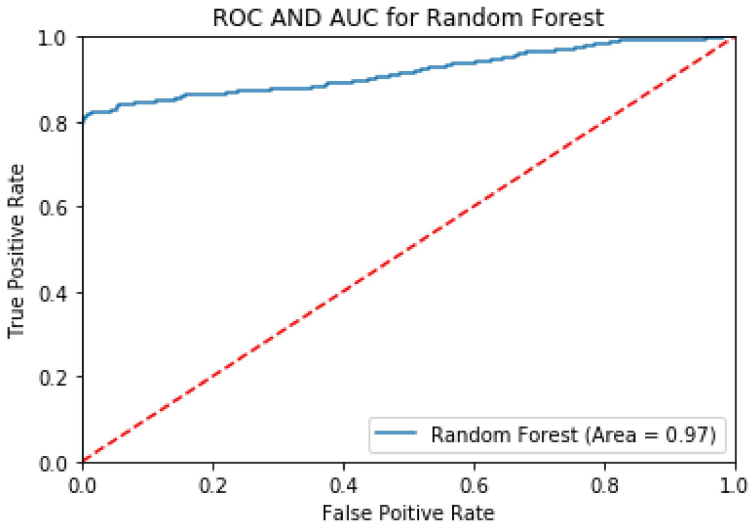
```
plot_ConfusionMatrix_metrics(confusion_matrix(test_y, pred_y), test_y, pr
```

```
C:\Users\sragh\Anaconda3\lib\site-packages\ipykernel_launc
her.py:46: UserWarning: Tight layout not applied. The left
and right margins cannot be made large enough to accommoda
te all axes decorations.
```



In [232]:

```
roc_auc_plot(test_y, pred_y, 'Random Forest')
```



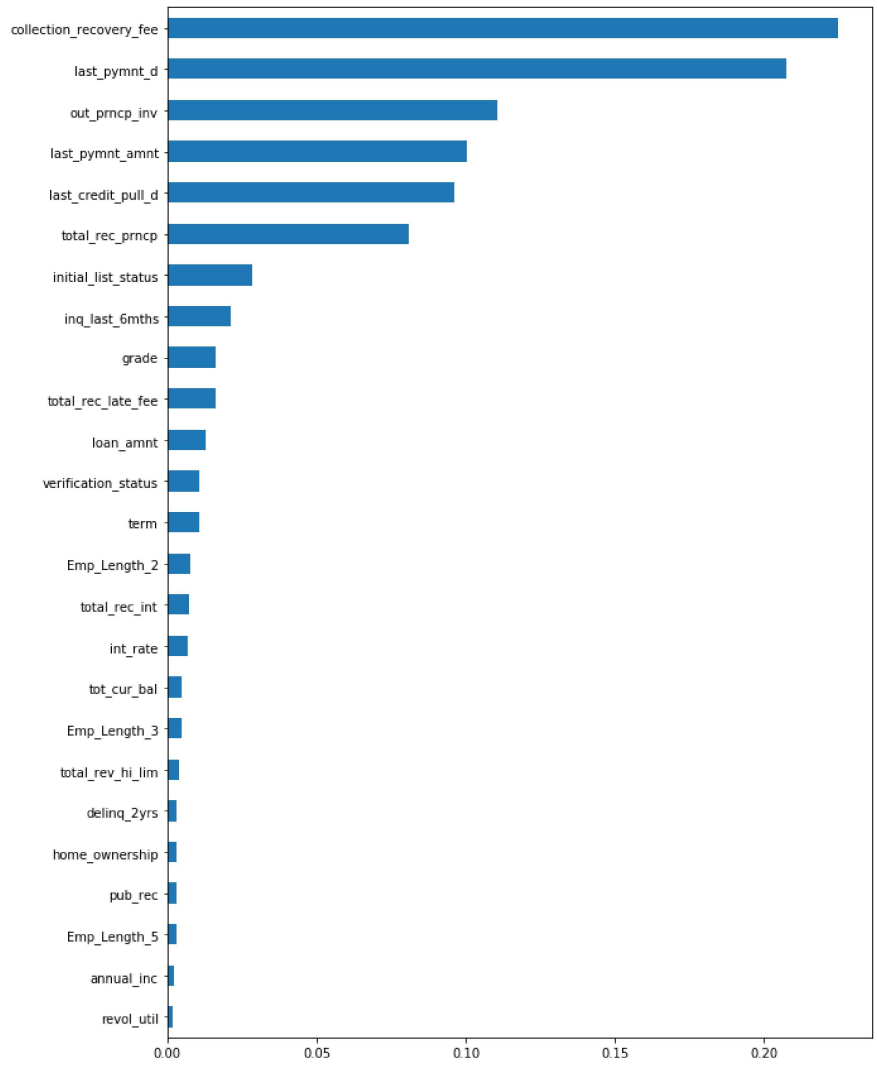
In [233]:

```
print(f'\033[1m Majority Class Precision:\033[0m {classification_report(
print(f'\033[1m Minority Class Recall:\033[0m {classification_report(tes
```

Majority Class Precision: 1.00
Minority Class Recall: 0.97

In [235]:

```
feat_importances = pd.Series(ran_for_mod.feature_importances_, index=tra
feat_importances.nlargest(25).plot(kind='barh', figsize = (10, 15)).inve
```



As seen if the loan defaulter prediction are more important then Logistic is the better model, whereas for loan non-defaulter prediction Bagging proves as a better model