Loan (3)

May 23, 2019

1 Homework 3 - Loan

Name: Troy Zhongyi Zhang
Netid: zhongyiz@uchicago.edu

1.1 Data Pre-processing

```
In [232]: import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          df2 = pd.read_csv("Data for Cleaning & Modeling.csv", header = None)
          loan = df2.drop([7, 9,14,15,17,24,25],axis=1)
          loan
```

```
D:\Users\Owners\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2698: DtypeWarning
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[232]:

	0	1	2	3	4	5	6
0	X1	X2	X3	X4	X5	X6	X7
1	11.89%	54734	80364	$25,000	$25,000	$19,080	36 months
2	10.71%	55742	114426	$7,000	$7,000	$673	36 months
3	16.99%	57167	137225	$25,000	$25,000	$24,725	36 months
4	13.11%	57245	138150	$1,200	$1,200	$1,200	36 months
5	13.57%	57416	139635	$10,800	$10,800	$10,692	36 months
6	19.05%	58524	149512	$7,200	$7,200	$7,200	36 months
7	10.08%	58915	153417	$7,500	$5,025	$557	36 months
8	14.26%	59006	154254	$3,000	$3,000	$2,988	36 months
9	7.88%	61390	182594	$4,000	$4,000	$3,900	36 months
10	14.96%	61419	182917	$5,600	$5,600	$5,525	36 months
11	9.88%	62102	191024	$3,200	$3,200	$3,200	36 months
12	11.14%	65426	232106	$4,000	$4,000	$3,892	36 months
13	11.34%	65640	234569	$5,000	$2,650	$495	36 months
14	12.21%	66431	243540	$2,525	$2,525	$2,375	36 months
15	13.47%	66749	246329	$10,625	$10,625	$5,325	36 months
16	11.49%	66943	247802	$2,800	$2,800	$2,700	60 months
17	13.24%	66964	247990	$7,500	$7,500	$986	36 months
18	8.59%	67503	252415	$10,000	$10,000	$10,000	36 months

19	7.14%	68163	258249	$3,000	$3,000	$3,000	36 months
20	8.63%	68381	260179	$6,625	$6,625	$6,475	36 months
21	11.03%	68817	264119	$10,000	$10,000	$2,475	36 months
22	NaN	68926	264924	$2,300	$2,300	$590	36 months
23	8.94%	69001	265533	$15,000	$15,000	$14,875	36 months
24	10.39%	69124	266619	$18,000	$18,000	$11,636	36 months
25	15.13%	69168	266943	$5,000	$5,000	$4,975	36 months
26	8.00%	69251	267771	$6,000	$6,000	$500	36 months
27	15.33%	69550	270212	$11,200	$11,200	$11,188	60 months
28	8.63%	69828	272798	$15,000	$15,000	$13,138	36 months
29	13.55%	69924	274280	$10,000	$10,000	$8,790	36 months
...	...	...	...	...	...	...	...
399971	13.98%	28752830	31225959	$5,000	$5,000	$5,000	36 months
399972	9.17%	28752839	31225968	$12,500	$12,500	$12,500	36 months
399973	8.39%	28752840	31225969	$12,000	$12,000	$12,000	36 months
399974	16.29%	28752841	31225970	$12,800	$12,800	$12,800	60 months
399975	NaN	28752846	31225976	$6,400	$6,400	$6,400	36 months
399976	13.98%	28752855	31225985	$25,900	$25,900	$25,900	36 months
399977	7.12%	28752856	31225986	$8,000	$8,000	$8,000	36 months
399978	18.24%	28752859	31225989	$14,375	$14,375	$14,350	36 months
399979	16.99%	28752860	31225990	$7,500	$7,500	$7,500	36 months
399980	9.17%	28752862	31225992	$5,000	$5,000	$5,000	36 months
399981	20.20%	28752866	31225999	$30,000	$30,000	$30,000	60 months
399982	12.49%	28752869	31226001	$5,000	$5,000	$5,000	36 months
399983	20.20%	28752893	31226025	$25,000	$25,000	$25,000	60 months
399984	15.61%	28752896	31226028	$6,000	$6,000	$6,000	36 months
399985	15.61%	28752917	31226049	$23,325	$23,325	$23,325	36 months
399986	11.67%	28752929	31226063	$20,000	$20,000	$20,000	60 months
399987	7.12%	28752934	31226068	$15,000	$15,000	$15,000	36 months
399988	13.98%	28752960	31226095	$8,000	$8,000	$8,000	36 months
399989	15.61%	28752969	31226104	$2,000	$2,000	$2,000	36 months
399990	14.99%	28752981	31226117	$14,100	$14,100	$14,100	36 months
399991	14.49%	28752994	31226130	$7,500	$7,500	$7,500	36 months
399992	17.57%	28753004	31226140	$26,000	$26,000	$26,000	60 months
399993	13.98%	28753020	31226156	$20,000	$20,000	$20,000	36 months
399994	17.57%	28753030	31226166	$8,000	$8,000	$8,000	36 months
399995	NaN	28753078	31226214	$10,000	$10,000	$10,000	60 months
399996	12.99%	28753086	31226222	$10,000	$10,000	$10,000	60 months
399997	16.29%	28753097	31226234	$13,150	$13,150	$13,150	36 months
399998	10.99%	28753099	31226236	$20,000	$20,000	$20,000	60 months
399999	17.57%	28753118	31226256	$18,475	$18,475	$18,475	60 months
400000	13.35%	28753146	31226285	$16,000	$16,000	$16,000	36 months

	8	10	11	...	20	21	22	23	26	27	\
0	X9	X11	X12	...	X21	X22	X23	X24	X27	X28	
1	B4	< 1 year	RENT	...	19.48	0	Feb-94	0	10	0	
2	B5	< 1 year	RENT	...	14.29	0	Oct-00	0	7	0	
3	D3	1 year	RENT	...	10.5	0	Jun-00	0	10	0	

4	C2	10+ years	OWN	...	5.47	0	Jan-85	0	5	0
5	C3	6 years	RENT	...	11.63	0	Dec-96	1	14	0
6	D4	9 years	RENT	...	2.05	0	Apr-94	0	6	0
7	B3	3 years	RENT	...	8.1	0	Nov-00	1	3	0
8	C5	3 years	MORTGAGE	...	14.97	1	Jul-98	0	13	0
9	A5	< 1 year	MORTGAGE	...	16.98	0	May-93	0	11	0
10	D2	1 year	RENT	...	4	0	Jun-01	0	5	1
11	B1	5 years	RENT	...	6.51	0	Jun-06	0	7	0
12	B1	< 1 year	MORTGAGE	...	11.08	0	Aug-95	0	14	0
13	C2	10+ years	MORTGAGE	...	17.25	0	May-97	1	20	0
14	B5	3 years	RENT	...	10	0	Sep-01	1	5	0
15	C4	10+ years	MORTGAGE	...	22.09	0	Jul-90	2	5	1
16	B4	< 1 year	RENT	...	4	0	Dec-00	0	7	1
17	D3	3 years	MORTGAGE	...	9.16	1	Aug-96	2	11	0
18	A4	1 year	MORTGAGE	...	15.49	0	Dec-91	0	7	0
19	A3	9 years	MORTGAGE	...	6.5	1	Feb-98	0	14	0
20	A5	< 1 year	RENT	...	9.73	0	Dec-03	0	3	0
21	C1	10+ years	MORTGAGE	...	13.04	0	Nov-85	1	7	0
22	D2	10+ years	RENT	...	2.26	0	Dec-97	0	4	0
23	A5	< 1 year	MORTGAGE	...	7.07	0	Nov-91	1	6	0
24	B4	10+ years	MORTGAGE	...	3.8	0	Aug-00	0	2	0
25	E4	2 years	RENT	...	2.74	0	Feb-05	3	2	0
26	A3	< 1 year	MORTGAGE	...	16.08	0	Dec-94	1	16	0
27	D3	2 years	RENT	...	17.19	0	Oct-05	1	7	0
28	A5	10+ years	OWN	...	2.59	0	Jun-75	0	4	0
29	D4	3 years	RENT	...	7.94	0	Nov-02	0	11	0
...	...	...	...	...	...	...	...	...	...	...
399971	C3	7 years	NaN	...	15.52	0	Feb-07	2	9	0
399972	B1	5 years	OWN	...	17.15	1	Nov-95	0	15	0
399973	A5	n/a	NaN	...	24.56	0	Jul-00	2	10	0
399974	D2	3 years	MORTGAGE	...	21.45	1	Apr-05	1	7	0
399975	NaN	6 years	MORTGAGE	...	24.97	0	Dec-01	0	31	0
399976	C3	< 1 year	RENT	...	17.16	0	Mar-03	0	15	8
399977	A3	10+ years	MORTGAGE	...	17.82	1	Aug-94	0	16	0
399978	D5	4 years	NaN	...	29.63	0	Jul-11	1	6	0
399979	NaN	n/a	RENT	...	20.78	0	Aug-02	1	4	0
399980	B1	3 years	RENT	...	6.77	1	Oct-90	0	10	0
399981	NaN	10+ years	MORTGAGE	...	9.85	0	Dec-01	1	6	0
399982	B5	n/a	RENT	...	14.2	0	Jan-70	0	10	1
399983	E3	n/a	NaN	...	25.5	0	Dec-99	0	16	0
399984	NaN	2 years	RENT	...	30.27	0	Sep-08	1	9	0
399985	D1	10+ years	MORTGAGE	...	20	0	Apr-88	0	16	0
399986	NaN	5 years	NaN	...	26	2	Feb-91	0	17	0
399987	NaN	10+ years	MORTGAGE	...	18.26	0	Feb-95	0	24	0
399988	C3	2 years	RENT	...	34.06	0	Sep-99	3	13	1
399989	D1	4 years	MORTGAGE	...	7.58	0	Sep-94	1	7	0
399990	C5	n/a	RENT	...	16.86	1	Aug-02	0	9	1
399991	NaN	1 year	NaN	...	10.44	0	May-07	2	8	0

399992	D4	10+ years	MORTGAGE	...	17.99	1	Jun-83	3	15	1
399993	C3	3 years	OWN	...	5.53	0	Apr-87	2	8	2
399994	D4	< 1 year	OWN	...	26.93	0	Sep-91	2	7	0
399995	C5	10+ years	MORTGAGE	...	13.91	3	Jul-93	2	13	0
399996	C1	8 years	RENT	...	21.51	0	Nov-03	0	9	0
399997	D2	1 year	OWN	...	29.76	0	Oct-07	0	11	0
399998	B3	1 year	MORTGAGE	...	24.13	0	Oct-04	0	14	0
399999	D4	10+ years	OWN	...	31.43	0	Mar-94	0	19	0
400000	C2	4 years	NaN	...	16.48	0	Sep-10	0	9	0

	28	29	30	31
0	X29	X30	X31	X32
1	28854	52.10%	42	f
2	33623	76.70%	7	f
3	19878	66.30%	17	f
4	2584	40.40%	31	f
5	3511	25.60%	40	f
6	3874	90.10%	25	f
7	33667	73.20%	11	f
8	4740	39.50%	23	f
9	50807	51%	19	f
10	3839	76.80%	9	f
11	3198	51.10%	11	f
12	24220	68.60%	33	f
13	69909	51.10%	51	f
14	5630	23%	8	f
15	13846	71%	16	f
16	2183	19.50%	23	f
17	5122	18.20%	37	f
18	6068	16.70%	49	f
19	3021	4.80%	25	f
20	6282	44.60%	9	f
21	5394	53.40%	23	f
22	2211	88.40%	13	f
23	7586	52.70%	19	f
24	8311	59.80%	9	f
25	591	84.40%	6	f
26	29797	23.20%	39	f
27	7248	69.20%	11	f
28	5656	27.60%	25	f
29	21162	57.70%	14	f
...	...	...	...	...
399971	13912	70.30%	11	f
399972	2080	13.30%	36	w
399973	96700	43.90%	18	w
399974	0	0%	27	w
399975	13157	31.90%	42	w
399976	37646	46.20%	38	f

```
399977   7770  13.70%   36    f
399978   4158     66%    9    f
399979   5881  96.40%    6    w
399980   6554     69%   13    f
399981  27326     79%   14    w
399982   9798  57.60%   41    f
399983  49994    101%   43    f
399984   8372  61.10%    9    f
399985  26082  77.90%   42    w
399986  12133  80.90%   28    w
399987  45624  43.40%   64    f
399988  11645  52.50%   28    f
399989   6928  62.30%   29    f
399990   7529  42.30%   18    w
399991   5598  48.30%   15    f
399992  27890  55.90%   30    w
399993  13818  62.20%   12    f
399994   1295  61.70%   11    w
399995   4676  64.10%   30    f
399996  10268  76.10%   20    w
399997   8931  37.80%   21    f
399998  28976  69.30%   48    w
399999  11982     39%   31    f
400000   3864  53.70%   12    f

[400001 rows x 25 columns]
```

```
In [233]: loan['ir']=loan[0]
          loan = loan[pd.notnull(loan['ir'])]

In [234]: loan = loan.drop([0])

In [235]: #X2 and X3
          loan[1] = loan[1].astype(float)
          loan[2] = loan[2].astype(float)

In [236]: #X4
          loan[3] = loan[3].str.replace('$','')
          loan[3] = loan[3].str.replace(',','')
          loan[3] = loan[3].replace('',np.nan).astype(float)
          loan[3] = loan[3].fillna((loan[3].mean()))

In [237]: #X5
          loan[4] = loan[4].str.replace('$','')
          loan[4] = loan[4].str.replace(',','')
          loan[4] = loan[4].replace('',np.nan).astype(float)
          loan[4] = loan[4].fillna((loan[4].mean()))

In [238]: #X6
          loan[5] = loan[5].str.replace('$','')
```

```
loan[5] = loan[5].str.replace(',','')
loan[5] = loan[5].replace('',np.nan).astype(float)
loan[5] = loan[5].fillna((loan[5].mean()))
#loan[5] = loan[5].astype('category').cat.codes
```

In [239]:
```
#X7
loan['month'] = loan[6].astype(str).str[0:3]
loan['month'] = pd.to_numeric(loan['month'], errors='coerce').fillna(36).astype(np.i
loan['month'] = loan['month'].astype(float)
loan['month'] = loan['month'].astype('category').cat.codes
```

In [240]:
```
#X9
loan[8] = loan[8].replace(['A1'], '0')
loan[8] = loan[8].replace(['A2'], '1')
loan[8] = loan[8].replace(['A3'], '2')
loan[8] = loan[8].replace(['A4'], '3')
loan[8] = loan[8].replace(['A5'], '4')

loan[8] = loan[8].replace(['B1'], '5')
loan[8] = loan[8].replace(['B2'], '6')
loan[8] = loan[8].replace(['B3'], '7')
loan[8] = loan[8].replace(['B4'], '8')
loan[8] = loan[8].replace(['B5'], '9')

loan[8] = loan[8].replace(['C1'], '10')
loan[8] = loan[8].replace(['C2'], '11')
loan[8] = loan[8].replace(['C3'], '12')
loan[8] = loan[8].replace(['C4'], '13')
loan[8] = loan[8].replace(['C5'], '14')

loan[8] = loan[8].replace(['D1'], '15')
loan[8] = loan[8].replace(['D2'], '16')
loan[8] = loan[8].replace(['D3'], '17')
loan[8] = loan[8].replace(['D4'], '18')
loan[8] = loan[8].replace(['D5'], '19')

loan[8] = loan[8].replace(['E1'], '20')
loan[8] = loan[8].replace(['E2'], '21')
loan[8] = loan[8].replace(['E3'], '22')
loan[8] = loan[8].replace(['E4'], '23')
loan[8] = loan[8].replace(['E5'], '24')

loan[8] = loan[8].replace(['F1'], '25')
loan[8] = loan[8].replace(['F2'], '26')
loan[8] = loan[8].replace(['F3'], '27')
loan[8] = loan[8].replace(['F4'], '28')
loan[8] = loan[8].replace(['F5'], '29')
```

```
loan[8] = loan[8].replace(['G1'], '30')
loan[8] = loan[8].replace(['G2'], '31')
loan[8] = loan[8].replace(['G3'], '32')
loan[8] = loan[8].replace(['G4'], '33')
loan[8] = loan[8].replace(['G5'], '34')

loan[8] = pd.to_numeric(loan[8], errors='coerce').fillna(0).astype(np.int64)
loan[8] = loan[8].astype(float)
loan[8] = loan[8].astype('category').cat.codes
```

In [241]:
```
#X11
loan[10] = loan[10].replace(['< 1 year'], '0')
loan[10] = loan[10].replace(['1 year'], '1')
loan[10] = loan[10].replace(['2 years'], '2')
loan[10] = loan[10].replace(['3 years'], '3')
loan[10] = loan[10].replace(['4 years'], '4')
loan[10] = loan[10].replace(['5 years'], '5')
loan[10] = loan[10].replace(['6 years'], '6')
loan[10] = loan[10].replace(['7 years'], '7')
loan[10] = loan[10].replace(['8 years'], '8')
loan[10] = loan[10].replace(['9 years'], '9')
loan[10] = loan[10].replace(['10+ years'], '10')
loan[10] = loan[10].replace(['n/a'], '0')

loan[10] = pd.to_numeric(loan[10], errors='coerce').fillna(0).astype(np.int64)
loan[10] = loan[10].astype(float)
loan[10] = loan[10].astype('category').cat.codes
```

In [242]:
```
#X12
loan[11] = loan[11].replace(['MORTGAGE'], '0')
loan[11] = loan[11].replace(['RENT'], '1')
loan[11] = loan[11].replace(['OWN'], '2')
loan[11] = loan[11].replace(['OTHER'], '3')
loan[11] = loan[11].replace(['NONE'], '4')
loan[11] = loan[11].replace(['ANY'], '5')

loan[11] = pd.to_numeric(loan[11], errors='coerce').fillna(0).astype(np.int64)
loan[11] = loan[11].astype(float)
loan[11] = loan[11].astype('category').cat.codes
```

In [243]:
```
#X13
#loan[12] = loan[12].astype(str)
#loan[12] = pd.to_numeric(loan[12], errors='coerce').fillna(loan[12].mean()).astype(
#loan[12] = loan[12].astype(float)

loan[12] = loan[12].replace('',np.nan).astype(float)
loan[12] = loan[12].fillna((loan[12].mean()))
loan[12] = round(loan[12],2)
```

In [244]:
```
# X14
loan[13] = loan[13].replace(['VERIFIED - income'], '0')
loan[13] = loan[13].replace(['not verified'], '1')
loan[13] = loan[13].replace(['VERIFIED - income source'], '2')

loan[13] = pd.to_numeric(loan[13], errors='coerce').fillna(0).astype(np.int64)
loan[13] = loan[13].astype(float)
loan[13] = loan[13].astype('category').cat.codes
```

In [245]:
```
# X17
loan[16] = loan[16].str.replace('debt_consolidation','0')
loan[16] = loan[16].str.replace('credit_card','1')
loan[16] = loan[16].str.replace('car','2')
loan[16] = loan[16].str.replace('educational','3')
loan[16] = loan[16].str.replace('home_improvement','4')
loan[16] = loan[16].str.replace('house','5')
loan[16] = loan[16].str.replace('major_purchase','6')
loan[16] = loan[16].str.replace('medical','7')
loan[16] = loan[16].str.replace('moving','8')
loan[16] = loan[16].str.replace('other','9')
loan[16] = loan[16].str.replace('renewable_energy','10')
loan[16] = loan[16].str.replace('small_business','11')
loan[16] = loan[16].str.replace('vacation','12')
loan[16] = loan[16].str.replace('wedding','13')
loan['title'] = pd.to_numeric(loan[16], errors='coerce').fillna(0).astype(np.int64)
loan['title'] = loan['title'].astype(float)
loan['title'] = loan['title'].astype('category').cat.codes
```

In [246]:
```
#import ast
#loan[33] = loan[33].apply(ast.literal_eval)
```

In [247]:
```
#X19
loan['zip'] = loan[18].astype(str).str[0]
#loan.iloc[0:8,22:]
#loan[33] = pd.factorize(loan[33])[0]

loan['zip'] = pd.to_numeric(loan['zip'], errors='coerce').fillna(0).astype(np.int64)
loan['zip'] = loan['zip'].astype(float)
loan['zip']=loan['zip'].astype('category').cat.codes
```

In [248]:
```
#X20
#State:
#loan[19]=loan[19].replace('100',np.nan).astype(float)
loan[19] = loan[19].replace(['CT','ME','MA','NH','RI','VT'], '0')
loan[19] = loan[19].replace(['NJ', 'NY','PA'], '1')
loan[19] = loan[19].replace(['IL', 'IN','MI','OH','WI'], '2')
loan[19] = loan[19].replace(['IA', 'KS','MN','MO','NE','ND','SD'], '3')
loan[19] = loan[19].replace(['DE', 'FL','GA','MD','NC','SC','DC','WV'], '4')
loan[19] = loan[19].replace(['AL', 'KY','MS','TN'], '5')
```

```
loan[19] = loan[19].replace(['AR', 'LA','OK','TX'], 6)
loan[19] = loan[19].replace(['AZ', 'CO','ID','MT','NV','NM','UT','WY'], 7)
loan[19] = loan[19].replace(['AK', 'CA','HI','OR','WA'], '8')

loan['state'] = pd.to_numeric(loan[19], errors='coerce').fillna(0).astype(np.int64)
loan['state'] = loan['state'].astype(float)
loan['state']=loan['state'].astype('category').cat.codes
```

In [249]:
```
#X21
loan[20] = loan[20].replace('',np.nan).astype(float)
loan[20] = loan[20].fillna((loan[20].mean()))
```

In [250]:
```
#X22
loan[21] = pd.to_numeric(loan[21], errors='coerce').fillna(0).astype(np.int64)
loan[21] = loan[21].replace(['12','13','14','15','16','17','18','19','20','21','22',
                             '25','26','27','28','29'], '11')
loan[21] = loan[21].astype(float)
```

In [251]:
```
#X23
loan['report'] = loan[22].astype(str).str[4:6]
loan['report'] = pd.to_numeric(loan['report'], errors='coerce').fillna(19).astype(np
loan['report'] = loan['report'].astype(str)
```

In [252]:
```
#X23
def year(row):
    if 0 <= int(row) <= 10:
        year = '200' + str(row)
    elif 10 < int(row) < 35:
        year = '20' + str(row)
    else:
        year = '19' + str(row)
    return year
loan['report'] = loan['report'].apply(year)
loan['report'] = loan['report'].astype(int)
```

In [253]:
```
loan['report']
```

Out[253]:
```
1          1994
2          2000
3          2000
4          1985
5          1996
6          1994
7          2000
8          1998
9          1993
10         2001
11         2006
12         1995
```

13	1997
14	2001
15	1990
16	2000
17	1996
18	1991
19	1998
20	2003
21	1985
23	1991
24	2000
25	2005
26	1994
27	2005
28	1975
29	2002
31	1997
32	2003
...	
399969	1993
399970	2002
399971	2007
399972	1995
399973	2000
399974	2005
399976	2003
399977	1994
399978	2011
399979	2002
399980	1990
399981	2001
399982	1970
399983	1999
399984	2008
399985	1988
399986	1991
399987	1995
399988	1999
399989	1994
399990	2002
399991	2007
399992	1983
399993	1987
399994	1991
399996	2003
399997	2007
399998	2004
399999	1994

```
400000    20010
Name: report, Length: 338990, dtype: int32
```

In [254]:
```
#X24
loan[23] = loan[23].fillna(0).astype(float)
```

In [255]:
```
#X27
loan[26] = loan[26].fillna(0).astype(float)
```

In [256]:
```
#X28, X29, and X30
loan[27] = loan[27].fillna(0).astype(float)
loan[28] = loan[28].astype(float)
loan[28] = loan[28].fillna(loan[28].mean()).astype(float)
loan[29] = loan[29].str.replace('%','')
loan[29] = loan[29].astype(float)
loan[29] = loan[29].fillna(loan[29].mean()).astype(float)
```

In [257]:
```
#X31
loan[30] = loan[30].astype(float)
loan[30] = loan[30].fillna(loan[30].mean()).astype(float)
```

In [258]:
```
#X32
loan[31] = loan[31].fillna('f')
loan[31] = loan[31].astype('category').cat.codes
```

In [259]:
```
loan = loan.drop(['ir'],axis=1)
loan[6]=loan['month']
loan[16]=loan['title']
loan[18]=loan['zip']
loan[19]=loan['state']
loan[22]=loan['report']
loan = loan.drop(['month','title','zip','state','report'],axis=1)
```

In [260]:
```
loan[0] = loan[0].str.replace('%','')
loan[0] = loan[0].astype(float)
# I still found one missing value in my big dataset, and I had checked for a long ti
# but wasn't able to find it. Dropping only one row would not affect much on my mode
loan = loan.dropna(axis=0)
loan
```

Out[260]:

	0	1	2	3	4	5	6	8	10	\
1	11.89	54734.0	80364.0	25000.0	25000.0	19080.0	0	8	0	
2	10.71	55742.0	114426.0	7000.0	7000.0	673.0	0	9	0	
3	16.99	57167.0	137225.0	25000.0	25000.0	24725.0	0	17	1	
4	13.11	57245.0	138150.0	1200.0	1200.0	1200.0	0	11	10	
5	13.57	57416.0	139635.0	10800.0	10800.0	10692.0	0	12	6	
6	19.05	58524.0	149512.0	7200.0	7200.0	7200.0	0	18	9	
7	10.08	58915.0	153417.0	7500.0	5025.0	557.0	0	7	3	
8	14.26	59006.0	154254.0	3000.0	3000.0	2988.0	0	14	3	

9	7.88	61390.0	182594.0	4000.0	4000.0	3900.0	0	4	0
10	14.96	61419.0	182917.0	5600.0	5600.0	5525.0	0	16	1
11	9.88	62102.0	191024.0	3200.0	3200.0	3200.0	0	5	5
12	11.14	65426.0	232106.0	4000.0	4000.0	3892.0	0	5	0
13	11.34	65640.0	234569.0	5000.0	2650.0	495.0	0	11	10
14	12.21	66431.0	243540.0	2525.0	2525.0	2375.0	0	9	3
15	13.47	66749.0	246329.0	10625.0	10625.0	5325.0	0	13	10
16	11.49	66943.0	247802.0	2800.0	2800.0	2700.0	1	8	0
17	13.24	66964.0	247990.0	7500.0	7500.0	986.0	0	17	3
18	8.59	67503.0	252415.0	10000.0	10000.0	10000.0	0	3	1
19	7.14	68163.0	258249.0	3000.0	3000.0	3000.0	0	2	9
20	8.63	68381.0	260179.0	6625.0	6625.0	6475.0	0	4	0
21	11.03	68817.0	264119.0	10000.0	10000.0	2475.0	0	10	10
23	8.94	69001.0	265533.0	15000.0	15000.0	14875.0	0	4	0
24	10.39	69124.0	266619.0	18000.0	18000.0	11636.0	0	8	10
25	15.13	69168.0	266943.0	5000.0	5000.0	4975.0	0	23	2
26	8.00	69251.0	267771.0	6000.0	6000.0	500.0	0	2	0
27	15.33	69550.0	270212.0	11200.0	11200.0	11188.0	1	17	2
28	8.63	69828.0	272798.0	15000.0	15000.0	13138.0	0	4	10
29	13.55	69924.0	274280.0	10000.0	10000.0	8790.0	0	18	3
31	12.29	70348.0	280311.0	9600.0	9600.0	9265.0	0	14	4
32	9.01	76597.0	76583.0	5000.0	5000.0	1775.0	0	6	1
...	...	...	...	...	...	...	..	..	..
399969	8.39	28752808.0	31225935.0	5275.0	5275.0	5275.0	0	4	0
399970	24.50	28752829.0	31225958.0	17500.0	17500.0	17500.0	1	27	10
399971	13.98	28752830.0	31225959.0	5000.0	5000.0	5000.0	0	12	7
399972	9.17	28752839.0	31225968.0	12500.0	12500.0	12500.0	0	5	5
399973	8.39	28752840.0	31225969.0	12000.0	12000.0	12000.0	0	4	0
399974	16.29	28752841.0	31225970.0	12800.0	12800.0	12800.0	1	16	3
399976	13.98	28752855.0	31225985.0	25900.0	25900.0	25900.0	0	12	0
399977	7.12	28752856.0	31225986.0	8000.0	8000.0	8000.0	0	2	10
399978	18.24	28752859.0	31225989.0	14375.0	14375.0	14350.0	0	19	4
399979	16.99	28752860.0	31225990.0	7500.0	7500.0	7500.0	0	0	0
399980	9.17	28752862.0	31225992.0	5000.0	5000.0	5000.0	0	5	3
399981	20.20	28752866.0	31225999.0	30000.0	30000.0	30000.0	1	0	10
399982	12.49	28752869.0	31226001.0	5000.0	5000.0	5000.0	0	9	0
399983	20.20	28752893.0	31226025.0	25000.0	25000.0	25000.0	1	22	0
399984	15.61	28752896.0	31226028.0	6000.0	6000.0	6000.0	0	0	2
399985	15.61	28752917.0	31226049.0	23325.0	23325.0	23325.0	0	15	10
399986	11.67	28752929.0	31226063.0	20000.0	20000.0	20000.0	1	0	5
399987	7.12	28752934.0	31226068.0	15000.0	15000.0	15000.0	0	0	10
399988	13.98	28752960.0	31226095.0	8000.0	8000.0	8000.0	0	12	2
399989	15.61	28752969.0	31226104.0	2000.0	2000.0	2000.0	0	15	4
399990	14.99	28752981.0	31226117.0	14100.0	14100.0	14100.0	0	14	0
399991	14.49	28752994.0	31226130.0	7500.0	7500.0	7500.0	0	0	1
399992	17.57	28753004.0	31226140.0	26000.0	26000.0	26000.0	1	18	10
399993	13.98	28753020.0	31226156.0	20000.0	20000.0	20000.0	0	12	3
399994	17.57	28753030.0	31226166.0	8000.0	8000.0	8000.0	0	18	0

399996	12.99	28753086.0	31226222.0	10000.0	10000.0	10000.0	1	10	8
399997	16.29	28753097.0	31226234.0	13150.0	13150.0	13150.0	0	16	1
399998	10.99	28753099.0	31226236.0	20000.0	20000.0	20000.0	1	7	1
399999	17.57	28753118.0	31226256.0	18475.0	18475.0	18475.0	1	18	10
400000	13.35	28753146.0	31226285.0	16000.0	16000.0	16000.0	0	11	4

	11	...	20	21	22	23	26	27	28	29	30	31
1	1	...	19.48	0.0	1994	0.0	10.0	0.0	28854.0	52.1	42.0	0
2	1	...	14.29	0.0	2000	0.0	7.0	0.0	33623.0	76.7	7.0	0
3	1	...	10.50	0.0	2000	0.0	10.0	0.0	19878.0	66.3	17.0	0
4	2	...	5.47	0.0	1985	0.0	5.0	0.0	2584.0	40.4	31.0	0
5	1	...	11.63	0.0	1996	1.0	14.0	0.0	3511.0	25.6	40.0	0
6	1	...	2.05	0.0	1994	0.0	6.0	0.0	3874.0	90.1	25.0	0
7	1	...	8.10	0.0	2000	1.0	3.0	0.0	33667.0	73.2	11.0	0
8	0	...	14.97	1.0	1998	0.0	13.0	0.0	4740.0	39.5	23.0	0
9	0	...	16.98	0.0	1993	0.0	11.0	0.0	50807.0	51.0	19.0	0
10	1	...	4.00	0.0	2001	0.0	5.0	1.0	3839.0	76.8	9.0	0
11	1	...	6.51	0.0	2006	0.0	7.0	0.0	3198.0	51.1	11.0	0
12	0	...	11.08	0.0	1995	0.0	14.0	0.0	24220.0	68.6	33.0	0
13	0	...	17.25	0.0	1997	1.0	20.0	0.0	69909.0	51.1	51.0	0
14	1	...	10.00	0.0	2001	1.0	5.0	0.0	5630.0	23.0	8.0	0
15	0	...	22.09	0.0	1990	2.0	5.0	1.0	13846.0	71.0	16.0	0
16	1	...	4.00	0.0	2000	0.0	7.0	1.0	2183.0	19.5	23.0	0
17	0	...	9.16	1.0	1996	2.0	11.0	0.0	5122.0	18.2	37.0	0
18	0	...	15.49	0.0	1991	0.0	7.0	0.0	6068.0	16.7	49.0	0
19	0	...	6.50	1.0	1998	0.0	14.0	0.0	3021.0	4.8	25.0	0
20	1	...	9.73	0.0	2003	0.0	3.0	0.0	6282.0	44.6	9.0	0
21	0	...	13.04	0.0	1985	1.0	7.0	0.0	5394.0	53.4	23.0	0
23	0	...	7.07	0.0	1991	1.0	6.0	0.0	7586.0	52.7	19.0	0
24	0	...	3.80	0.0	2000	0.0	2.0	0.0	8311.0	59.8	9.0	0
25	1	...	2.74	0.0	2005	3.0	2.0	0.0	591.0	84.4	6.0	0
26	0	...	16.08	0.0	1994	1.0	16.0	0.0	29797.0	23.2	39.0	0
27	1	...	17.19	0.0	2005	1.0	7.0	0.0	7248.0	69.2	11.0	0
28	2	...	2.59	0.0	1975	0.0	4.0	0.0	5656.0	27.6	25.0	0
29	1	...	7.94	0.0	2002	0.0	11.0	0.0	21162.0	57.7	14.0	0
31	1	...	8.80	0.0	1997	0.0	5.0	1.0	4822.0	58.1	27.0	0
32	0	...	10.00	2.0	2003	0.0	5.0	0.0	14354.0	36.6	7.0	0
...	..	...	...	...	...	...	...	...	...	...	...	..
399969	1	...	20.49	0.0	1993	0.0	10.0	1.0	3788.0	51.9	30.0	0
399970	0	...	20.06	0.0	2002	0.0	7.0	0.0	15275.0	88.3	18.0	1
399971	0	...	15.52	0.0	2007	2.0	9.0	0.0	13912.0	70.3	11.0	0
399972	2	...	17.15	1.0	1995	0.0	15.0	0.0	2080.0	13.3	36.0	1
399973	0	...	24.56	0.0	2000	2.0	10.0	0.0	96700.0	43.9	18.0	1
399974	0	...	21.45	1.0	2005	1.0	7.0	0.0	0.0	0.0	27.0	1
399976	1	...	17.16	0.0	2003	0.0	15.0	8.0	37646.0	46.2	38.0	0
399977	0	...	17.82	1.0	1994	0.0	16.0	0.0	7770.0	13.7	36.0	0
399978	0	...	29.63	0.0	2011	1.0	6.0	0.0	4158.0	66.0	9.0	0
399979	1	...	20.78	0.0	2002	1.0	4.0	0.0	5881.0	96.4	6.0	1

```
399980   1 ...   6.77  1.0   1990  0.0  10.0  0.0   6554.0   69.0  13.0   0
399981   0 ...   9.85  0.0   2001  1.0   6.0  0.0  27326.0   79.0  14.0   1
399982   1 ...  14.20  0.0   1970  0.0  10.0  1.0   9798.0   57.6  41.0   0
399983   0 ...  25.50  0.0   1999  0.0  16.0  0.0  49994.0  101.0  43.0   0
399984   1 ...  30.27  0.0   2008  1.0   9.0  0.0   8372.0   61.1   9.0   0
399985   0 ...  20.00  0.0   1988  0.0  16.0  0.0  26082.0   77.9  42.0   1
399986   0 ...  26.00  2.0   1991  0.0  17.0  0.0  12133.0   80.9  28.0   1
399987   0 ...  18.26  0.0   1995  0.0  24.0  0.0  45624.0   43.4  64.0   0
399988   1 ...  34.06  0.0   1999  3.0  13.0  1.0  11645.0   52.5  28.0   0
399989   0 ...   7.58  0.0   1994  1.0   7.0  0.0   6928.0   62.3  29.0   0
399990   1 ...  16.86  1.0   2002  0.0   9.0  1.0   7529.0   42.3  18.0   1
399991   0 ...  10.44  0.0   2007  2.0   8.0  0.0   5598.0   48.3  15.0   0
399992   0 ...  17.99  1.0   1983  3.0  15.0  1.0  27890.0   55.9  30.0   1
399993   2 ...   5.53  0.0   1987  2.0   8.0  2.0  13818.0   62.2  12.0   0
399994   2 ...  26.93  0.0   1991  2.0   7.0  0.0   1295.0   61.7  11.0   1
399996   1 ...  21.51  0.0   2003  0.0   9.0  0.0  10268.0   76.1  20.0   1
399997   2 ...  29.76  0.0   2007  0.0  11.0  0.0   8931.0   37.8  21.0   0
399998   0 ...  24.13  0.0   2004  0.0  14.0  0.0  28976.0   69.3  48.0   1
399999   2 ...  31.43  0.0   1994  0.0  19.0  0.0  11982.0   39.0  31.0   0
400000   0 ...  16.48  0.0  20010  0.0   9.0  0.0   3864.0   53.7  12.0   0

[338989 rows x 25 columns]
```

1.2 Data Pre-processing for test set

```
In [261]: dff = pd.read_csv("Holdout for Testing.csv", header = None)
          testloan = dff.drop([7, 9,14,15,17,24,25],axis=1)
          testloan
```

```
D:\Users\Owners\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2698: DtypeWarning
  interactivity=interactivity, compiler=compiler, result=result)
```

```
Out[261]:          0         1         2         3         4         5           6   8   \
          0       X1        X2        X3        X4        X5        X6          X7  X9
          1      NaN  44409194  47416907   $6,000    $6,000    $6,000   36 months  C5
          2      NaN  44017917  47034722  $24,000   $24,000   $24,000   36 months  A1
          3      NaN  44259158  47306871  $35,000   $35,000   $35,000   36 months  C2
          4      NaN  44429213  47476932  $10,000   $10,000   $10,000   60 months  D1
          5      NaN  44299188  47346901  $24,000   $24,000   $24,000   60 months  B1
          6      NaN  44057008  47073801  $18,000   $18,000   $18,000   60 months  B3
          7      NaN  41399898  44316683   $8,000    $8,000    $8,000   36 months  C3
          8      NaN  43801789  46818598  $24,000   $24,000   $24,000   36 months  B2
          9      NaN  44409120  47416833  $18,000   $18,000   $18,000   60 months  C4
          10     NaN  44319155  47376870  $25,000   $25,000   $25,000   36 months  A2
          11     NaN  44144338  47161355  $12,800   $12,800   $12,800   60 months  C1
          12     NaN  44106988  47123755  $13,000   $13,000   $13,000   36 months  A5
          13     NaN  44087155  47103930   $1,500    $1,500    $1,500   36 months  C2
```

14	NaN	43996610	47013371	$7,900	$7,900	$7,900	36 months	E1
15	NaN	44127043	47143823	$27,000	$27,000	$27,000	60 months	B2
16	NaN	44144047	47161046	$3,500	$3,500	$3,500	36 months	A2
17	NaN	44143754	47160744	$3,600	$3,600	$3,600	36 months	B1
18	NaN	44127050	47143830	$21,350	$21,350	$21,350	36 months	D4
19	NaN	44017855	47034659	$30,000	$30,000	$30,000	36 months	C1
20	NaN	44037602	47054406	$21,000	$21,000	$21,000	60 months	C4
21	NaN	44087130	47103904	$7,000	$7,000	$7,000	36 months	B4
22	NaN	43925814	46942579	$6,000	$6,000	$6,000	36 months	D3
23	NaN	43925762	46942527	$18,000	$18,000	$18,000	60 months	C3
24	NaN	44026670	47043446	$20,000	$20,000	$20,000	36 months	B4
25	NaN	44143958	47160953	$22,000	$22,000	$22,000	60 months	C2
26	NaN	43660027	46676772	$15,000	$15,000	$15,000	60 months	C5
27	NaN	44066999	47083782	$8,000	$8,000	$8,000	36 months	A1
28	NaN	43986704	47003463	$35,000	$35,000	$34,975	60 months	F1
29	NaN	44077993	47094813	$30,750	$30,750	$30,750	60 months	D5
...	...	...	...	...	...	...	...	..
79971	NaN	38212849	40996632	$23,000	$23,000	$23,000	60 months	B3
79972	NaN	38332529	41116303	$28,000	$28,000	$28,000	36 months	E2
79973	NaN	38362816	41146592	$24,000	$24,000	$24,000	36 months	B3
79974	NaN	38162640	40946430	$24,000	$24,000	$24,000	60 months	C3
79975	NaN	38212875	40996660	$15,000	$15,000	$15,000	36 months	B5
79976	NaN	38262179	41045939	$35,000	$35,000	$35,000	36 months	C3
79977	NaN	38332457	41116229	$11,375	$11,375	$11,375	36 months	D3
79978	NaN	38182410	40966185	$10,425	$10,425	$10,425	36 months	C4
79979	NaN	38342504	41126286	$35,000	$35,000	$35,000	60 months	E5
79980	NaN	38282579	41066360	$16,000	$16,000	$16,000	60 months	C3
79981	NaN	38312355	41096125	$15,000	$15,000	$14,950	60 months	B1
79982	NaN	38172531	40956302	$14,000	$14,000	$14,000	36 months	B5
79983	NaN	38332535	41116309	$7,475	$7,475	$7,475	36 months	C2
79984	NaN	38192869	40976643	$4,200	$4,200	$4,200	36 months	A2
79985	NaN	38282591	41066372	$2,100	$2,100	$2,100	36 months	C5
79986	NaN	38292500	41076265	$9,500	$9,500	$9,500	36 months	B1
79987	NaN	38212790	40996571	$24,000	$24,000	$24,000	36 months	D1
79988	NaN	38382517	41166303	$20,000	$20,000	$20,000	36 months	B4
79989	NaN	38212800	40996581	$30,000	$30,000	$29,975	60 months	D5
79990	NaN	38162576	40946363	$6,875	$6,875	$6,875	36 months	A1
79991	NaN	38232610	41016397	$19,200	$19,200	$19,200	60 months	E5
79992	NaN	38242654	41026414	$12,000	$12,000	$12,000	60 months	D2
79993	NaN	38192880	40976654	$15,000	$15,000	$15,000	36 months	C5
79994	NaN	38152386	40936153	$5,000	$5,000	$5,000	36 months	B5
79995	NaN	38142335	40926114	$18,000	$18,000	$17,950	60 months	B1
79996	NaN	38272852	41056632	$6,400	$6,400	$6,400	36 months	A2
79997	NaN	38232598	41016384	$30,000	$30,000	$30,000	60 months	E2
79998	NaN	38282597	41066378	$17,600	$17,600	$17,600	36 months	D4
79999	NaN	38232613	41016400	$2,500	$2,500	$2,500	36 months	C2
80000	NaN	38262186	41045946	$11,800	$11,800	$11,800	36 months	B5

	10	11	...	20	21	22	23	26	27	28	\
0	X11	X12	...	X21	X22	X23	X24	X27	X28	X29	
1	10+ years	MORTGAGE	...	28.31	0	2-Nov	1	18	0	19861	
2	8 years	RENT	...	16.03	0	Dec-68	1	12	0	17001	
3	10+ years	MORTGAGE	...	32.49	0	Oct-98	0	16	0	25797	
4	10+ years	RENT	...	32.96	0	Feb-99	1	13	1	9586	
5	10+ years	MORTGAGE	...	31.03	0	2-Dec	0	27	0	31842	
6	10+ years	MORTGAGE	...	34.33	0	Aug-98	0	8	0	18652	
7	8 years	RENT	...	14.2	0	6-Jun	0	7	0	13775	
8	< 1 year	RENT	...	16.32	0	1-Aug	1	10	0	20417	
9	7 years	MORTGAGE	...	17.72	0	2-Jan	0	9	0	3948	
10	10+ years	MORTGAGE	...	10.98	1	Dec-00	0	12	0	12689	
11	10+ years	MORTGAGE	...	38.39	0	Oct-91	0	18	0	50734	
12	2 years	MORTGAGE	...	20.44	0	Nov-81	0	7	0	8487	
13	5 years	RENT	...	16.26	0	5-Sep	1	6	1	6057	
14	1 year	RENT	...	22.13	0	4-Jul	0	9	0	25155	
15	10+ years	MORTGAGE	...	22.62	0	Mar-98	1	13	0	24083	
16	6 years	MORTGAGE	...	15.22	0	Jun-93	0	16	0	20122	
17	10+ years	MORTGAGE	...	6.49	0	Apr-90	0	6	0	1011	
18	9 years	MORTGAGE	...	31.79	0	3-Jun	1	20	0	26655	
19	5 years	RENT	...	14.35	1	Nov-76	0	12	0	9677	
20	10+ years	MORTGAGE	...	18.43	0	3-May	0	16	0	24386	
21	9 years	RENT	...	11.77	0	6-Mar	0	4	0	4494	
22	3 years	RENT	...	32.65	0	4-Mar	1	8	0	6605	
23	10+ years	OWN	...	31.71	0	4-Jul	1	12	1	15352	
24	10+ years	MORTGAGE	...	11.04	1	Jan-83	2	16	0	44866	
25	< 1 year	MORTGAGE	...	15.34	1	Jun-94	0	23	1	14553	
26	3 years	MORTGAGE	...	31.12	0	4-Sep	2	25	0	15596	
27	10+ years	MORTGAGE	...	14.22	0	Sep-84	0	13	0	18243	
28	10+ years	MORTGAGE	...	28.13	0	3-Apr	1	8	0	3360	
29	10+ years	MORTGAGE	...	25.87	0	May-83	0	14	0	19642	
...	...	...	...	...	...	...	...	...	...	...	
79971	1 year	RENT	...	18.37	0	1-Jul	0	18	2	17392	
79972	10+ years	OWN	...	18.36	0	1-Jun	2	17	1	15252	
79973	7 years	MORTGAGE	...	6.27	0	1-Dec	0	8	0	27888	
79974	10+ years	MORTGAGE	...	30.29	0	2-Oct	1	15	1	6363	
79975	10+ years	RENT	...	31.12	0	3-Jun	0	11	0	15378	
79976	< 1 year	RENT	...	15.64	0	Feb-00	0	2	0	0	
79977	< 1 year	MORTGAGE	...	22.26	0	1-Mar	0	10	1	8967	
79978	4 years	MORTGAGE	...	10.23	0	6-Aug	1	6	0	6057	
79979	< 1 year	MORTGAGE	...	27.2	1	May-89	0	22	0	145645	
79980	1 year	RENT	...	18.93	1	Feb-96	1	17	0	15748	
79981	4 years	OWN	...	31.35	0	Nov-98	0	29	0	40895	
79982	9 years	MORTGAGE	...	22.26	0	Jul-88	1	22	0	38226	
79983	3 years	RENT	...	5.97	0	1-Jun	0	3	0	2562	
79984	4 years	RENT	...	23.35	0	Jan-97	0	12	0	24586	
79985	5 years	RENT	...	17.09	0	Nov-90	1	12	0	37754	
79986	3 years	MORTGAGE	...	19.4	0	Oct-95	0	25	0	4188	

79987	6 years	MORTGAGE	...	12.97	0	2-Oct	1	12	0	57667
79988	6 years	RENT	...	27.74	0	4-Jul	1	9	0	14277
79989	10+ years	MORTGAGE	...	37.92	0	Mar-00	0	26	0	15153
79990	10+ years	RENT	...	10.89	0	Jan-78	0	24	0	3613
79991	6 years	OWN	...	12.44	1	1-Apr	1	11	0	9273
79992	5 years	MORTGAGE	...	30.26	0	1-Nov	0	25	0	12838
79993	n/a	RENT	...	30.48	0	4-Sep	1	12	1	8773
79994	3 years	RENT	...	11	0	3-Oct	0	4	1	5906
79995	10+ years	MORTGAGE	...	11.91	1	Jan-98	0	11	0	14171
79996	8 years	MORTGAGE	...	15.74	0	6-Apr	0	13	0	7185
79997	8 years	MORTGAGE	...	19.28	0	Aug-96	0	19	0	25151
79998	2 years	MORTGAGE	...	17.5	0	4-Dec	0	18	0	12161
79999	10+ years	MORTGAGE	...	16.52	0	Jun-99	0	13	0	13031
80000	4 years	MORTGAGE	...	9.6	0	5-Dec	0	8	0	13773

	29	30	31
0	X30	X31	X32
1	64.50%	33	f
2	26.20%	36	w
3	49.90%	33	w
4	43.80%	21	w
5	41.30%	43	w
6	64.80%	21	w
7	63.20%	10	w
8	68.70%	19	w
9	66.90%	23	w
10	41.30%	25	w
11	83.60%	31	w
12	83.20%	30	w
13	47%	12	w
14	99%	25	w
15	66.70%	32	w
16	42.60%	44	w
17	5.70%	25	w
18	57%	39	w
19	30.20%	36	w
20	81%	27	w
21	23.70%	23	f
22	70.30%	10	f
23	88.70%	22	w
24	72.70%	43	w
25	27.10%	39	w
26	28.90%	44	w
27	81.40%	19	w
28	33.30%	17	f
29	59.30%	46	w
...	...	...	...
79971	68.70%	21	w

```
79972  74.40%   33    f
79973  36.30%   30    f
79974  55.80%   31    f
79975     75%   32    f
79976     NaN   17    f
79977  77.30%   19    w
79978  31.40%    7    f
79979  64.70%   44    f
79980  67.60%   23    f
79981  24.20%   84    f
79982     53%   49    f
79983     61%    7    f
79984  27.80%   14    f
79985  97.60%   18    f
79986  48.70%   30    f
79987  67.50%   43    f
79988  82.10%   19    f
79989  32.30%   49    w
79990   6.90%   60    f
79991  54.50%   32    f
79992  26.70%   62    f
79993  35.20%   24    f
79994  86.90%    9    f
79995  71.20%   38    f
79996  47.90%   15    f
79997  51.30%   38    f
79998  37.10%   36    f
79999  76.70%   23    f
80000  93.70%   26    f

[80001 rows x 25 columns]
```

In [262]:
```
testloan['ir']=testloan[0]

testloan = testloan.drop([0])
testloan[1] = testloan[1].astype(float)
testloan[2] = testloan[2].astype(float)
testloan[3] = testloan[3].str.replace('$','')
testloan[3] = testloan[3].str.replace(',','')
testloan[3] = testloan[3].replace('',np.nan).astype(float)
testloan[3] = testloan[3].fillna((testloan[3].mean()))
testloan[4] = testloan[4].str.replace('$','')
testloan[4] = testloan[4].str.replace(',','')
testloan[4] = testloan[4].replace('',np.nan).astype(float)
testloan[4] = testloan[4].fillna((testloan[4].mean()))
testloan[5] = testloan[5].str.replace('$','')
testloan[5] = testloan[5].str.replace(',','')
testloan[5] = testloan[5].replace('',np.nan).astype(float)
```

```
testloan[5] = testloan[5].fillna((testloan[5].mean()))
#testloan[5] = testloan[5].astype('category').cat.codes
testloan['month'] = testloan[6].astype(str).str[0:3]
testloan['month'] = pd.to_numeric(testloan['month'], errors='coerce').fillna(36).asty
testloan['month'] = testloan['month'].astype(float)
testloan['month'] = testloan['month'].astype('category').cat.codes
testloan[8] = testloan[8].replace(['A1'], '0')
testloan[8] = testloan[8].replace(['A2'], '1')
testloan[8] = testloan[8].replace(['A3'], '2')
testloan[8] = testloan[8].replace(['A4'], '3')
testloan[8] = testloan[8].replace(['A5'], '4')

testloan[8] = testloan[8].replace(['B1'], '5')
testloan[8] = testloan[8].replace(['B2'], '6')
testloan[8] = testloan[8].replace(['B3'], '7')
testloan[8] = testloan[8].replace(['B4'], '8')
testloan[8] = testloan[8].replace(['B5'], '9')

testloan[8] = testloan[8].replace(['C1'], '10')
testloan[8] = testloan[8].replace(['C2'], '11')
testloan[8] = testloan[8].replace(['C3'], '12')
testloan[8] = testloan[8].replace(['C4'], '13')
testloan[8] = testloan[8].replace(['C5'], '14')

testloan[8] = testloan[8].replace(['D1'], '15')
testloan[8] = testloan[8].replace(['D2'], '16')
testloan[8] = testloan[8].replace(['D3'], '17')
testloan[8] = testloan[8].replace(['D4'], '18')
testloan[8] = testloan[8].replace(['D5'], '19')

testloan[8] = testloan[8].replace(['E1'], '20')
testloan[8] = testloan[8].replace(['E2'], '21')
testloan[8] = testloan[8].replace(['E3'], '22')
testloan[8] = testloan[8].replace(['E4'], '23')
testloan[8] = testloan[8].replace(['E5'], '24')

testloan[8] = testloan[8].replace(['F1'], '25')
testloan[8] = testloan[8].replace(['F2'], '26')
testloan[8] = testloan[8].replace(['F3'], '27')
testloan[8] = testloan[8].replace(['F4'], '28')
testloan[8] = testloan[8].replace(['F5'], '29')

testloan[8] = testloan[8].replace(['G1'], '30')
testloan[8] = testloan[8].replace(['G2'], '31')
testloan[8] = testloan[8].replace(['G3'], '32')
testloan[8] = testloan[8].replace(['G4'], '33')
testloan[8] = testloan[8].replace(['G5'], '34')
```

```
testloan[8] = pd.to_numeric(testloan[8], errors='coerce').fillna(0).astype(np.int64)
testloan[8] = testloan[8].astype(float)
testloan[8] = testloan[8].astype('category').cat.codes
testloan[10] = testloan[10].replace(['< 1 year'], '0')
testloan[10] = testloan[10].replace(['1 year'], '1')
testloan[10] = testloan[10].replace(['2 years'], '2')
testloan[10] = testloan[10].replace(['3 years'], '3')
testloan[10] = testloan[10].replace(['4 years'], '4')
testloan[10] = testloan[10].replace(['5 years'], '5')
testloan[10] = testloan[10].replace(['6 years'], '6')
testloan[10] = testloan[10].replace(['7 years'], '7')
testloan[10] = testloan[10].replace(['8 years'], '8')
testloan[10] = testloan[10].replace(['9 years'], '9')
testloan[10] = testloan[10].replace(['10+ years'], '10')
testloan[10] = testloan[10].replace(['n/a'], '0')

testloan[10] = pd.to_numeric(testloan[10], errors='coerce').fillna(0).astype(np.int64
testloan[10] = testloan[10].astype(float)
testloan[10] = testloan[10].astype('category').cat.codes
testloan[11] = testloan[11].replace(['MORTGAGE'], '0')
testloan[11] = testloan[11].replace(['RENT'], '1')
testloan[11] = testloan[11].replace(['OWN'], '2')
testloan[11] = testloan[11].replace(['OTHER'], '3')
testloan[11] = testloan[11].replace(['NONE'], '4')
testloan[11] = testloan[11].replace(['ANY'], '5')

testloan[11] = pd.to_numeric(testloan[11], errors='coerce').fillna(0).astype(np.int64
testloan[11] = testloan[11].astype(float)
testloan[11] = testloan[11].astype('category').cat.codes
testloan[12] = testloan[12].replace('',np.nan).astype(float)
testloan[12] = testloan[12].fillna((testloan[12].mean()))
testloan[12] = round(testloan[12],2)
testloan[13] = testloan[13].replace(['VERIFIED - income'], '0')
testloan[13] = testloan[13].replace(['not verified'], '1')
testloan[13] = testloan[13].replace(['VERIFIED - income source'], '2')

testloan[13] = pd.to_numeric(testloan[13], errors='coerce').fillna(0).astype(np.int64
testloan[13] = testloan[13].astype(float)
testloan[13] = testloan[13].astype('category').cat.codes
testloan[16] = testloan[16].str.replace('debt_consolidation','0')
testloan[16] = testloan[16].str.replace('credit_card','1')
testloan[16] = testloan[16].str.replace('car','2')
testloan[16] = testloan[16].str.replace('educational','3')
testloan[16] = testloan[16].str.replace('home_improvement','4')
testloan[16] = testloan[16].str.replace('house','5')
testloan[16] = testloan[16].str.replace('major_purchase','6')
testloan[16] = testloan[16].str.replace('medical','7')
testloan[16] = testloan[16].str.replace('moving','8')
```

```
testloan[16] = testloan[16].str.replace('other','9')
testloan[16] = testloan[16].str.replace('renewable_energy','10')
testloan[16] = testloan[16].str.replace('small_business','11')
testloan[16] = testloan[16].str.replace('vacation','12')
testloan[16] = testloan[16].str.replace('wedding','13')
testloan['title'] = pd.to_numeric(testloan[16], errors='coerce').fillna(0).astype(np
testloan['title'] = testloan['title'].astype(float)
testloan['title'] = testloan['title'].astype('category').cat.codes
testloan['zip'] = testloan[18].astype(str).str[0]

testloan['zip'] = pd.to_numeric(testloan['zip'], errors='coerce').fillna(0).astype(n
testloan['zip'] = testloan['zip'].astype(float)
testloan['zip']= testloan['zip'].astype('category').cat.codes
testloan[19] = testloan[19].replace(['CT','ME','MA','NH','RI','VT'], '0')
testloan[19] = testloan[19].replace(['NJ', 'NY','PA'], '1')
testloan[19] = testloan[19].replace(['IL', 'IN','MI','OH','WI'], '2')
testloan[19] =  testloan[19].replace(['IA', 'KS','MN','MO','NE','ND','SD'], '3')
testloan[19] = testloan[19].replace(['DE', 'FL','GA','MD','NC','SC','DC','WV'], '4')
testloan[19] = testloan[19].replace(['AL', 'KY','MS','TN'], '5')
testloan[19] = testloan[19].replace(['AR', 'LA','OK','TX'], 6)
testloan[19] = testloan[19].replace(['AZ', 'CO','ID','MT','NV','NM','UT','WY'], 7)
testloan[19] = testloan[19].replace(['AK', 'CA','HI','OR','WA'], '8')

testloan['state'] = pd.to_numeric(testloan[19], errors='coerce').fillna(0).astype(np
testloan['state'] = testloan['state'].astype(float)
testloan['state']= testloan['state'].astype('category').cat.codes
testloan[20] = testloan[20].replace('',np.nan).astype(float)
testloan[20] = testloan[20].fillna((testloan[20].mean()))
testloan[21] = pd.to_numeric(testloan[21], errors='coerce').fillna(0).astype(np.int64
testloan[21] = testloan[21].replace(['12','13','14','15','16','17','18','19','20','2
                                     '25','26','27','28','29'], '11')
testloan[21] = testloan[21].astype(float)
testloan['report'] = testloan[22].astype(str).str[4:6]
testloan['report'] = pd.to_numeric(testloan['report'], errors='coerce').fillna(19).as
testloan['report'] = testloan['report'].astype(str)
def year(row):
    if 0 <= int(row) <= 10:
        year = '200' + str(row)
    elif 10 < int(row) < 35:
        year = '20' + str(row)
    else:
        year = '19' + str(row)
    return year
testloan['report'] = testloan['report'].apply(year)
testloan['report'] = testloan['report'].astype(int)
testloan['report']
testloan[23] = testloan[23].fillna(0).astype(float)
testloan[26] = testloan[26].fillna(0).astype(float)
```

```
testloan[27] = testloan[27].fillna(0).astype(float)
testloan[28] = testloan[28].astype(float)
testloan[28] = testloan[28].fillna(testloan[28].mean()).astype(float)
testloan[29] = testloan[29].str.replace('%','')
testloan[29] = testloan[29].astype(float)
testloan[29] = testloan[29].fillna(testloan[29].mean()).astype(float)
testloan[30] = testloan[30].astype(float)
testloan[30] = testloan[30].fillna(testloan[30].mean()).astype(float)
testloan[31] = testloan[31].fillna('f')
testloan[31] = testloan[31].astype('category').cat.codes
testloan = testloan.drop(['ir'],axis=1)
testloan[6]= testloan['month']
testloan[16]= testloan['title']
testloan[18]= testloan['zip']
testloan[19]= testloan['state']
testloan[22]= testloan['report']
testloan = testloan.drop(['month','title','zip','state','report'],axis=1)
testloan[0] = testloan[0].str.replace('%','')
testloan[0] = testloan[0].astype(float)
testloan.columns = ['X1','X2','X3','X4','X5','X6','X7','X9','11','X12','X13','X14','X
                    'X19','X20','X21','X22','X23','X24','X27','X28','X29','X30','X31
```

```
In [263]: testloan[['X2','X3','X4','X5','X6','X7','X9','11','X12','X13','X14','X17',
                    'X19','X20','X21','X22','X23','X24','X27','X28','X29','X30','X31','X32']] =
                    'X3','X4','X5','X6','X7','X9','11','X12','X13','X14','X17',
                    'X19','X20','X21','X22','X23','X24','X27','X28','X29','X30','X31','X32']].
```

```
In [279]: #testloan['X1']
```

1.3 Modeling

1.3.1 Decision Tree

```
In [265]: y=loan.iloc[:,0]
          x=loan.iloc[:,1:31]
```

```
In [266]: import matplotlib.pyplot as plt
          from sklearn.metrics import classification_report
          from sklearn.model_selection import cross_val_score

          import sklearn.model_selection as cv
          from sklearn.tree import DecisionTreeRegressor

          (x_train, x_test, y_train, y_test) = cv.train_test_split(x, y, test_size=.20)
          # Instantiate dt
          dt = DecisionTreeRegressor(max_depth=4,
                                     min_samples_leaf=0.11,
                                     random_state=3)
```

```
# Fit dt to the training set
dt.fit(x_train, y_train)
```

```
Out[266]: DecisionTreeRegressor(criterion='mse', max_depth=4, max_features=None,
                                max_leaf_nodes=None, min_impurity_decrease=0.0,
                                min_impurity_split=None, min_samples_leaf=0.11,
                                min_samples_split=2, min_weight_fraction_leaf=0.0,
                                presort=False, random_state=3, splitter='best')
```

```
In [267]: from sklearn.metrics import mean_squared_error as MSE

          # Compute y_pred
          y_pred_dt = dt.predict(x_test)

          # Compute mse_dt
          mse_dt = MSE(y_test, y_pred_dt)

          # Compute rmse_dt
          rmse_dt = mse_dt**(1/2)

          # Print rmse_dt
          print("Test set RMSE of dt: {:.2f}".format(rmse_dt))
```

```
Test set RMSE of dt: 2.34
```

```
In [268]: from sklearn.model_selection import cross_val_score
          # Compute the array containing the 10-folds CV MSEs
          MSE_CV_scores = - cross_val_score(dt, x_train, y_train, cv=10,
                                            scoring='neg_mean_squared_error',
                                            n_jobs=-1)

          # Compute the 10-folds CV RMSE
          RMSE_CV = (MSE_CV_scores.mean())**(1/2)

          # Print RMSE_CV
          print('CV RMSE: {:.2f}'.format(RMSE_CV))
```

```
CV RMSE: 2.35
```

```
In [269]: # Import mean_squared_error from sklearn.metrics as MSE
          from sklearn.metrics import mean_squared_error as MSE

          # Fit dt to the training set
          dt.fit(x_train, y_train)

          # Predict the labels of the training set
          y_pred_train = dt.predict(x_train)
```

```
# Evaluate the training set RMSE of dt
RMSE_train = (MSE(y_train, y_pred_train))**(1/2)

# Print RMSE_train
print('Train RMSE: {:.2f}'.format(RMSE_train))
```

```
Train RMSE: 2.35
```

In [270]:
```
# Import train_test_split from sklearn.model_selection
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
# Set SEED for reproducibility
SEED = 1

# Split the data into 70% train and 30% test
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state

# Instantiate a DecisionTreeRegressor dt
dt = DecisionTreeRegressor(max_depth=4, min_samples_leaf=0.26, random_state=SEED)


# Compute the array containing the 10-folds CV MSEs
MSE_CV_scores = - cross_val_score(dt, X_train, y_train, cv=10,
                                  scoring='neg_mean_squared_error',
                                  n_jobs=-1)

# Compute the 10-folds CV RMSE
RMSE_CV = (MSE_CV_scores.mean())**(1/2)

# Print RMSE_CV
print('CV RMSE: {:.2f}'.format(RMSE_CV))
```

```
CV RMSE: 3.02
```

In [271]:
```
# Import mean_squared_error from sklearn.metrics as MSE
from sklearn.metrics import mean_squared_error as MSE

# Fit dt to the training set
dt.fit(X_train, y_train)

# Predict the labels of the training set
y_pred_train = dt.predict(X_train)

# Evaluate the training set RMSE of dt
RMSE_train = (MSE(y_train, y_pred_train))**(1/2)
```

```
        # Print RMSE_train
        print('Train RMSE: {:.2f}'.format(RMSE_train))
```

```
Train RMSE: 3.02
```

In [272]:
```
        # Import train_test_split from sklearn.model_selection
        from sklearn.model_selection import train_test_split

        # Set SEED for reproducibility
        SEED = 1

        # Split the data into 70% train and 30% test
        X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state

        # Instantiate a DecisionTreeRegressor dt
        dt = DecisionTreeRegressor(max_depth= 6 , min_samples_leaf= 0.15 , random_state=SEED)
```

In [273]:
```
        from sklearn.metrics import accuracy_score
        from sklearn.model_selection import train_test_split

        from sklearn import linear_model
        from sklearn.linear_model import LogisticRegression
        from sklearn.neighbors import KNeighborsRegressor
        from sklearn.naive_bayes import GaussianNB
        from sklearn.svm import SVC, LinearSVC
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.ensemble import AdaBoostRegressor
        from sklearn.ensemble import GradientBoostingRegressor
        from sklearn.ensemble import RandomForestRegressor
        from sklearn.linear_model import SGDRegressor
        from sklearn.linear_model import Perceptron

        from sklearn.metrics import mean_squared_error as MSE
```

In [274]:
```
        from sklearn import preprocessing
        from sklearn import utils

        lab_enc = preprocessing.LabelEncoder()
        encoded = lab_enc.fit_transform(y_train)


        print(utils.multiclass.type_of_target(y_train))


        print(utils.multiclass.type_of_target(y_train.astype('int')))


        print(utils.multiclass.type_of_target(encoded))
```

```
continuous
multiclass
multiclass
```

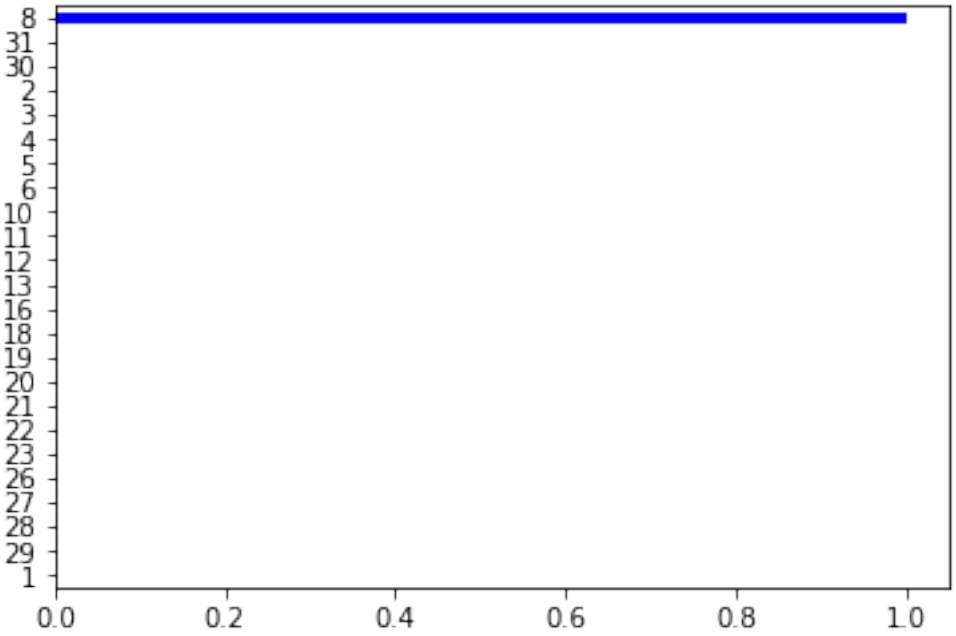
```
In [277]: y=loan.iloc[:,0]
          X=loan.iloc[:,1:31]
          X_train, X_test, y_train, y_test = \
          train_test_split(X, y,
          test_size=0.2,
          random_state=SEED)
```

1.3.2 Random Forest

```
In [278]: from sklearn.ensemble import RandomForestRegressor
          rf = RandomForestRegressor(n_estimators=400,
          min_samples_leaf=0.12,
          random_state=SEED)
          # Fit 'rf' to the training set
          rf.fit(X_train, y_train)
          # Predict the test set labels 'y_pred'
          y_pred = rf.predict(X_test)
          y_pred_train=rf.predict(X_train)
          # Evaluate the test set RMSE
          rmse_test_rf = MSE(y_test, y_pred)**(1/2)
          rmse_train_rf = MSE(y_train, y_pred_train)**(1/2)
          # Print the test set RMSE
          print('Test set RMSE of rf: {:.2f}'.format(rmse_test_rf))
          print('Train set RMSE of rf: {:.2f}'.format(rmse_train_rf))
```

```
Test set RMSE of rf: 2.85
Train set RMSE of rf: 2.86
```

```
In [280]: import pandas as pd
          import matplotlib.pyplot as plt
          # Create a pd.Series of features importances
          importances_rf = pd.Series(rf.feature_importances_,
          index = X.columns)
          # Sort importances_rf
          sorted_importances_rf = importances_rf.sort_values()
          # Make a horizontal bar plot
          sorted_importances_rf.plot(kind='barh', color='blue')
          plt.show()
```



According to the Decision Tree model, I found that the 80% and 20% training and testing set division way gave a Lower RMSE than 70% and 30% training and testing set division. The RMSE for "80% 20%" was about 2.35, but the "70% 30% division" gave an RMSE of 3.02. I will use 80% and 20% training and testing set division way.

1.3.3 Try more models: Gradient Boost

```
In [281]: from sklearn.ensemble import GradientBoostingRegressor
          from sklearn.metrics import mean_squared_error
          from sklearn import ensemble
          params = {'n_estimators': 500, 'max_depth': 4, 'min_samples_split': 2,
                    'learning_rate': 0.01, 'loss': 'ls'}
          gbr = ensemble.GradientBoostingRegressor(**params)

          gbr.fit(X_train, y_train)
          y_pred = gbr.predict(X_test)
          y_pred_train=gbr.predict(X_train)

          mse = mean_squared_error(y_test, gbr.predict(X_test))
          print("MSE: %.4f" % mse)

          rmse_test_gbr = MSE(y_test, y_pred)**(1/2)
          rmse_train_gbr = MSE(y_train, y_pred_train)**(1/2)
          print('Test set RMSE of gradient boosting: {:.2f}'.format(rmse_test_gbr))
          print('Train set RMSE of gradient boosting: {:.2f}'.format(rmse_train_gbr))
```

```
MSE: 1.9739
```

```
Test set RMSE of gradient boosting: 1.40
Train set RMSE of gradient boosting: 1.40
```

1.3.4 Summary for RMSE

```
In [287]: print('RMSE Comparison:', '\n',
          'Decision Tree:', round(rmse_dt,2), '\n',
          'Random Forest:', round(rmse_test_rf,2), '\n',
          'Gradient boost:', round(rmse_test_gbr, 2))
```

```
RMSE Comparison:
 Decision Tree: 2.34
 Random Forest: 2.85
 Gradient boost: 1.4
```

If the CV RMSE is greater than training RMSE, my model will be overfitting, If CV error almost equal to my training error but they both greater than desired error, my model will be underfitting. In my model, both my CV and train set RMSE are 2.35, which means my model is good. Compared with the RMSE regard to each model for the testing set, the Gradient Boost gave me the lowest RMSE. However, I still want to see how well random forest works. I will predict my results for the holdout_test data set with both Gradient Boost and Random Forest. The Decision Tree Classifier located in the middle. However, I don't want to see the prediction results by decision tree since I already have two final result datasets.

1.3.5 Predictions for both Gradient Boost and Random Forest

Gradient Boost First

```
In [288]: testloan = testloan.loc[:,testloan.columns != 'X1']
          pred_test_gbr = gbr.predict(testloan)

          dff3 = pd.read_csv("Holdout for Testing.csv", header = None)
          dff3 = dff3.drop([0])
          dff3[0] = pred_test_gbr
          dff3[0] = dff3[0].astype(str)+'%'
          dff3.columns = ['X1','X2','X3','X4','X5','X6','X7','X8','X9','X10','11','X12','X13',
                          'X17','X18','X19','X20','X21','X22','X23','X24','X25','X26','X27','X
                          'X30','X31','X32']
```

```
D:\Users\Owners\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2698: DtypeWarning
  interactivity=interactivity, compiler=compiler, result=result)
```

```
In [289]: dff3.to_csv('C:/Users/Owners/Desktop/GadientBoost_holdout_testing_loan.csv')
```

Then Random Forest

```
In [290]: testloan = testloan.loc[:,testloan.columns != 'X1']
          pred_test_rf = rf.predict(testloan)

          dff2 = pd.read_csv("Holdout for Testing.csv", header = None)
          dff2 = dff2.drop([0])
          dff2[0] = pred_test_rf
          dff2[0] = dff2[0].astype(str)+'%'
          dff2.columns = ['X1','X2','X3','X4','X5','X6','X7','X8','X9','X10','11','X12','X13',
                          'X17','X18','X19','X20','X21','X22','X23','X24','X25','X26','X27','X
                          'X30','X31','X32']
```

```
D:\Users\Owners\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py:2698: DtypeWarning
  interactivity=interactivity, compiler=compiler, result=result)
```

```
In [291]: dff2.to_csv('C:/Users/Owners/Desktop/RandomForest_holdout_testing_loan.csv')
```
