In [3]:

```
import pandas as pd
import numpy as np
import seaborn as sns
```

In [4]:

```
df=pd.read_csv('datasets/loan_data.csv')
df
```

Out[4]:

	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with
0	1	debt_consolidation	0.1189	829.10	11.350407	19.48	737	5639.
1	1	credit_card	0.1071	228.22	11.082143	14.29	707	2760.
2	1	debt_consolidation	0.1357	366.86	10.373491	11.63	682	4710.
3	1	debt_consolidation	0.1008	162.34	11.350407	8.10	712	2699.
4	1	credit_card	0.1426	102.92	11.299732	14.97	667	4066.
...	...	...	...	...	...	...	...	
9573	0	all_other	0.1461	344.76	12.180755	10.39	672	10474.
9574	0	all_other	0.1253	257.70	11.141862	0.21	722	4380.
9575	0	debt_consolidation	0.1071	97.81	10.596635	13.09	687	3450.
9576	0	home_improvement	0.1600	351.58	10.819778	19.18	692	1800.
9577	0	debt_consolidation	0.1392	853.43	11.264464	16.28	732	4740.

9578 rows × 14 columns

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   credit.policy      9578 non-null   int64  
 1   purpose            9578 non-null   object 
 2   int.rate           9578 non-null   float64
 3   installment        9578 non-null   float64
 4   log.annual.inc     9578 non-null   float64
 5   dti                9578 non-null   float64
 6   fico               9578 non-null   int64  
 7   days.with.cr.line  9578 non-null   float64
 8   revol.bal          9578 non-null   int64  
 9   revol.util         9578 non-null   float64
 10  inq.last.6mths     9578 non-null   int64  
 11  delinq.2yrs        9578 non-null   int64  
 12  pub.rec            9578 non-null   int64  
 13  not.fully.paid     9578 non-null   int64  
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

work with missing data

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
credit.policy        0
purpose              0
int.rate             0
installment          0
log.annual.inc       0
dti                  0
fico                 0
days.with.cr.line    0
revol.bal            0
revol.util           0
inq.last.6mths       0
delinq.2yrs          0
pub.rec              0
not.fully.paid       0
dtype: int64
```

categorical data

In [7]:

```
df=pd.get_dummies(df,columns=['purpose'],drop_first=True)
df
```

Out[7]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal
0	1	0.1189	829.10	11.350407	19.48	737	5639.958333	28854
1	1	0.1071	228.22	11.082143	14.29	707	2760.000000	33623
2	1	0.1357	366.86	10.373491	11.63	682	4710.000000	3511
3	1	0.1008	162.34	11.350407	8.10	712	2699.958333	33667
4	1	0.1426	102.92	11.299732	14.97	667	4066.000000	4740
...	...	...	...	...	...	...	...	...
9573	0	0.1461	344.76	12.180755	10.39	672	10474.000000	215372
9574	0	0.1253	257.70	11.141862	0.21	722	4380.000000	184
9575	0	0.1071	97.81	10.596635	13.09	687	3450.041667	10036
9576	0	0.1600	351.58	10.819778	19.18	692	1800.000000	0
9577	0	0.1392	853.43	11.264464	16.28	732	4740.000000	37879

9578 rows × 19 columns

In [8]:

```
df.describe()
```

Out[8]:

fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs	pub.rec
78.000000	9578.000000	9.578000e+03	9578.000000	9578.000000	9578.000000	9578.000000
10.846314	4560.767197	1.691396e+04	46.799236	1.577469	0.163708	0.062122
37.970537	2496.930377	3.375619e+04	29.014417	2.200245	0.546215	0.262126
12.000000	178.958333	0.000000e+00	0.000000	0.000000	0.000000	0.000000
82.000000	2820.000000	3.187000e+03	22.600000	0.000000	0.000000	0.000000
07.000000	4139.958333	8.596000e+03	46.300000	1.000000	0.000000	0.000000
37.000000	5730.000000	1.824950e+04	70.900000	2.000000	0.000000	0.000000
27.000000	17639.958330	1.207359e+06	119.000000	33.000000	13.000000	5.000000

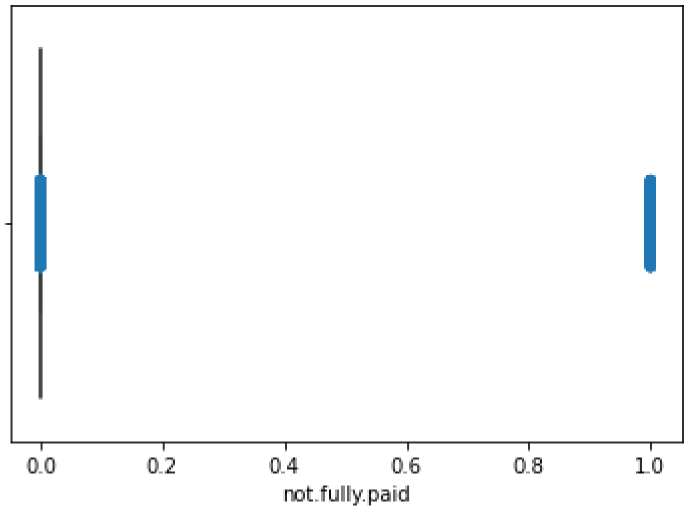
outliers

In [9]:

```
sns.boxplot(x='not.fully.paid',data=df)
sns.stripplot(x='not.fully.paid',data=df)
```

Out[9]:

```
<AxesSubplot:xlabel='not.fully.paid'>
```



In [10]:

```
from datasist.structdata import detect_outliers
outlier_indices=detect_outliers(df,0,['not.fully.paid'])
outlier_indices
```

```
 142,
 143,
 145,
 150,
 160,
 161,
 164,
 165,
 182,
 187,
 193,
 201,
 204,
 205,
 207,
 211,
 218,
 222,
 226,
 233,
```

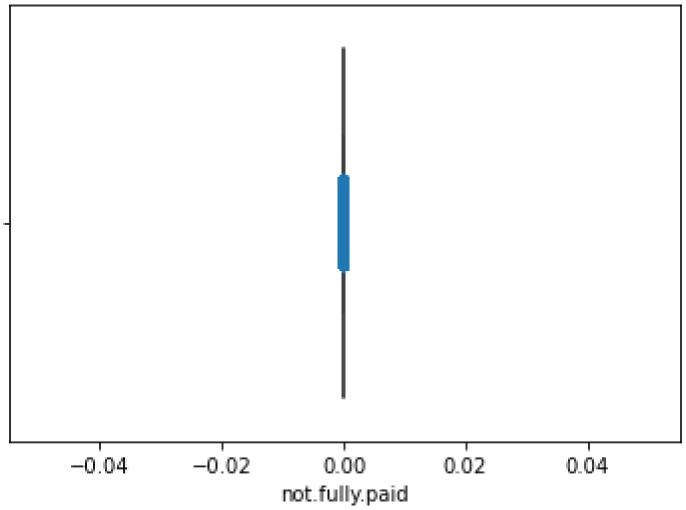
In [11]:

```
df.drop(outlier_indices,inplace=True)

sns.boxplot(x='not.fully.paid',data=df)
sns.stripplot(x='not.fully.paid',data=df)
```

Out[11]:

```
<AxesSubplot:xlabel='not.fully.paid'>
```



In [13]:

```
x=df.drop('credit.policy',axis=1)
x
```

Out[13]:

	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	de
0	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.1	0	
1	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.7	0	
2	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.6	1	
3	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.2	1	
4	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.5	0	
...	...	...	...	...	...	...	...	...	...	
9561	0.0788	115.74	10.999095	10.17	722	4410.000000	11586	61.6	4	
9562	0.1348	508.87	10.933107	17.76	717	3870.041667	8760	28.2	6	
9564	0.1385	511.56	12.323856	12.33	687	6420.041667	385489	51.2	4	
9567	0.1311	101.24	10.968198	8.23	687	2790.041667	1514	13.8	5	

In [14]:

```
y=df['credit.policy']
y
```

Out[14]:

```
0       1
1       1
2       1
3       1
4       1
       ..
9561    0
9562    0
9564    0
9567    0
9568    0
Name: credit.policy, Length: 8045, dtype: int64
```

In [15]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
x_train
```

Out[15]:

	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.l
7504	0.1422	342.85	10.714418	12.48	682	7050.041667	4719	77.4	
1176	0.0832	157.43	10.335010	10.13	742	4307.000000	20337	39.2	
9059	0.1505	180.40	10.126471	5.95	692	2279.958333	225	5.0	
1444	0.1051	243.81	10.858922	7.82	722	4379.958333	5911	20.9	
2574	0.0932	351.42	11.095398	19.15	747	3749.958333	4148	14.3	
...	...	...	...	...	...	...	...	...	
473	0.1324	405.72	11.314475	15.12	672	5760.041667	27905	81.8	
2909	0.0768	124.77	10.499573	0.93	742	1919.958333	1410	67.1	
8813	0.1507	52.05	9.035987	10.00	652	1019.958333	1764	53.5	
6806	0.1253	368.13	10.645425	18.89	702	3060.041667	6811	25.7	
6884	0.1357	244.58	11.350407	6.45	692	12061.000000	7983	43.6	

6436 rows × 18 columns

In [16]:

```
x_test
```

Out[16]:

	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.l
6094	0.1114	518.30	11.512925	21.88	747	4620.000000	29860	47.6	
146	0.0964	90.68	11.156251	18.00	732	3691.000000	55720	10.0	
8078	0.1122	472.94	11.407565	15.00	687	1950.041667	11220	72.4	
5081	0.0894	317.72	11.407565	10.47	767	9630.000000	93093	1.4	
1001	0.1229	166.77	10.471638	14.96	677	4829.958333	18099	69.2	
...	...	...	...	...	...	...	...	...	
4925	0.1635	883.23	11.884461	14.32	682	4619.958333	30427	86.9	
2236	0.1347	508.84	11.842287	6.76	707	11550.041670	0	0.0	
4829	0.1357	101.91	10.596635	22.98	672	4500.000000	17590	80.7	
1056	0.1292	168.28	10.596535	16.29	667	2729.958333	14244	85.8	
2931	0.1284	33.62	10.085809	24.35	687	3480.000000	2533	38.4	

1609 rows × 18 columns

In [18]:

```
y_test.value_counts()
```

Out[18]:

```
1    1342
0     267
Name: credit.policy, dtype: int64
```

In [19]:

```
from imblearn.over_sampling import SMOTE
sampler=SMOTE()
x_train,y_train=sampler.fit_resample(x_train,y_train)
y_train.value_counts()
```

Out[19]:

```
1    5354
0    5354
Name: credit.policy, dtype: int64
```

In [25]:

```
x=df.drop('installment',axis=1)
y=df['installment']
```

In [26]:

```
y.value_counts()
```

Out[26]:

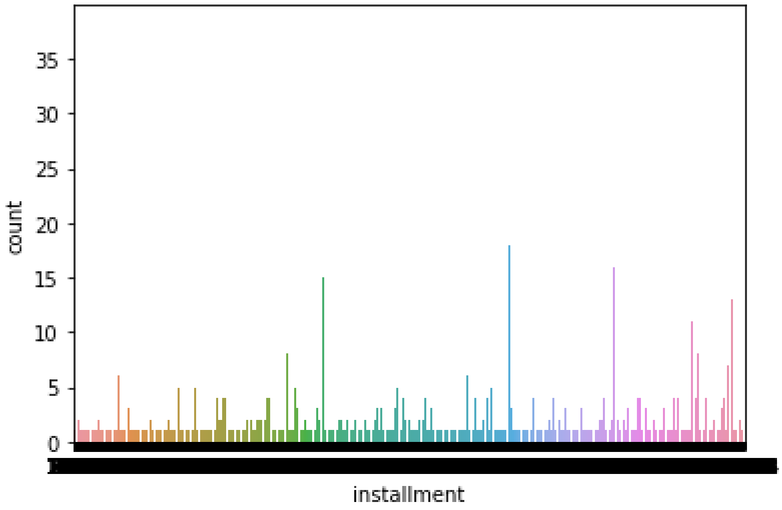
```
317.72    38
316.11    33
319.47    29
381.26    27
156.10    24
          ..
680.69     1
127.82     1
759.31     1
241.55     1
35.83      1
Name: installment, Length: 4111, dtype: int64
```

In [27]:

```
sns.countplot(x='installment',data=df)
```

Out[27]:

```
<AxesSubplot:xlabel='installment', ylabel='count'>
```



In [28]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
y_train.value_counts()
```

Out[28]:

```
317.72    30
316.11    29
319.47    23
381.26    22
156.10    20
          ..
377.25     1
252.85     1
477.92     1
827.41     1
71.00      1
Name: installment, Length: 3507, dtype: int64
```

In [30]:

```
df
```

Out[30]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal
0	1	0.1189	829.10	11.350407	19.48	737	5639.958333	28854
1	1	0.1071	228.22	11.082143	14.29	707	2760.000000	33623
2	1	0.1357	366.86	10.373491	11.63	682	4710.000000	3511
3	1	0.1008	162.34	11.350407	8.10	712	2699.958333	33667
4	1	0.1426	102.92	11.299732	14.97	667	4066.000000	4740
...	...	...	...	...	...	...	...	...
9561	0	0.0788	115.74	10.999095	10.17	722	4410.000000	11586
9562	0	0.1348	508.87	10.933107	17.76	717	3870.041667	8760
9564	0	0.1385	511.56	12.323856	12.33	687	6420.041667	385489
9567	0	0.1311	101.24	10.968198	8.23	687	2790.041667	1514
9568	0	0.1979	37.06	10.645425	22.17	667	5916.000000	28854

8045 rows × 19 columns

In [31]:

```
x=df[['credit.policy','not.fully.paid','purpose_credit_card']]
y=df['installment']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
x_train
```

Out[31]:

	credit.policy	not.fully.paid	purpose_credit_card
3565	1	0	0
9441	0	0	0
731	1	0	0
9101	0	0	0
2638	1	0	0
...	...	...	...
5885	1	0	0
4917	1	0	0
1032	1	0	0
1918	1	0	0
5537	1	0	0

6033 rows × 3 columns

In [32]:

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
x_train
```

Out[32]:

```
array([[1., 0., 0.],
       [0., 0., 0.],
       [1., 0., 0.],
       ...,
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.]])
```

In [33]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
x_train
```

Out[33]:

	credit.policy	not.fully.paid	purpose_credit_card
3360	1	0	0
749	1	0	0
2612	1	0	0
6284	1	0	0
6315	1	0	0
...	...	...	...
5464	1	0	0
4515	1	0	0
8852	0	0	0
4239	1	0	0
5396	1	0	0

6033 rows × 3 columns

In [34]:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.transform(x_test)
x_train
```

Out[34]:

```
array([[ 0.44333846,  0.        , -0.39800771],
       [ 0.44333846,  0.        , -0.39800771],
       [ 0.44333846,  0.        , -0.39800771],
       ...,
       [-2.25561302,  0.        , -0.39800771],
       [ 0.44333846,  0.        , -0.39800771],
       [ 0.44333846,  0.        , -0.39800771]])
```

In [ ]:

```

```