Data Visualization and Cleaning Assignment

Created by: Venkata Reddy Konasani

statinfer.com (https://statinfer.com/)

The end goal is to build a credit risk model that predicts the risky customers. In this assignment we are going to disuss data visualizations and cleaning process. Read the column names and descriptions before you start this assignment. Download the data from this below link

https://drive.google.com/drive/folders/1ISpTlfUik4enxgDUkqPzNrDBevN9vuQN?usp=sharing (https://drive.google.com/drive/folders/1ISpTlfUik4enxgDUkqPzNrDBevN9vuQN?usp=sharing)

Note

Some questions may not be relvant for some variables. You can igonre questions that are not applicable.

1.Data Importing and basic Details

Import loans dataset. Print some sample rows and basic information. Read the data dictionary and understand the variables before starting this assignment

In [1]:

```
import pandas as pd
loans=pd.read_csv("/content/drive/My Drive/Training/ML_Full_Semester/Assignments/Data_v
iz_Cleaning_Give_me_Credit/Datasets/Loans/cs-training.csv")
```

In [2]:

```
print(loans.shape)
print(loans.columns)
```

```
(150000, 12)
Index(['Sr_No', 'SeriousDlqin2yrs', 'RevolvingUtilizationOfUnsecuredLine
s',
       'age', 'NumberOfTime30-59DaysPastDueNotWorse', 'DebtRatio',
       'MonthlyIncome', 'NumberOfOpenCreditLinesAndLoans',
       'NumberOfTimes90DaysLate', 'NumberRealEstateLoansOrLines',
       'NumberOfTime60-89DaysPastDueNotWorse', 'NumberOfDependents'],
      dtype='object')
```

In [3]:

```
loans.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150000 entries, 0 to 149999
Data columns (total 12 columns):
 #   Column                                Non-Null Count   Dtype  
---  ------                                --------------   -----  
 0   Sr_No                                 150000 non-null  int64  
 1   SeriousDlqin2yrs                      150000 non-null  int64  
 2   RevolvingUtilizationOfUnsecuredLines  150000 non-null  float64
 3   age                                   150000 non-null  int64  
 4   NumberOfTime30-59DaysPastDueNotWorse  150000 non-null  int64  
 5   DebtRatio                             150000 non-null  float64
 6   MonthlyIncome                         120269 non-null  float64
 7   NumberOfOpenCreditLinesAndLoans       150000 non-null  int64  
 8   NumberOfTimes90DaysLate               150000 non-null  int64  
 9   NumberRealEstateLoansOrLines          150000 non-null  int64  
 10  NumberOfTime60-89DaysPastDueNotWorse  150000 non-null  int64  
 11  NumberOfDependents                    146076 non-null  float64
dtypes: float64(4), int64(8)
memory usage: 13.7 MB
```

In [4]:

```
loans.sample(5).T
```

Out[4]:

	149458	69106	20001	13492
Sr_No	149459.000000	69107.000000	20002.000000	134924.000
SeriousDlqin2yrs	0.000000	0.000000	1.000000	0.000
RevolvingUtilizationOfUnsecuredLines	0.365909	0.017334	0.885052	0.5957
age	61.000000	66.000000	70.000000	43.000
NumberOfTime30-59DaysPastDueNotWorse	0.000000	0.000000	0.000000	0.000
DebtRatio	0.461703	0.010278	0.836776	0.544
MonthlyIncome	3733.000000	6615.000000	3436.000000	9900.000
NumberOfOpenCreditLinesAndLoans	14.000000	8.000000	17.000000	10.000
NumberOfTimes90DaysLate	0.000000	0.000000	0.000000	0.000
NumberRealEstateLoansOrLines	0.000000	1.000000	1.000000	4.000
NumberOfTime60-89DaysPastDueNotWorse	0.000000	0.000000	0.000000	0.000
NumberOfDependents	0.000000	0.000000	0.000000	0.000

Variable1 = "SeriousDlqin2yrs "

Is this a target column or a predictor column ?

In [5]:

```
#Target
```

What type of column is this? Numerical continuous, Numerical Discrete, DataTime, Geo

In [6]:

```
#Numerical discrete column
loans["SeriousDlqin2yrs"].dtypes
```

Out[6]:

```
dtype('int64')
```

Perform Univariate Analysis. If it is continuous then histograms and box plots, if it is discrete or categorical then bar charts and frequency tables.

In [7]:

```
loans["SeriousDlqin2yrs"].value_counts()
```

Out[7]:

```
0    139974
1     10026
Name: SeriousDlqin2yrs, dtype: int64
```

In [8]:

```
loans["SeriousDlqin2yrs"].value_counts()
```

Out[8]:

```
0    139974
1     10026
Name: SeriousDlqin2yrs, dtype: int64
```

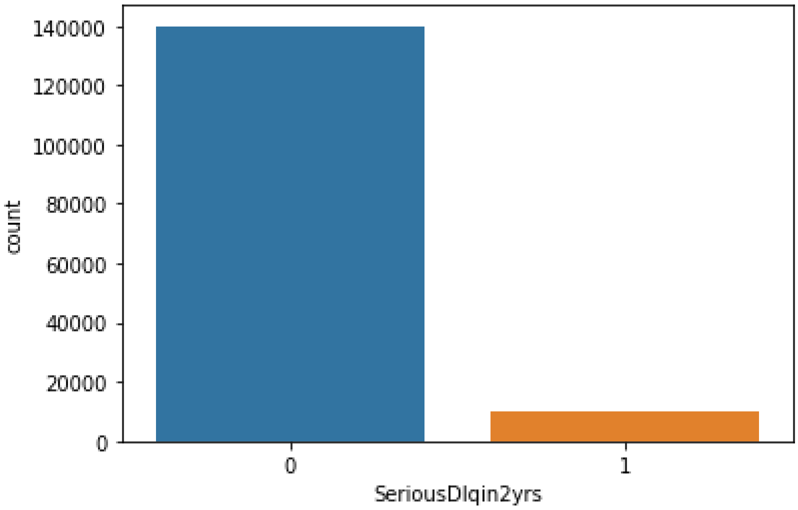
In [9]:

```
import seaborn as sns
sns.countplot(x="SeriousDlqin2yrs",data=loans)
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: F
utureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17bd8cda90>
```



List down the percentage of missing and percentage of lowside and high side outliers

In [10]:

```
loans["SeriousDlqin2yrs"].isnull().sum()
```

Out[10]:

```
0
```

Validate this variable, Perform some data checks. If you find any issues then write down the issues.

In [11]:

```
#No issues
```

If you find any issues then clean the variable, by applying appropriate imputation or any other substitutaion technique.

In [12]:

```
#NA
```

Create a cross tab or pivot table with the target variable and check the power of prediction. See if there are any apparent patterns in this variable

In [13]:

```
#NA
```

If you have any additional findings, that are missing above, highlight them here

In [14]:

```
#NA
```

Variable2 = "RevolvingUtilizationOfUnsecuredLines"

Is this a target column or a predictor column ?

In [15]:

```
#Predictor
```

What type of column is this? Numerical continuous, Numerical Discrete, DataTime, Geo

In [16]:

```
# Numerical continuous
loans["RevolvingUtilizationOfUnsecuredLines"].dtypes
```

Out[16]:

```
dtype('float64')
```

Perform Univariate Analysis. If it is continuous then histograms and box plots, if it is discrete or categorical then bar charts and frequency tables.

In [17]:

```
loans["RevolvingUtilizationOfUnsecuredLines"].describe()
```

Out[17]:

```
count    150000.000000
mean          6.048438
std         249.755371
min           0.000000
25%           0.029867
50%           0.154181
75%           0.559046
max       50708.000000
Name: RevolvingUtilizationOfUnsecuredLines, dtype: float64
```

In [18]:

```
loans["RevolvingUtilizationOfUnsecuredLines"].quantile([0,0.01,0.02,0.03,0.04,0.05,0.06
,0.07,0.08,0.09,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.91,0.92,0.93,0.94,0.95,0.96,0.97,
0.98,0.99,1])
```

Out[18]:

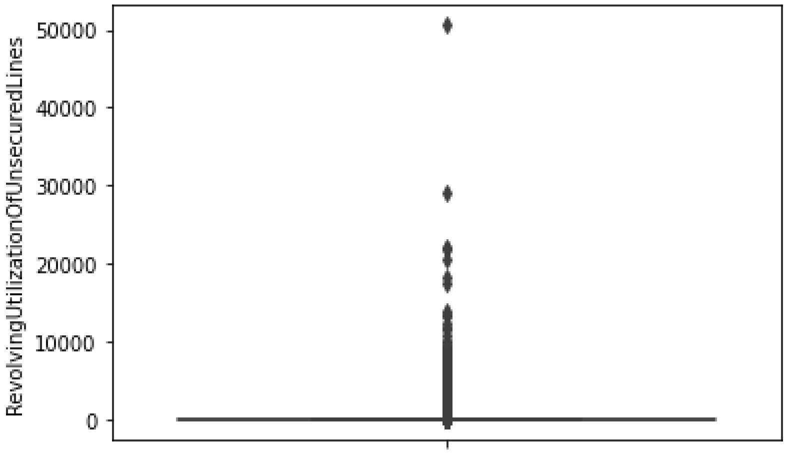
```
0.00        0.000000
0.01        0.000000
0.02        0.000000
0.03        0.000000
0.04        0.000000
0.05        0.000000
0.06        0.000000
0.07        0.000000
0.08        0.000708
0.09        0.001733
0.10        0.002969
0.20        0.019222
0.30        0.043461
0.40        0.083181
0.50        0.154181
0.60        0.271493
0.70        0.445136
0.80        0.698857
0.90        0.981278
0.91        1.000000
0.92        1.000000
0.93        1.000000
0.94        1.000000
0.95        1.000000
0.96        1.000000
0.97        1.000000
0.98        1.006199
0.99        1.092956
1.00    50708.000000
Name: RevolvingUtilizationOfUnsecuredLines, dtype: float64
```

In [19]:

```
sns.boxplot(y=loans["RevolvingUtilizationOfUnsecuredLines"])
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17bd2fb7b8>
```

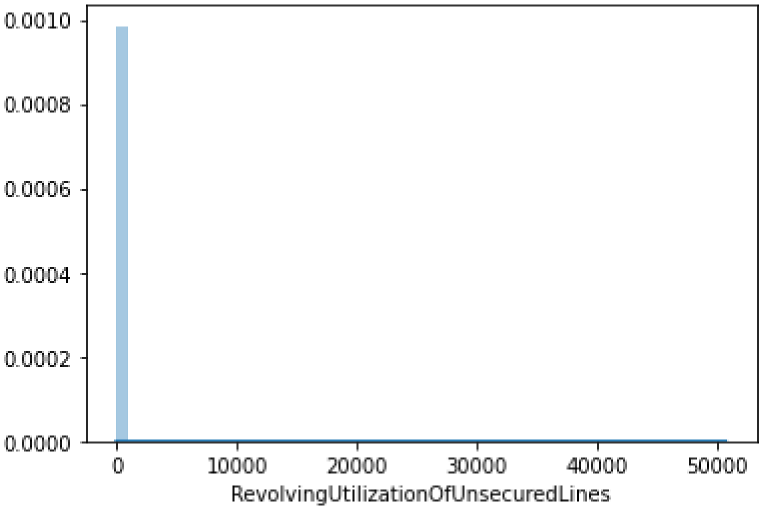


In [20]:

```
sns.distplot(loans["RevolvingUtilizationOfUnsecuredLines"])
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17bd2fb390>
```



List down the percentage of missing and percentage of lowside and high side outliers

In [21]:

```
loans["RevolvingUtilizationOfUnsecuredLines"].isnull().sum()
```

Out[21]:

```
0
```

In [22]:

```
#3% high side outliers
```

Validate this variable, Perform some data checks. If you find any issues then write down the issues.

In [23]:

```
# 3% issues - No missing values
```

If you find any issues then clean the variable, by applying appropriate imputation or any other substitutaion technique.

In [24]:

```
loans["util_new"]=loans["RevolvingUtilizationOfUnsecuredLines"]
loans["util_new"][loans["util_new"]>1]=loans["util_new"].median()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [25]:

```
loans["util_new"].describe()
```

Out[25]:

```
count    150000.000000
mean          0.300469
std           0.334855
min           0.000000
25%           0.029867
50%           0.154178
75%           0.506929
max           1.000000
Name: util_new, dtype: float64
```

Create a cross tab or pivot table with the target variable and check the power of prediction. See if there are any apparent patterns in this variable

In [26]:

```
util_pivot=pd.pivot_table(data=loans, values='util_new', columns="SeriousDlqin2yrs", ag
gfunc='mean')
print(util_pivot)
```

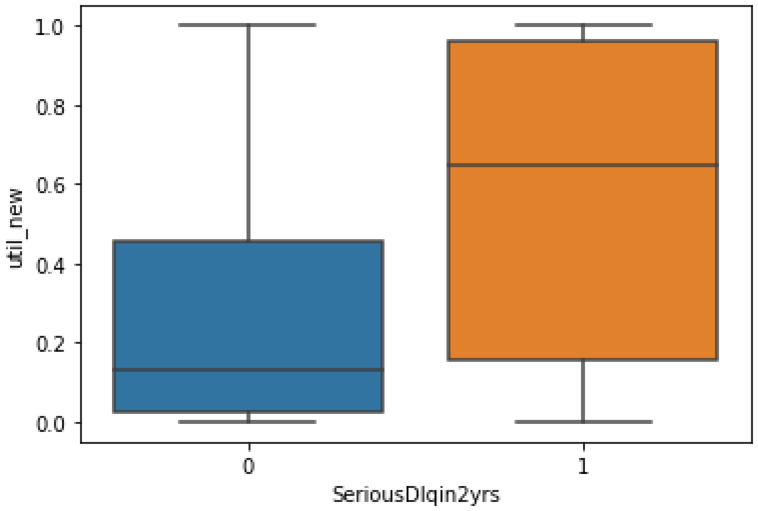
```
SeriousDlqin2yrs         0         1
util_new          0.280592  0.577982
```

In [27]:

```
sns.boxplot(x=loans["SeriousDlqin2yrs"], y=loans["util_new"])
```

Out[27]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17bce5aba8>
```

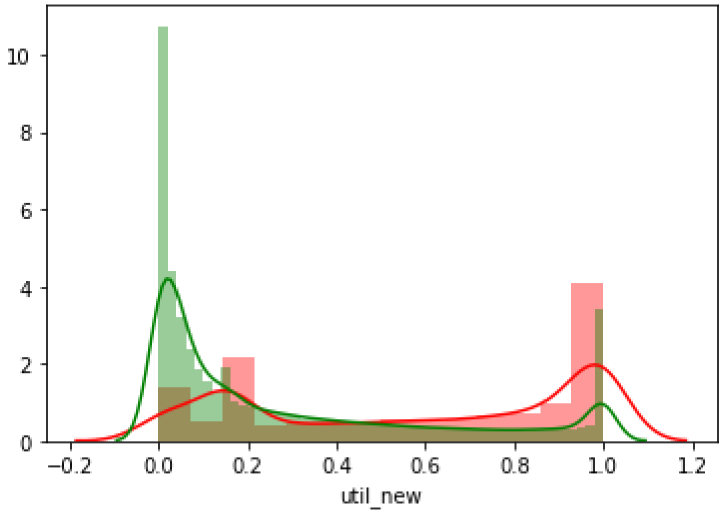


In [28]:

```
sns.distplot(loans[loans["SeriousDlqin2yrs"]==1]["util_new"], color="red")
sns.distplot(loans[loans["SeriousDlqin2yrs"]==0]["util_new"], color="green")
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17bcd299e8>
```

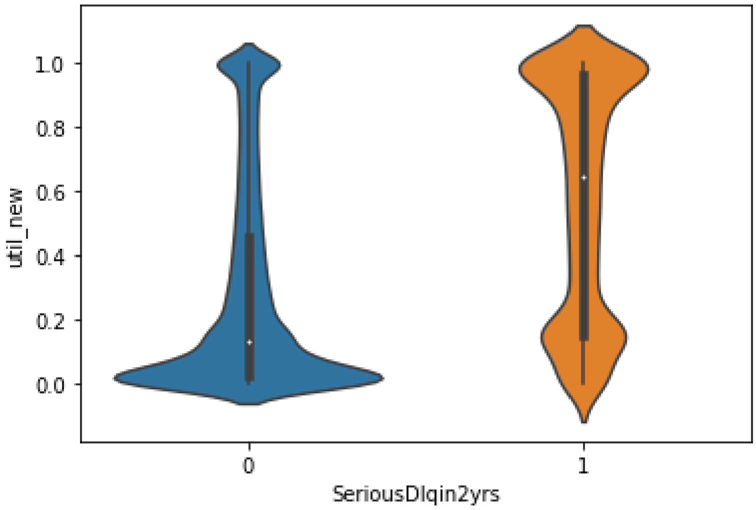


In [29]:

```
sns.violinplot(x=loans["SeriousDlqin2yrs"], y=loans["util_new"])
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17bcbc2f98>
```



If you have any additional findings, that are missing above, highlight them here

In [30]:

```
# Util is high when serious delinquency=1
```

Variable3= "age"

Is this a target column or a predictor column ?

In [31]:

```
#Predictor
```

What type of column is this? Numerical continuous, Numerical Discrete, DataTime, Geo

In [32]:

```
#Numeric Discrete
```

Perform Univariate Analysis. If it is continuous then histograms and box plots, if it is discrete or categorical then bar charts and frequency tables.

In [33]:

```
loans["age"].value_counts()
```

Out[33]:

```
49     3837
48     3806
50     3753
63     3719
47     3719
       ... 
101       3
109       2
107       1
105       1
0         1
Name: age, Length: 86, dtype: int64
```

In [34]:

```
loans["age"].describe()
```

Out[34]:

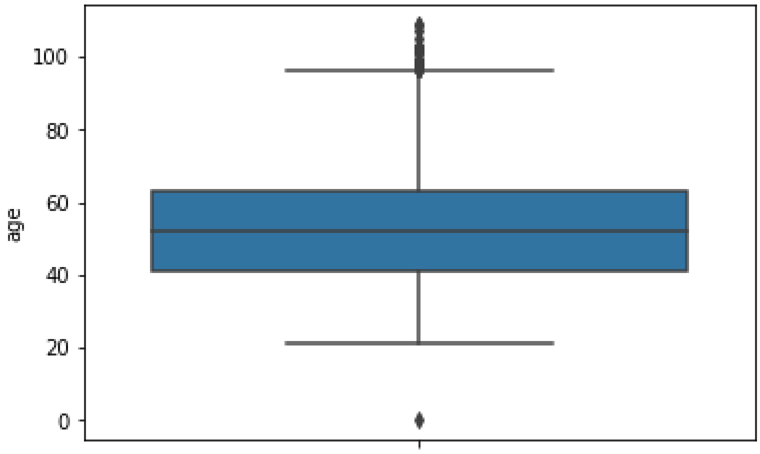
```
count    150000.000000
mean         52.295207
std          14.771866
min           0.000000
25%          41.000000
50%          52.000000
75%          63.000000
max         109.000000
Name: age, dtype: float64
```

In [35]:

```
sns.boxplot(y=loans["age"])
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17bccb1320>
```

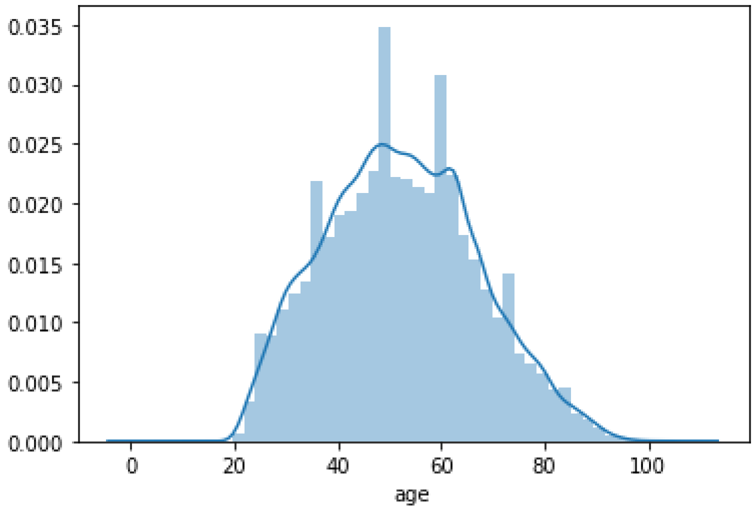


In [36]:

```
sns.distplot(loans["age"])
```

Out[36]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17ba300828>
```



List down the percentage of missing and percentage of lowside and high side outliers

In [37]:

```
loans["age"].quantile([0,0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.2,0.3,0.4,
0.5,0.6,0.7,0.8,0.9,0.91,0.92,0.93,0.94,0.95,0.96,0.97,0.98,0.99,1])
```

Out[37]:

```
0.00      0.0
0.01     24.0
0.02     25.0
0.03     27.0
0.04     28.0
0.05     29.0
0.06     30.0
0.07     30.0
0.08     31.0
0.09     32.0
0.10     33.0
0.20     39.0
0.30     44.0
0.40     48.0
0.50     52.0
0.60     56.0
0.70     61.0
0.80     65.0
0.90     72.0
0.91     73.0
0.92     74.0
0.93     75.0
0.94     76.0
0.95     78.0
0.96     79.0
0.97     81.0
0.98     84.0
0.99     87.0
1.00    109.0
Name: age, dtype: float64
```

Validate this variable, Perform some data checks. If you find any issues then write down the issues.

In [38]:

```
#1 % high side outliers
```

If you find any issues then clean the variable, by applying appropriate imputation or any other substitutaion technique.

In [39]:

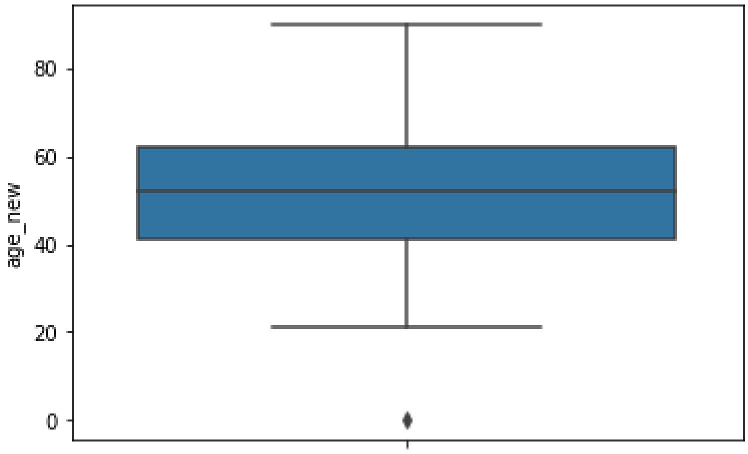
```
loans["age_new"]=loans["age"]
loans["age_new"][loans["age_new"]>90]=loans["age"].median()
sns.boxplot(y=loans["age_new"])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[39]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17bcd92668>
```



Create a cross tab or pivot table with the target variable and check the power of prediction. See if there are any apparent patterns in this variable

In [40]:

```
age_pivot=pd.pivot_table(data=loans, values='age_new', columns="SeriousDlqin2yrs", aggf
unc='mean')
print(age_pivot)
```

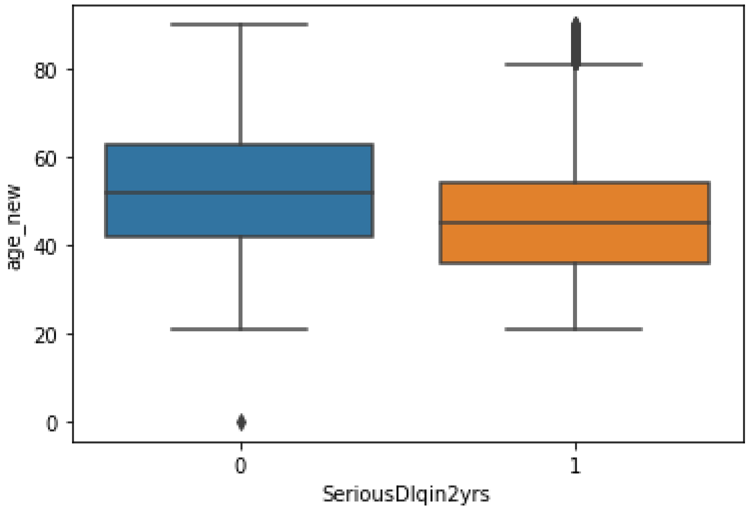
```
SeriousDlqin2yrs          0          1
age_new           52.610878  45.880311
```

In [41]:

```
sns.boxplot(x=loans["SeriousDlqin2yrs"], y=loans["age_new"])
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17ba1d3390>
```

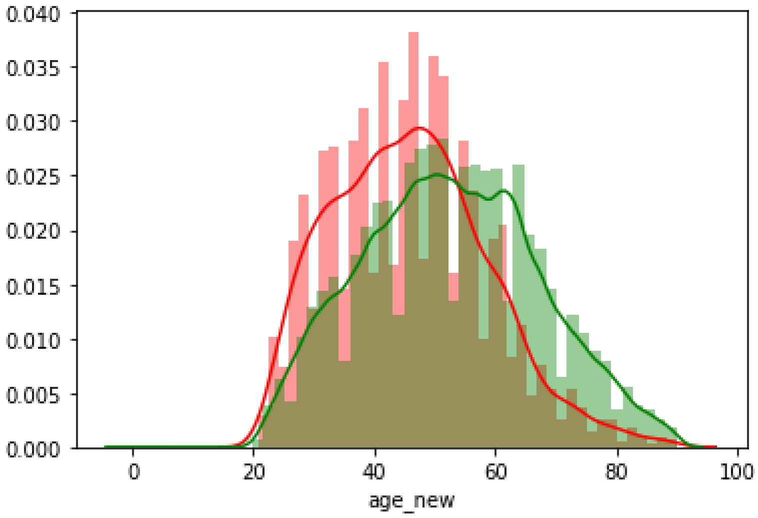


In [42]:

```
sns.distplot(loans[loans["SeriousDlqin2yrs"]==1]["age_new"], color="red")
sns.distplot(loans[loans["SeriousDlqin2yrs"]==0]["age_new"], color="green")
```

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17ba14e080>
```

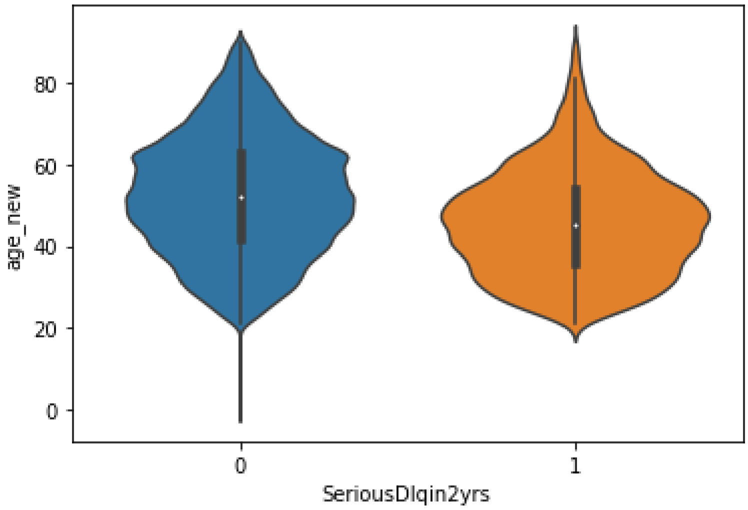


In [43]:

```
sns.violinplot(x=loans["SeriousDlqin2yrs"], y=loans["age_new"])
```

Out[43]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b9faecc0>
```



If you have any additional findings, that are missing above, highlight them here

In [44]:

```
#NA
```

Variable4 = "NumberOfTime30-59DaysPastDueNotWorse"

Is this a target column or a predictor column ?

In [45]:

```
#predictor
```

What type of column is this? Numerical continuous, Numerical Discrete, DataTime, Geo

In [46]:

```
#Numerical discrete
```

Perform Univariate Analysis. If it is continuous then histograms and box plots, if it is discrete or categorical then bar charts and frequency tables.

In [47]:

```
loans["NumberOfTime30-59DaysPastDueNotWorse"].value_counts()
```

Out[47]:

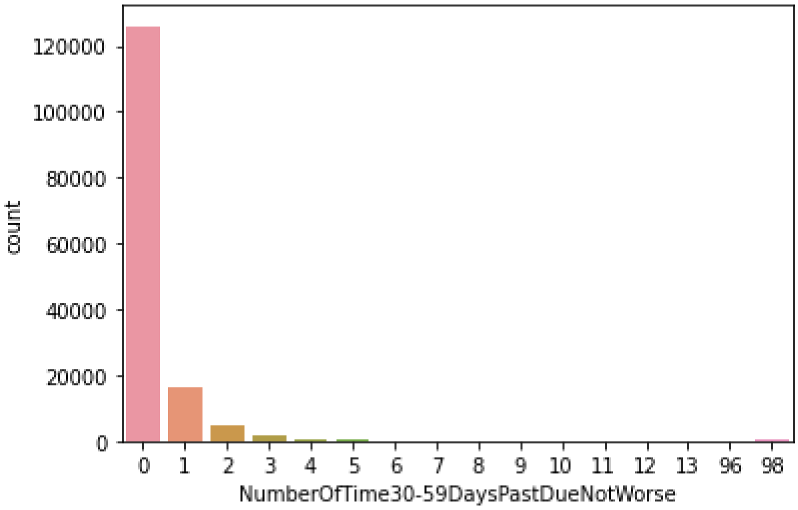
```
0     126018
1      16033
2       4598
3       1754
4        747
5        342
98       264
6        140
7         54
8         25
9         12
96         5
10         4
12         2
13         1
11         1
Name: NumberOfTime30-59DaysPastDueNotWorse, dtype: int64
```

In [48]:

```
sns.countplot(x="NumberOfTime30-59DaysPastDueNotWorse",  data=loans)
```

Out[48]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b9f80860>
```



List down the percentage of missing and percentage of lowside and high side outliers

In [49]:

```
loans["NumberOfTime30-59DaysPastDueNotWorse"].value_counts()/len(loans)
```

Out[49]:

```
0     0.840120
1     0.106887
2     0.030653
3     0.011693
4     0.004980
5     0.002280
98    0.001760
6     0.000933
7     0.000360
8     0.000167
9     0.000080
96    0.000033
10    0.000027
12    0.000013
13    0.000007
11    0.000007
Name: NumberOfTime30-59DaysPastDueNotWorse, dtype: float64
```

Validate this variable, Perform some data checks. If you find any issues then write down the issues.

In [50]:

```
#Less than 1% have issues
```

If you find any issues then clean the variable, by applying appropriate imputation or any other substitutaion technique.

In [51]:

```
cross_tab_30dpd_target=pd.crosstab(loans['NumberOfTime30-59DaysPastDueNotWorse'],loans[
'SeriousDlqin2yrs'])
cross_tab_30dpd_target
```

Out[51]:

SeriousDlqin2yrs	0	1
NumberOfTime30-59DaysPastDueNotWorse		
0	120977	5041
1	13624	2409
2	3379	1219
3	1136	618
4	429	318
5	188	154
6	66	74
7	26	28
8	17	8
9	8	4
10	1	3
11	0	1
12	1	1
13	0	1
96	1	4
98	121	143

In [52]:

```
cross_tab_30dpd_target_percent=cross_tab_30dpd_target.astype(float).div(cross_tab_30dpd
_target.sum(axis=1), axis=0)
round(cross_tab_30dpd_target_percent,2)
```

Out[52]:

SeriousDlqin2yrs	0	1
NumberOfTime30-59DaysPastDueNotWorse		
0	0.96	0.04
1	0.85	0.15
2	0.73	0.27
3	0.65	0.35
4	0.57	0.43
5	0.55	0.45
6	0.47	0.53
7	0.48	0.52
8	0.68	0.32
9	0.67	0.33
10	0.25	0.75
11	0.00	1.00
12	0.50	0.50
13	0.00	1.00
96	0.20	0.80
98	0.46	0.54

In [53]:

```
loans['num_30_59_dpd_new']=loans['NumberOfTime30-59DaysPastDueNotWorse']
loans['num_30_59_dpd_new'][loans['num_30_59_dpd_new']>12]=6
loans['num_30_59_dpd_new']

loans['num_30_59_dpd_new'].value_counts(sort=False)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[53]:

```
0     126018
1      16033
2       4598
3       1754
4        747
5        342
6        410
7         54
8         25
9         12
10         4
11         1
12         2
Name: num_30_59_dpd_new, dtype: int64
```

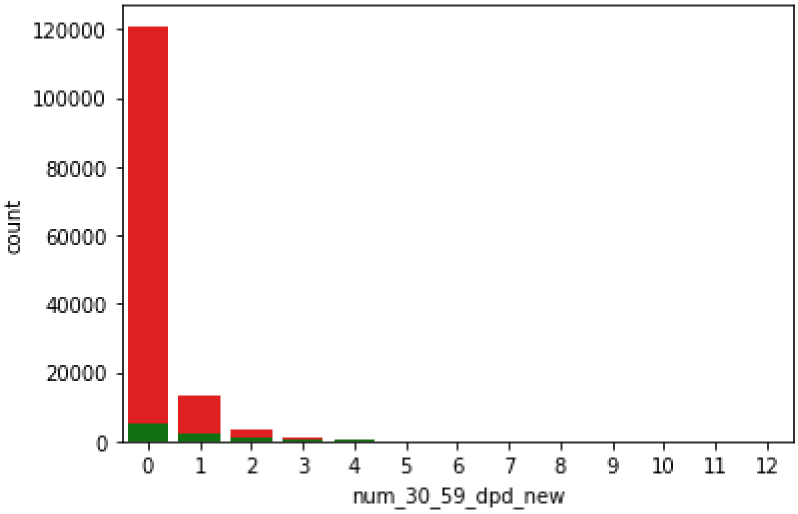
Create a cross tab or pivot table with the target variable and check the power of prediction. See if there are any apparent patterns in this variable

In [54]:

```
sns.countplot(loans[loans["SeriousDlqin2yrs"]==0]["num_30_59_dpd_new"], color="red")
sns.countplot(loans[loans["SeriousDlqin2yrs"]==1]["num_30_59_dpd_new"], color="green")
```

Out[54]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b9c46710>
```

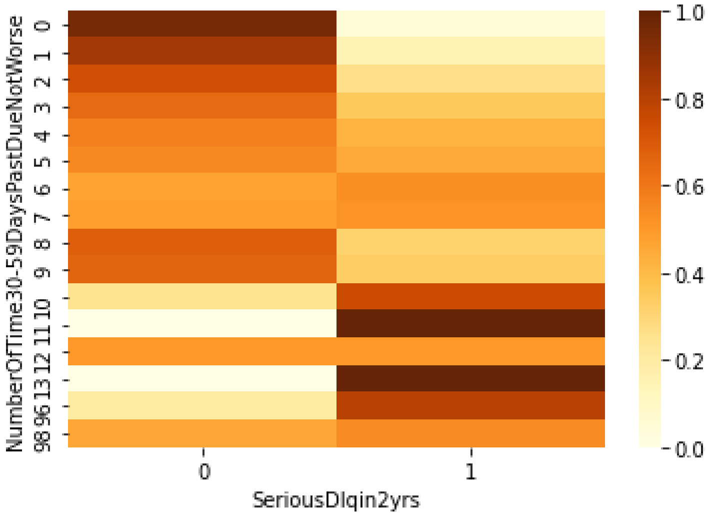


In [55]:

```
cross_tab_30dpd_target_percent=cross_tab_30dpd_target.astype(float).div(cross_tab_30dpd
_target.sum(axis=1), axis=0)
round(cross_tab_30dpd_target_percent,2)
sns.heatmap(cross_tab_30dpd_target_percent, cmap="YlOrBr")
```

Out[55]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b9ee49e8>
```



If you have any additional findings, that are missing above, highlight them here

In [56]:

```
# As the 30DPD incrases the delinquencies are increasing
```

Variable5 = "DebtRatio"

Is this a target column or a predictor column ?

In [57]:

```
#Predictor
```

What type of column is this? Numerical continuous, Numerical Discrete, DataTime, Geo

In [58]:

```
# Numerical continuous
loans["DebtRatio"].dtypes
```

Out[58]:

```
dtype('float64')
```

Perform Univariate Analysis. If it is continuous then histograms and box plots, if it is discrete or categorical then bar charts and frequency tables.

In [59]:

```
loans["DebtRatio"].describe()
```

Out[59]:

```
count    150000.000000
mean        353.005076
std        2037.818523
min           0.000000
25%           0.175074
50%           0.366508
75%           0.868254
max      329664.000000
Name: DebtRatio, dtype: float64
```

In [60]:

```
loans["DebtRatio"].quantile([0,0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.2,0.3
,0.4,0.5,0.6,0.7,0.75,0.76,0.77, 0.78,0.8,0.81,0.82, 0.83,0.86,0.90,0.93,0.94,0.95,0.96
,0.97,0.98,0.99,1])
```

Out[60]:

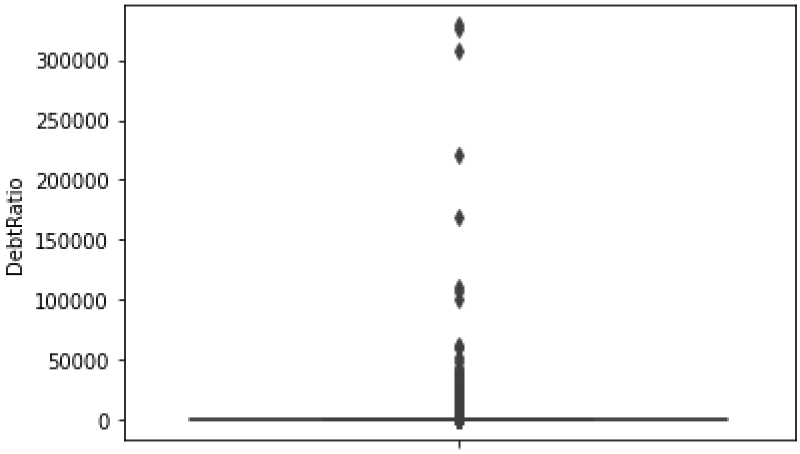
```
0.00         0.000000
0.01         0.000000
0.02         0.000000
0.03         0.000480
0.04         0.002278
0.05         0.004329
0.06         0.006799
0.07         0.009997
0.08         0.014780
0.09         0.021249
0.10         0.030874
0.20         0.133773
0.30         0.213697
0.40         0.287460
0.50         0.366508
0.60         0.467506
0.70         0.649189
0.75         0.868254
0.76         0.951184
0.77         1.058832
0.78         1.275069
0.80         4.000000
0.81        14.000000
0.82        31.000000
0.83        61.000000
0.86       453.000000
0.90      1267.000000
0.93      1917.070000
0.94      2172.060000
0.95      2449.000000
0.96      2791.000000
0.97      3225.000000
0.98      3839.000000
0.99      4979.040000
1.00    329664.000000
Name: DebtRatio, dtype: float64
```

In [61]:

```
sns.boxplot(y=loans["DebtRatio"])
```

Out[61]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17ba1f3588>
```

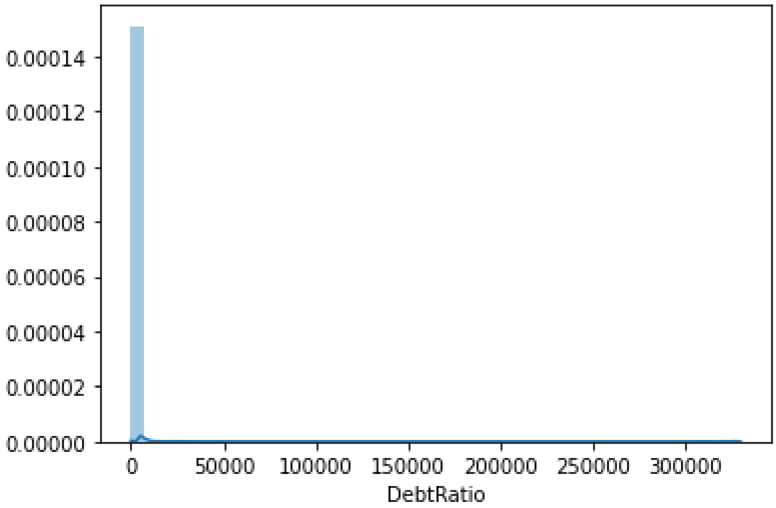


In [62]:

```
sns.distplot(loans["DebtRatio"])
```

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17ba1378d0>
```



List down the percentage of missing and percentage of lowside and high side outliers

In [63]:

```
loans["DebtRatio"].isnull().sum()
```

Out[63]:

```
0
```

In [64]:

```
#24% outliers
#No missing values
```

Validate this variable, Perform some data checks. If you find any issues then write down the issues.

In [65]:

```
#24% outliers
#No missing values
```

If you find any issues then clean the variable, by applying appropriate imputation or any other substitutaion technique.

In [66]:

```
loans["DebtRatio_new_ind"]=(loans["DebtRatio"]>1)*1
loans["DebtRatio_new_ind"].value_counts()
```

Out[66]:

```
0    114863
1     35137
Name: DebtRatio_new_ind, dtype: int64
```

In [67]:

```
loans["DebtRatio_new"]=loans["DebtRatio"]
loans["DebtRatio_new"][loans["DebtRatio_new"]>1]=loans["DebtRatio"].median()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [68]:

```
loans["DebtRatio_new"].describe()
```

Out[68]:

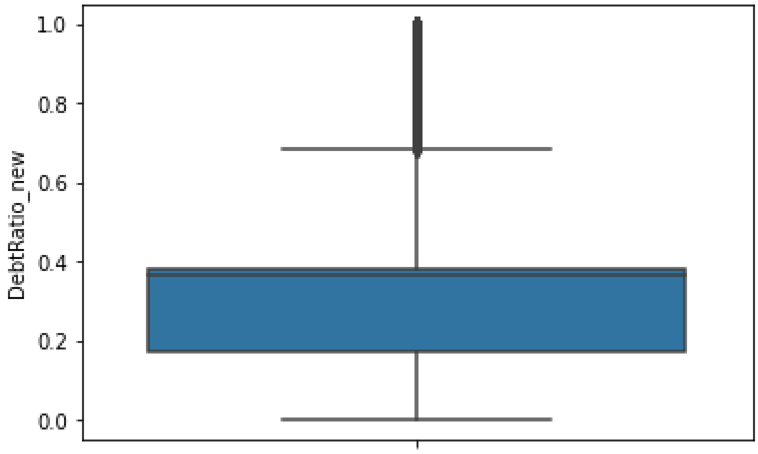
```
count    150000.000000
mean          0.317893
std           0.199835
min           0.000000
25%           0.175074
50%           0.366506
75%           0.380021
max           1.000000
Name: DebtRatio_new, dtype: float64
```

In [69]:

```
sns.boxplot(y=loans["DebtRatio_new"])
```

Out[69]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b8979160>
```

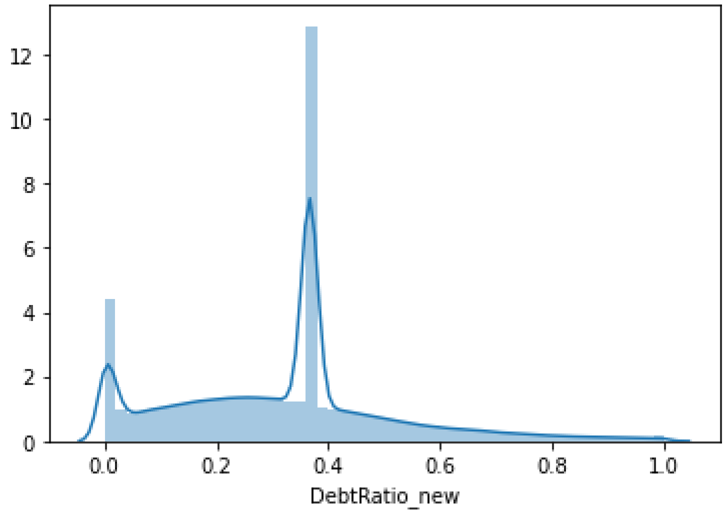


In [70]:

```
sns.distplot(loans["DebtRatio_new"])
```

Out[70]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b8958a90>
```



Create a cross tab or pivot table with the target variable and check the power of prediction. See if there are any apparent patterns in this variable

In [71]:

```python
Debt_rato_pivot=pd.pivot_table(data=loans, values='DebtRatio_new', columns="SeriousDlqin2yrs", aggfunc='mean')
print(Debt_rato_pivot)
```

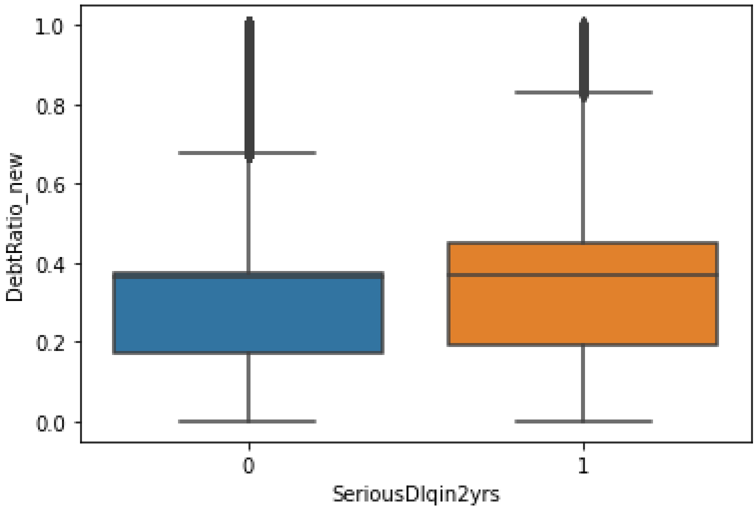
```
SeriousDlqin2yrs        0         1
DebtRatio_new     0.31543  0.352284
```

In [72]:

```python
sns.boxplot(x=loans["SeriousDlqin2yrs"], y=loans["DebtRatio_new"])
```

Out[72]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b884def0>
```

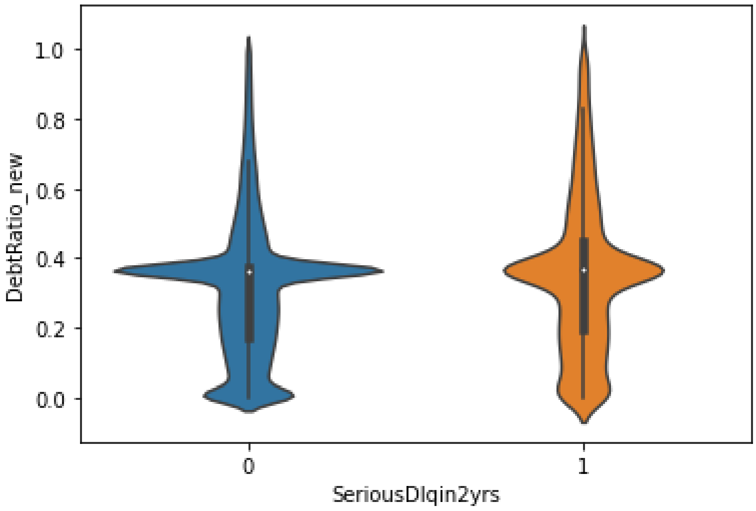


In [73]:

```python
sns.violinplot(x=loans["SeriousDlqin2yrs"], y=loans["DebtRatio_new"])
```

Out[73]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b87bfa58>
```



If you have any additional findings, that are missing above, highlight them here

In [74]:

```
#NA
```

Variable6 = "MonthlyIncome"

Is this a target column or a predictor column ?

In [75]:

```
#Predictor
```

What type of column is this? Numerical continuous, Numerical Discrete, DataTime, Geo

In [76]:

```
# Numerical continuous
loans["MonthlyIncome"].dtypes
```

Out[76]:

```
dtype('float64')
```

Perform Univariate Analysis. If it is continuous then histograms and box plots, if it is discrete or categorical then bar charts and frequency tables.

In [77]:

```
loans["MonthlyIncome"].describe().round()
```

Out[77]:

```
count     120269.0
mean        6670.0
std        14385.0
min            0.0
25%         3400.0
50%         5400.0
75%         8249.0
max      3008750.0
Name: MonthlyIncome, dtype: float64
```

In [78]:

```
loans["MonthlyIncome"].quantile([0,0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.2
,0.3,0.4,0.5,0.6,0.7, 0.83,0.86,0.90,0.93,0.94,0.95,0.96,0.97,0.98,0.99,1])
```

Out[78]:

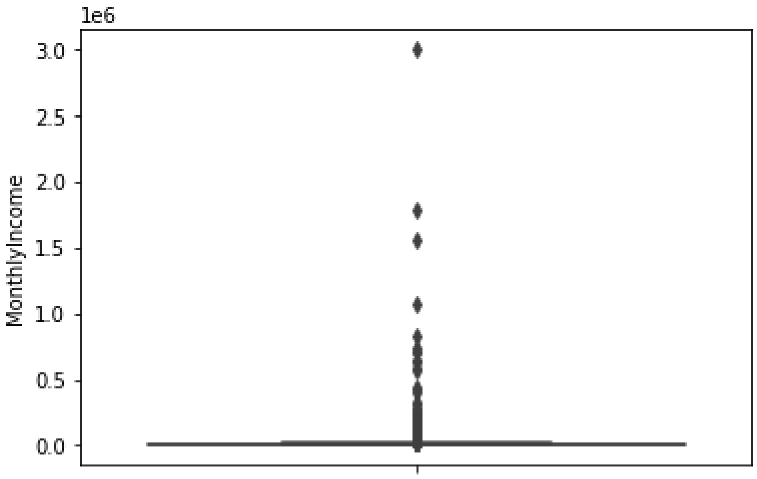
```
0.00          0.00
0.01          0.00
0.02        250.00
0.03        800.00
0.04       1012.16
0.05       1300.00
0.06       1500.00
0.07       1640.00
0.08       1800.00
0.09       1981.00
0.10       2005.00
0.20       3000.00
0.30       3800.00
0.40       4544.20
0.50       5400.00
0.60       6300.00
0.70       7500.00
0.83       9800.00
0.86      10417.00
0.90      11666.00
0.93      13000.00
0.94      13716.00
0.95      14587.60
0.96      15636.28
0.97      17000.00
0.98      19600.00
0.99      25000.00
1.00    3008750.00
Name: MonthlyIncome, dtype: float64
```

In [79]:

```
sns.boxplot(y=loans["MonthlyIncome"])
```

Out[79]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b87a1048>
```

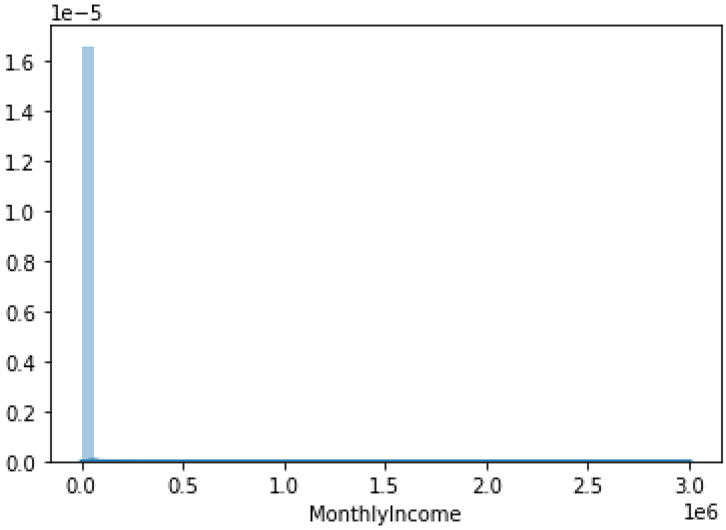


In [80]:

```
sns.distplot(loans["MonthlyIncome"])
```

Out[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b8710668>
```



List down the percentage of missing and percentage of lowside and high side outliers

In [81]:

```
loans["MonthlyIncome"].isnull().sum()
```

Out[81]:

```
29731
```

In [82]:

```
loans["MonthlyIncome"].isnull().sum()/len(loans)
```

Out[82]:

```
0.19820666666666667
```

Validate this variable, Perform some data checks. If you find any issues then write down the issues.

In [83]:

```
#2% igh side outliers
```

In [84]:

```
#19.8% missing values
```

If you find any issues then clean the variable, by applying appropriate imputation or any other substitutaion technique.

In [85]:

```
loans['MonthlyIncome_ind']=1
loans['MonthlyIncome_ind'][loans['MonthlyIncome'].isnull()]=0
loans['MonthlyIncome_ind'].value_counts(sort=False)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[85]:

```
0     29731
1    120269
Name: MonthlyIncome_ind, dtype: int64
```

In [86]:

```
loans['MonthlyIncome_new']=loans['MonthlyIncome']
loans['MonthlyIncome_new'][loans['MonthlyIncome'].isnull()]=loans['MonthlyIncome'].medi
an()
round(loans['MonthlyIncome_new'].describe())
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[86]:

```
count     150000.0
mean        6418.0
std        12890.0
min            0.0
25%         3903.0
50%         5400.0
75%         7400.0
max      3008750.0
Name: MonthlyIncome_new, dtype: float64
```

In [87]:

```
loans['MonthlyIncome_new'][loans['MonthlyIncome']>20000]=loans['MonthlyIncome'].median
()
round(loans['MonthlyIncome_new'].describe())
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

Out[87]:

```
count    150000.0
mean       5902.0
std        3298.0
min           0.0
25%        3903.0
50%        5400.0
75%        7100.0
max       20000.0
Name: MonthlyIncome_new, dtype: float64
```

Create a cross tab or pivot table with the target variable and check the power of prediction. See if there are any apparent patterns in this variable

In [88]:

```
MonthlyIncome_pivot=pd.pivot_table(data=loans, values='MonthlyIncome_new', columns="Ser
iousDlqin2yrs", aggfunc='mean')
print(MonthlyIncome_pivot)
```

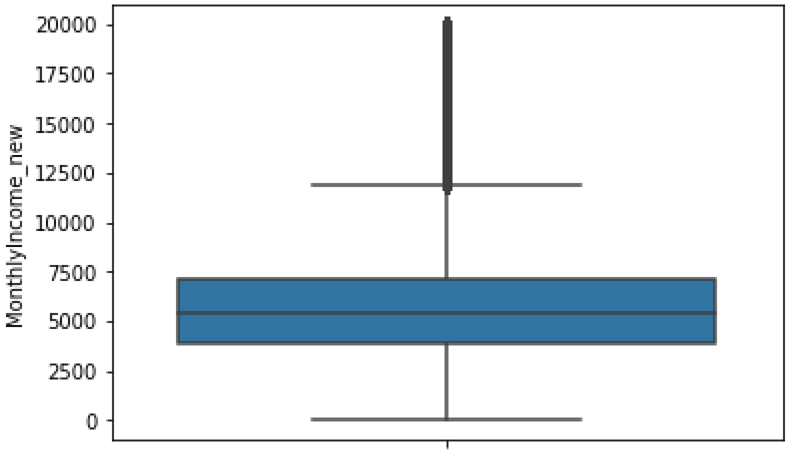
```
SeriousDlqin2yrs            0            1
MonthlyIncome_new  5949.503372  5237.481049
```

In [89]:

```
sns.boxplot(y=loans["MonthlyIncome_new"])
```

Out[89]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b8608e80>
```

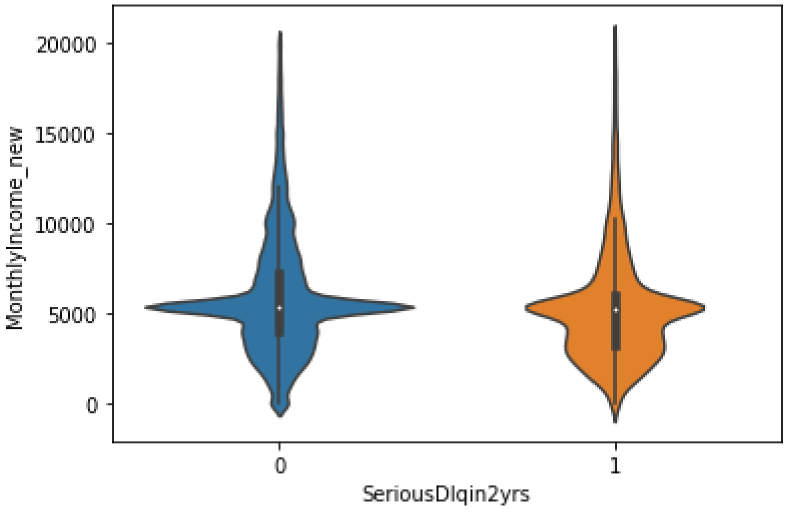


In [90]:

```
sns.violinplot(x=loans["SeriousDlqin2yrs"], y=loans["MonthlyIncome_new"])
```

Out[90]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b8571f60>
```



If you have any additional findings, that are missing above, highlight them here

In [91]:

```
#Monthly income seams to have no impact on the overall delinqunecy
```

Variable7 = "NumberOfOpenCreditLinesAndLoans"

Is this a target column or a predictor column ?

In [92]:

```
#Predictor
```

What type of column is this? Numerical continuous, Numerical Discrete, DataTime, Geo

In [93]:

```
#Numerical discrete
loans["NumberOfOpenCreditLinesAndLoans"].dtypes
```

Out[93]:

```
dtype('int64')
```

Perform Univariate Analysis. If it is continuous then histograms and box plots, if it is discrete or categorical then bar charts and frequency tables.

In [94]:

```
loans["NumberOfOpenCreditLinesAndLoans"].value_counts(sort=False)
```

Out[94]:

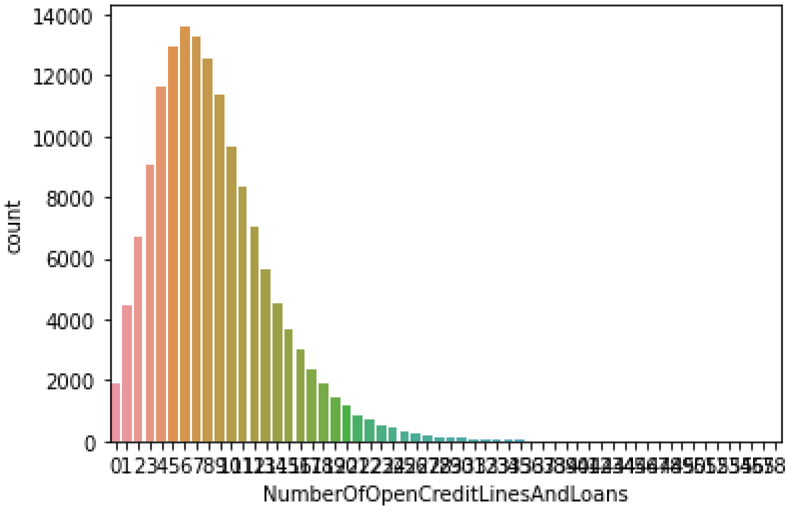
```
0      1888
1      4438
2      6666
3      9058
4     11609
5     12931
6     13614
7     13245
8     12562
9     11355
10     9624
11     8321
12     7005
13     5667
14     4546
15     3645
16     3000
17     2370
18     1874
19     1433
20     1169
21      864
22      685
23      533
24      422
25      337
26      239
27      194
28      150
29      114
30       88
31       74
32       52
33       47
34       35
35       27
36       18
37        7
38       13
39        9
40       10
41        4
42        8
43        8
44        2
45        8
46        3
47        2
48        6
49        4
50        2
51        2
52        3
53        1
54        4
56        2
57        2
58        1
Name: NumberOfOpenCreditLinesAndLoans, dtype: int64
```

In [95]:

```
sns.countplot(x="NumberOfOpenCreditLinesAndLoans",  data=loans)
```

Out[95]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b84fe208>
```

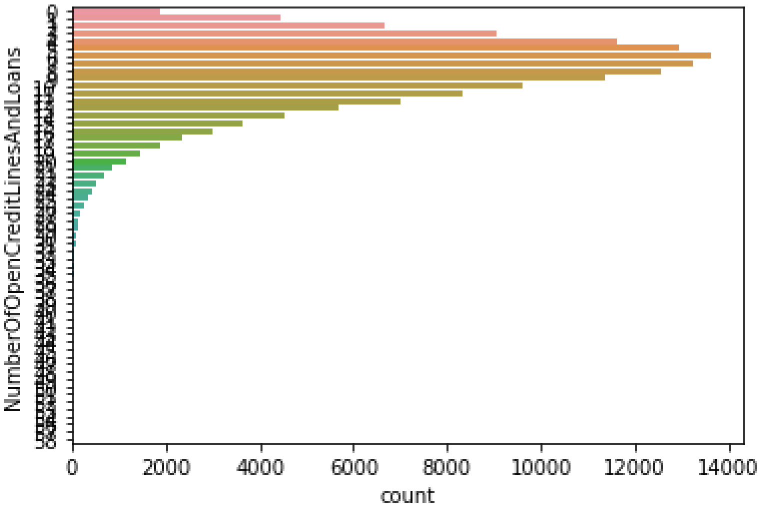


In [96]:

```
sns.countplot(y="NumberOfOpenCreditLinesAndLoans",  data=loans)
```

Out[96]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17ba1ebcf8>
```



In [97]:

```
loans["NumberOfOpenCreditLinesAndLoans"].value_counts()/len(loans)
```

Out[97]:

```
6     0.090760
7     0.088300
5     0.086207
8     0.083747
4     0.077393
9     0.075700
10    0.064160
3     0.060387
11    0.055473
12    0.046700
2     0.044440
13    0.037780
14    0.030307
1     0.029587
15    0.024300
16    0.020000
17    0.015800
0     0.012587
18    0.012493
19    0.009553
20    0.007793
21    0.005760
22    0.004567
23    0.003553
24    0.002813
25    0.002247
26    0.001593
27    0.001293
28    0.001000
29    0.000760
30    0.000587
31    0.000493
32    0.000347
33    0.000313
34    0.000233
35    0.000180
36    0.000120
38    0.000087
40    0.000067
39    0.000060
45    0.000053
43    0.000053
42    0.000053
37    0.000047
48    0.000040
41    0.000027
54    0.000027
49    0.000027
46    0.000020
52    0.000020
51    0.000013
56    0.000013
57    0.000013
50    0.000013
47    0.000013
44    0.000013
53    0.000007
58    0.000007
Name: NumberOfOpenCreditLinesAndLoans, dtype: float64
```

In [98]:

```
loans["NumberOfOpenCreditLinesAndLoans"].describe()
```

Out[98]:

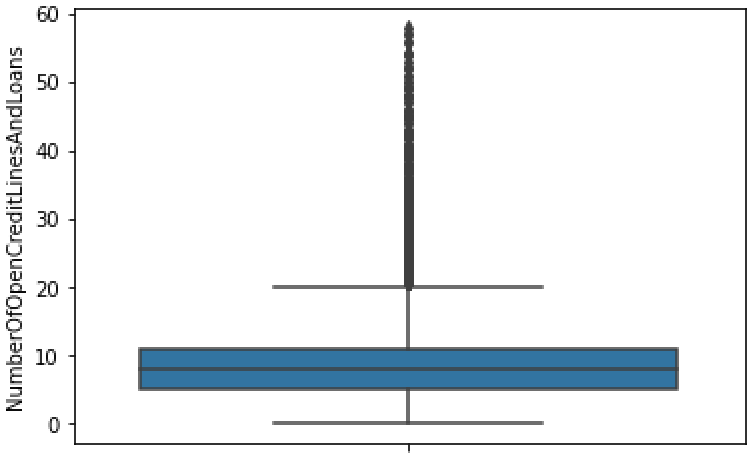
```
count    150000.000000
mean          8.452760
std           5.145951
min           0.000000
25%           5.000000
50%           8.000000
75%          11.000000
max          58.000000
Name: NumberOfOpenCreditLinesAndLoans, dtype: float64
```

In [99]:

```
sns.boxplot(y=loans["NumberOfOpenCreditLinesAndLoans"])
```

Out[99]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b8549e48>
```



List down the percentage of missing and percentage of lowside and high side outliers

In [100]:

```
loans["NumberOfOpenCreditLinesAndLoans"].isnull().sum()
```

Out[100]:

```
0
```

In [101]:

```
loans["NumberOfOpenCreditLinesAndLoans"].quantile([0,0.01,0.02,0.03,0.04,0.05,0.06,0.07
,0.08,0.09,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.91,0.92,0.93,0.94,0.95,0.96,0.97,0.98,
0.99,1])
```

Out[101]:

```
0.00     0.0
0.01     0.0
0.02     1.0
0.03     1.0
0.04     1.0
0.05     2.0
0.06     2.0
0.07     2.0
0.08     2.0
0.09     3.0
0.10     3.0
0.20     4.0
0.30     5.0
0.40     6.0
0.50     8.0
0.60     9.0
0.70    10.0
0.80    12.0
0.90    15.0
0.91    16.0
0.92    16.0
0.93    17.0
0.94    17.0
0.95    18.0
0.96    19.0
0.97    20.0
0.98    22.0
0.99    24.0
1.00    58.0
Name: NumberOfOpenCreditLinesAndLoans, dtype: float64
```

Validate this variable, Perform some data checks. If you find any issues then write down the issues.

In [102]:

```
#3% High side outliers
#No Missing values
```

If you find any issues then clean the variable, by applying appropriate imputation or any other substitutaion technique.

In [103]:

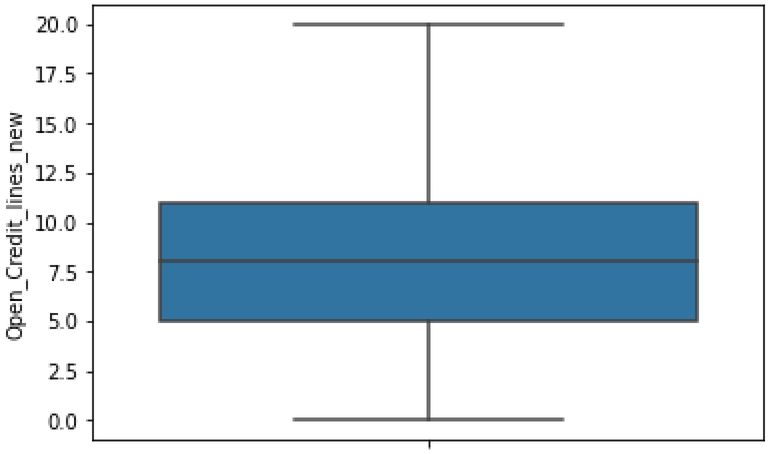
```
loans["Open_Credit_lines_new"]=loans["NumberOfOpenCreditLinesAndLoans"]
loans["Open_Credit_lines_new"][loans["Open_Credit_lines_new"]>20]=loans["NumberOfOpenCr
editLinesAndLoans"].median()
sns.boxplot(y=loans["Open_Credit_lines_new"])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[103]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b81d3a58>
```



Create a cross tab or pivot table with the target variable and check the power of prediction. See if there are any apparent patterns in this variable

In [104]:

```
credit_lines_pivot=pd.pivot_table(data=loans, values='Open_Credit_lines_new', columns=
"SeriousDlqin2yrs", aggfunc='mean')
print(credit_lines_pivot)
```

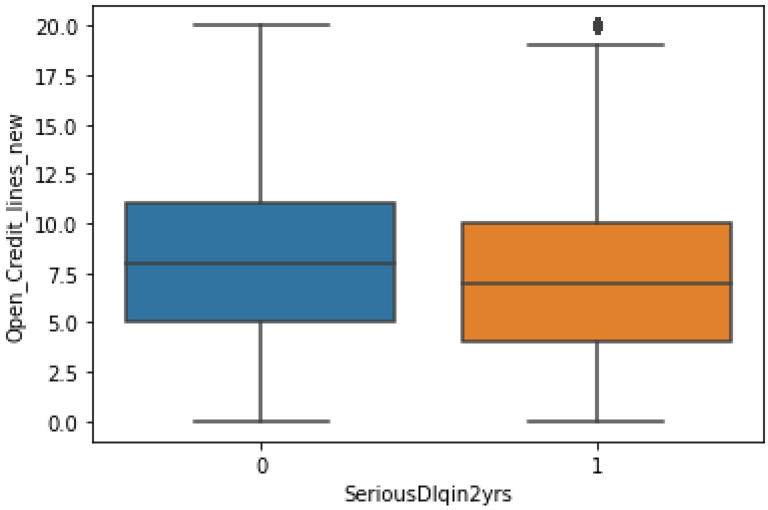
```
SeriousDlqin2yrs             0         1
Open_Credit_lines_new  8.05251  7.408039
```

In [105]:

```
sns.boxplot(x=loans["SeriousDlqin2yrs"], y=loans["Open_Credit_lines_new"])
```

Out[105]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b81399e8>
```

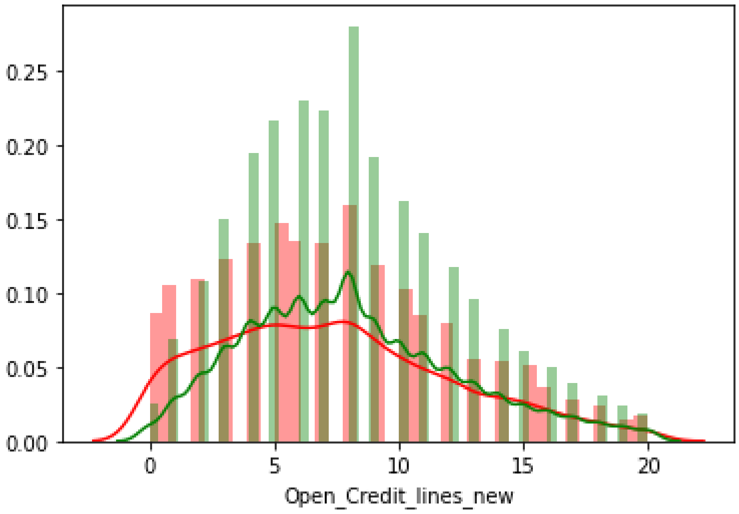


In [106]:

```
sns.distplot(loans[loans["SeriousDlqin2yrs"]==1]["Open_Credit_lines_new"], color="red")
sns.distplot(loans[loans["SeriousDlqin2yrs"]==0]["Open_Credit_lines_new"], color="gree
n")
```

Out[106]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b80ee780>
```

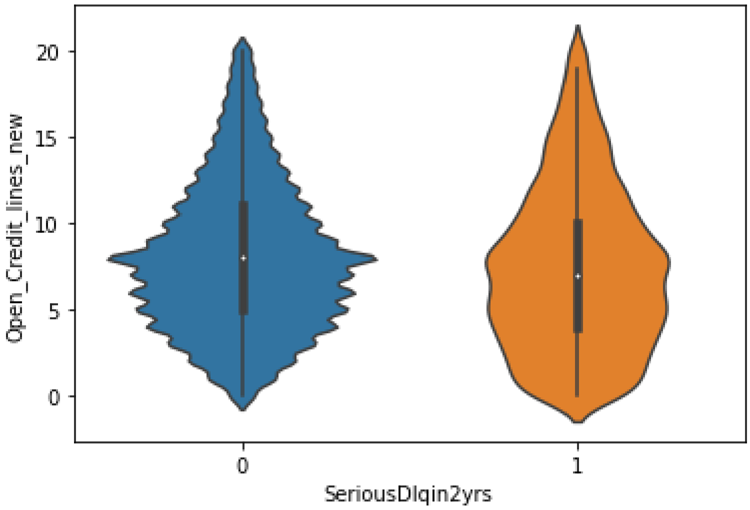


In [107]:

```
sns.violinplot(x=loans["SeriousDlqin2yrs"], y=loans["Open_Credit_lines_new"])
```

Out[107]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b7f8e208>
```



If you have any additional findings, that are missing above, highlight them here

In [108]:

```
#Looks like Open credit lines has no direct impact on the target
```

Variable8 = "NumberOfTimes90DaysLate"

This variable is similar to 30_59_DPD. Directly perform cleaning of this varibale with very few lines of code. No need of visualization

In [109]:

```
cross_tab_90dpd_target=pd.crosstab(loans['NumberOfTimes90DaysLate'],loans['SeriousDlqin2yrs'])
cross_tab_90dpd_target
```

Out[109]:

SeriousDlqin2yrs	0	1
NumberOfTimes90DaysLate		
0	135108	6554
1	3478	1765
2	779	776
3	282	385
4	96	195
5	48	83
6	32	48
7	7	31
8	6	15
9	5	14
10	3	5
11	2	3
12	1	1
13	2	2
14	1	1
15	2	0
17	0	1
96	1	4
98	121	143

In [110]:

```
cross_tab_90dpd_target_percent=cross_tab_90dpd_target.astype(float).div(cross_tab_90dpd
_target.sum(axis=1), axis=0)
round(cross_tab_90dpd_target_percent,2)
```

Out[110]:

SeriousDlqin2yrs	0	1
NumberOfTimes90DaysLate		
0	0.95	0.05
1	0.66	0.34
2	0.50	0.50
3	0.42	0.58
4	0.33	0.67
5	0.37	0.63
6	0.40	0.60
7	0.18	0.82
8	0.29	0.71
9	0.26	0.74
10	0.38	0.62
11	0.40	0.60
12	0.50	0.50
13	0.50	0.50
14	0.50	0.50
15	1.00	0.00
17	0.00	1.00
96	0.20	0.80
98	0.46	0.54

In [111]:

```
loans['num_90_dpd_new']=loans['NumberOfTimes90DaysLate']
loans['num_90_dpd_new'][loans['num_90_dpd_new']>12]=3
loans['num_90_dpd_new']

loans['num_90_dpd_new'].value_counts(sort=False)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[111]:

```
0     141662
1       5243
2       1555
3        945
4        291
5        131
6         80
7         38
8         21
9         19
10         8
11         5
12         2
Name: num_90_dpd_new, dtype: int64
```

Variable9 = "NumberRealEstateLoansOrLines"

This variable is similar to Number of Open Credit Lines and Loans. Directly perform cleaning of this varibale with very few lines of code. No need of visualization

In [112]:

```
loans["NumberRealEstateLoansOrLines"].describe()
```

Out[112]:

```
count    150000.000000
mean          1.018240
std           1.129771
min           0.000000
25%           0.000000
50%           1.000000
75%           2.000000
max          54.000000
Name: NumberRealEstateLoansOrLines, dtype: float64
```

In [113]:

```
loans["NumberRealEstateLoansOrLines"].quantile([0,0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.
08,0.09,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.91,0.92,0.93,0.94,0.95,0.96,0.97,0.98,0.9
9,1])
```

Out[113]:

```
0.00     0.0
0.01     0.0
0.02     0.0
0.03     0.0
0.04     0.0
0.05     0.0
0.06     0.0
0.07     0.0
0.08     0.0
0.09     0.0
0.10     0.0
0.20     0.0
0.30     0.0
0.40     1.0
0.50     1.0
0.60     1.0
0.70     1.0
0.80     2.0
0.90     2.0
0.91     2.0
0.92     2.0
0.93     2.0
0.94     3.0
0.95     3.0
0.96     3.0
0.97     3.0
0.98     4.0
0.99     4.0
1.00    54.0
Name: NumberRealEstateLoansOrLines, dtype: float64
```

In [114]:

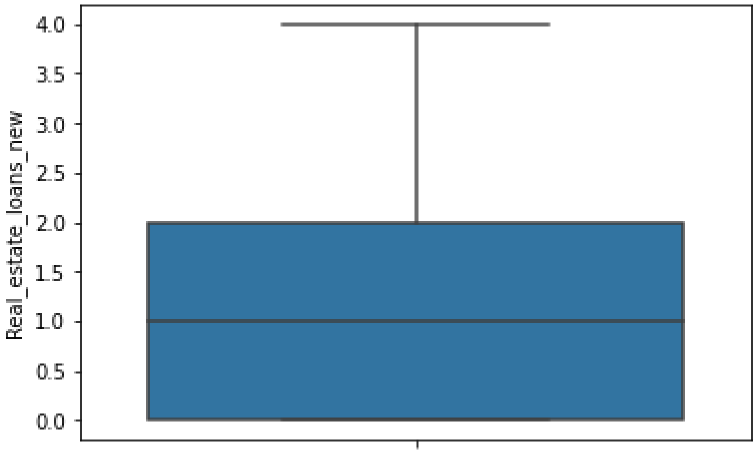
```
loans["Real_estate_loans_new"]=loans["NumberRealEstateLoansOrLines"]
loans["Real_estate_loans_new"][loans["Real_estate_loans_new"]>4]=loans["NumberRealEstat
eLoansOrLines"].median()
sns.boxplot(y=loans["Real_estate_loans_new"])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b7ee7438>
```



Variable10 = "NumberOfTime60-89DaysPastDueNotWorse"

This variable is similar to 30_59_DPD and 90_DPD. Directly perform cleaning of this varibale with very few lines of code. No need of visualization

In [115]:

```
cross_tab_60dpd_target=pd.crosstab(loans['NumberOfTime60-89DaysPastDueNotWorse'],loans[
'SeriousDlqin2yrs'])
cross_tab_60dpd_target
```

Out[115]:

SeriousDlqin2yrs	0	1
NumberOfTime60-89DaysPastDueNotWorse		
0	135140	7256
1	3954	1777
2	557	561
3	138	180
4	40	65
5	13	21
6	4	12
7	4	5
8	1	1
9	1	0
11	0	1
96	1	4
98	121	143

In [116]:

```
cross_tab_60dpd_target_percent=cross_tab_60dpd_target.astype(float).div(cross_tab_60dpd
_target.sum(axis=1), axis=0)
round(cross_tab_60dpd_target_percent,2)
```

Out[116]:

SeriousDlqin2yrs	0	1
NumberOfTime60-89DaysPastDueNotWorse		
0	0.95	0.05
1	0.69	0.31
2	0.50	0.50
3	0.43	0.57
4	0.38	0.62
5	0.38	0.62
6	0.25	0.75
7	0.44	0.56
8	0.50	0.50
9	1.00	0.00
11	0.00	1.00
96	0.20	0.80
98	0.46	0.54

In [117]:

```
loans['num_60_dpd_new']=loans['NumberOfTime60-89DaysPastDueNotWorse']
loans['num_60_dpd_new'][loans['num_60_dpd_new']>12]=3
loans['num_60_dpd_new']

loans['num_60_dpd_new'].value_counts(sort=False)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[117]:

```
0     142396
1       5731
2       1118
3        587
4        105
5         34
6         16
7          9
8          2
9          1
11         1
Name: num_60_dpd_new, dtype: int64
```

Variable11 = "NumberOfDependents"

This variable is similar to Number of Open Credit Lines and Real estate loans. Directly perform cleaning of this varibale with very few lines of code. No need of visualization

In [118]:

```
loans["NumberOfDependents"].describe()
```

Out[118]:

```
count    146076.000000
mean          0.757222
std           1.115086
min           0.000000
25%           0.000000
50%           0.000000
75%           1.000000
max          20.000000
Name: NumberOfDependents, dtype: float64
```

In [119]:

```
loans["NumberOfDependents"].quantile([0,0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,0.91,0.92,0.93,0.94,0.95,0.96,0.97,0.98,0.99,1])
```

Out[119]:

```
0.00     0.0
0.01     0.0
0.02     0.0
0.03     0.0
0.04     0.0
0.05     0.0
0.06     0.0
0.07     0.0
0.08     0.0
0.09     0.0
0.10     0.0
0.20     0.0
0.30     0.0
0.40     0.0
0.50     0.0
0.60     1.0
0.70     1.0
0.80     2.0
0.90     2.0
0.91     3.0
0.92     3.0
0.93     3.0
0.94     3.0
0.95     3.0
0.96     3.0
0.97     3.0
0.98     4.0
0.99     4.0
1.00    20.0
Name: NumberOfDependents, dtype: float64
```

In [120]:

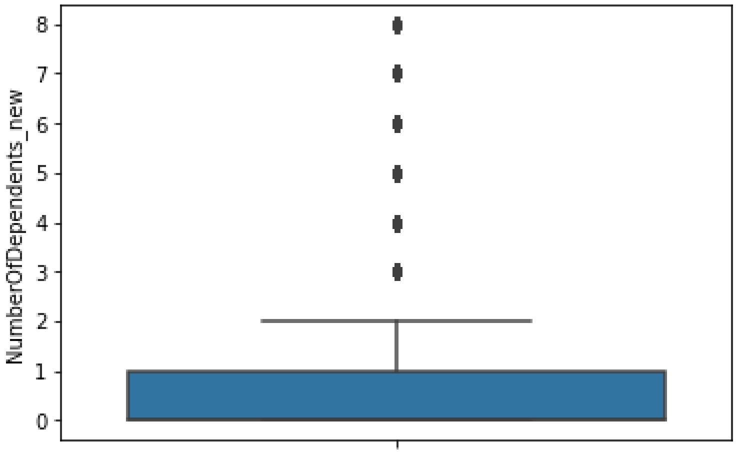
```
loans["NumberOfDependents_new"]=loans["NumberOfDependents"]
loans["NumberOfDependents_new"][loans["NumberOfDependents_new"]>8]=loans["NumberOfDependents"].median()
sns.boxplot(y=loans["NumberOfDependents_new"])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Out[120]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b7ed7d68>
```



Write down your final findings and treatment in two lins.

SeriousDlqin2yrs ==> Findings and Treatmet

In [120]:

RevolvingUtilizationOfUnsecuredLines ==> Findings and Treatmet

In [120]:

age ==> Findings and Treatmet

In [120]:

NumberOfTime30-59DaysPastDueNotWorse ==> Findings and Treatmet

In [120]:

Debt Ratio ==> Findings and Treatmet

In [120]:

Monthly Income ==> Findings and Treatmet

In [120]:

NumberOfOpenCreditLines and Loans ==> Findings and Treatmet

In [120]:

NumberOfTimes90DaysLate ==> Findings and Treatmet

In [120]:

NumberRealEstateLoansOrLines ==> Findings and Treatmet

In [120]:

NumberOfTime60-89DaysPastDueNotWorse ==> Findings and Treatmet

In [120]:

NumberOfDependents ==> Findings and Treatmet

In [120]: