In [13]:
```
import pandas as pd
import numpy  as np

ld = pd.read_csv("C:/Users/RAM1/Desktop/DSP/loandata.csv")
```

In [14]:
```
ld["Amount.Requested"]= pd.to_numeric(ld["Amount.Requested"],errors="coerse")
```

In [15]:
```
#Help function
help(pd.to_numeric)
```

```
Help on function to_numeric in module pandas.core.tools.numeric:

to_numeric(arg, errors='raise', downcast=None)
    Convert argument to a numeric type.

    Parameters
    ----------
    arg : list, tuple, 1-d array, or Series
    errors : {'ignore', 'raise', 'coerce'}, default 'raise'
        - If 'raise', then invalid parsing will raise an exception
        - If 'coerce', then invalid parsing will be set as NaN
        - If 'ignore', then invalid parsing will return the input
    downcast : {'integer', 'signed', 'unsigned', 'float'} , default None
        If not None, and if the data has been successfully cast to a
        numerical dtype (or if the data was numeric to begin with),
        downcast that resulting data to the smallest numerical dtype
        possible according to the following rules:

        - 'integer' or 'signed': smallest signed int dtype (min.: np.int8)
        - 'unsigned': smallest unsigned int dtype (min.: np.uint8)
        - 'float': smallest float dtype (min.: np.float32)

        As this behaviour is separate from the core conversion to
        numeric values, any errors raised during the downcasting
        will be surfaced regardless of the value of the 'errors' input.

        In addition, downcasting will only occur if the size
        of the resulting data's dtype is strictly larger than
        the dtype it is to be cast to, so if none of the dtypes
        checked satisfy that specification, no downcasting will be
        performed on the data.

        .. versionadded:: 0.19.0

    Returns
    -------
    ret : numeric if parsing succeeded.
        Return type depends on input.  Series if Series, otherwise ndarray

    Examples
    --------
    Take separate series and convert to numeric, coercing when told to

    >>> import pandas as pd
    >>> s = pd.Series(['1.0', '2', -3])
    >>> pd.to_numeric(s)
    0    1.0
    1    2.0
    2   -3.0
    dtype: float64
    >>> pd.to_numeric(s, downcast='float')
    0    1.0
    1    2.0
```

```
2   -3.0
dtype: float32
>>> pd.to_numeric(s, downcast='signed')
0    1
1    2
2   -3
dtype: int8
>>> s = pd.Series(['apple', '1.0', '2', -3])
>>> pd.to_numeric(s, errors='ignore')
0    apple
1      1.0
2        2
3       -3
dtype: object
>>> pd.to_numeric(s, errors='coerce')
0    NaN
1    1.0
2    2.0
3   -3.0
dtype: float64

See also
--------
pandas.DataFrame.astype : Cast argument to a specified dtype.
pandas.to_datetime : Convert argument to datetime.
pandas.to_timedelta : Convert argument to timedelta.
numpy.ndarray.astype : Cast a numpy array to a specified type.
```

In [16]:
```
ld["Amount.Requested"]= pd.to_numeric(ld["Amount.Requested"],errors="ignore")
ld.head()
```

Out[16]:

| | ID | Amount.Requested | Amount.Funded.By.Investors | Interest.Rate | Loan.Length | Loan.Pu |
|---|---|---|---|---|---|---|
| **0** | 81174.0 | 20000.0 | 20000 | 8.90% | 36 months | debt_consoli |
| **1** | 99592.0 | 19200.0 | 19200 | 12.12% | 36 months | debt_consoli |
| **2** | 80059.0 | 35000.0 | 35000 | 21.98% | 60 months | debt_consoli |
| **3** | 15825.0 | 10000.0 | 9975 | 9.99% | 36 months | debt_consoli |
| **4** | 33182.0 | 12000.0 | 12000 | 11.71% | 36 months | credi |

In [17]:
```
ld["Interest.Rate"]= ld["Interest.Rate"].str.replace("%","")
```

In [18]:
```
#To check data types
ld.dtypes
```

Out[18]:
```
ID                                float64
Amount.Requested                  float64
Amount.Funded.By.Investors         object
Interest.Rate                      object
Loan.Length                        object
Loan.Purpose                       object
Debt.To.Income.Ratio               object
State                              object
Home.Ownership                     object
Monthly.Income                    float64
FICO.Range                         object
Open.CREDIT.Lines                  object
Revolving.CREDIT.Balance           object
Inquiries.in.the.Last.6.Months    float64
Employment.Length                  object
dtype: object
```

In [19]:
```
ld["Interest.Rate"].dtype
ld["Interest.Rate"]=pd.to_numeric(ld["Interest.Rate"],errors="coerce")
```

In [20]:
```

ld["Debt.To.Income.Ratio"]=ld["Debt.To.Income.Ratio"].str.replace("%","")
ld["Debt.To.Income.Ratio"]=pd.to_numeric(ld["Debt.To.Income.Ratio"],errors="coerc
ld["Debt.To.Income.Ratio"].dtypes
```

Out[20]:
```
dtype('float64')
```

In [ ]:

In [21]:
```
ld["Loan.Length"].value_counts()
```

Out[21]:
```
36 months    1950
60 months     548
.               1
Name: Loan.Length, dtype: int64
```

In [22]:
```
ld["Loan.Length"]= ld["Loan.Length"].str.replace("months","")
ld["Loan.Length"].value_counts()
ld["Loan.Length"]=pd.to_numeric(ld["Loan.Length"],errors="coerse")
ld.head()
```

Out[22]:

| | ID | Amount.Requested | Amount.Funded.By.Investors | Interest.Rate | Loan.Length | Loan.Pu |
|---|---|---|---|---|---|---|
| **0** | 81174.0 | 20000.0 | 20000 | 8.90 | 36.0 | debt_consoli |
| **1** | 99592.0 | 19200.0 | 19200 | 12.12 | 36.0 | debt_consoli |
| **2** | 80059.0 | 35000.0 | 35000 | 21.98 | 60.0 | debt_consoli |
| **3** | 15825.0 | 10000.0 | 9975 | 9.99 | 36.0 | debt_consoli |
| **4** | 33182.0 | 12000.0 | 12000 | 11.71 | 36.0 | credi |

In [23]:
```
ld["Loan.Length"]=pd.to_numeric(ld["Loan.Length"],errors="ignore")
ld["Loan.Length"].dtypes
```

Out[23]:
```
dtype('float64')
```

In [24]:
```
ld["Amount.Funded.By.Investors"]=pd.to_numeric(ld["Amount.Funded.By.Investors"],e
ld.dtypes
```

Out[24]:
```
ID                                float64
Amount.Requested                  float64
Amount.Funded.By.Investors        float64
Interest.Rate                     float64
Loan.Length                       float64
Loan.Purpose                       object
Debt.To.Income.Ratio              float64
State                              object
Home.Ownership                     object
Monthly.Income                    float64
FICO.Range                         object
Open.CREDIT.Lines                  object
Revolving.CREDIT.Balance           object
Inquiries.in.the.Last.6.Months    float64
Employment.Length                  object
dtype: object
```

In [ ]:

In [25]:
```
ld["F1"]=ld["FICO.Range"].str[:3]
ld["F1"]=pd.to_numeric(ld["F1"],errors="coerse")
```

In [26]:
```
ld["F2"]=ld["FICO.Range"].str[4:7]
ld["F2"]=pd.to_numeric(ld["F1"],errors="coerse")
```

In [27]:
```
ld.dtypes
```

Out[27]:
```
ID                                float64
Amount.Requested                  float64
Amount.Funded.By.Investors        float64
Interest.Rate                     float64
Loan.Length                       float64
Loan.Purpose                       object
Debt.To.Income.Ratio              float64
State                              object
Home.Ownership                     object
Monthly.Income                    float64
FICO.Range                         object
Open.CREDIT.Lines                  object
Revolving.CREDIT.Balance           object
Inquiries.in.the.Last.6.Months    float64
Employment.Length                  object
F1                                  int64
F2                                  int64
dtype: object
```

In [28]:
```
ld["F3"]=(ld["F1"]+ld["F2"])/2
ld.head()
```

Out[28]:

| | ID | Amount.Requested | Amount.Funded.By.Investors | Interest.Rate | Loan.Length | Loan.Pu |
|---|---|---|---|---|---|---|
| **0** | 81174.0 | 20000.0 | 20000.0 | 8.90 | 36.0 | debt_consoli |
| **1** | 99592.0 | 19200.0 | 19200.0 | 12.12 | 36.0 | debt_consoli |
| **2** | 80059.0 | 35000.0 | 35000.0 | 21.98 | 60.0 | debt_consoli |
| **3** | 15825.0 | 10000.0 | 9975.0 | 9.99 | 36.0 | debt_consoli |
| **4** | 33182.0 | 12000.0 | 12000.0 | 11.71 | 36.0 | credi |

In [ ]:

In [ ]:

In [29]:
```
ld=ld.drop(["F1","F2","FICO.Range"],axis=1)
```

In [30]:
```
ld.dtypes
```

Out[30]:
```
ID                                float64
Amount.Requested                  float64
Amount.Funded.By.Investors        float64
Interest.Rate                     float64
Loan.Length                       float64
Loan.Purpose                       object
Debt.To.Income.Ratio              float64
State                              object
Home.Ownership                     object
Monthly.Income                    float64
Open.CREDIT.Lines                  object
Revolving.CREDIT.Balance           object
Inquiries.in.the.Last.6.Months    float64
Employment.Length                  object
F3                                float64
dtype: object
```

In [ ]:

In [ ]:

In [31]:
```
ld["Employment.Length"]=ld["Employment.Length"].str.replace("years","")
ld["Employment.Length"]=ld["Employment.Length"].str.replace("year","")
ld["Employment.Length"]=ld["Employment.Length"].str.replace("< 1","0")
ld["Employment.Length"]=ld["Employment.Length"].str.replace("+","")
ld.head()
ld["Employment.Length"].value_counts()
```

Out[31]:
```
10     653
0      249
2      243
3      235
5      202
4      191
1      177
6      163
7      127
8      108
9       72
.        2
Name: Employment.Length, dtype: int64
```

In [32]:
```
ld["Employment.Length"]=pd.to_numeric(ld["Employment.Length"],errors="coerse")
```

In [33]:
```
ld.dtypes
```

Out[33]:
```
ID                                float64
Amount.Requested                  float64
Amount.Funded.By.Investors        float64
Interest.Rate                     float64
Loan.Length                       float64
Loan.Purpose                       object
Debt.To.Income.Ratio              float64
State                              object
Home.Ownership                     object
Monthly.Income                    float64
Open.CREDIT.Lines                  object
Revolving.CREDIT.Balance           object
Inquiries.in.the.Last.6.Months    float64
Employment.Length                 float64
F3                                float64
dtype: object
```

In [34]:
```
ld["Open.CREDIT.Lines"]=pd.to_numeric(ld["Open.CREDIT.Lines"],errors="coerse")
```

In [35]:
```
ld["Home.Ownership"].value_counts()
```

Out[35]:
```
MORTGAGE    1147
RENT        1146
OWN          200
OTHER          5
NONE           1
Name: Home.Ownership, dtype: int64
```

In [36]:
```
dummy=pd.get_dummies(ld["Home.Ownership"])
```

In [37]:
```
dummy=dummy.drop(["NONE"],axis=1)
```

In [ ]:
```

```

In [38]:
```
dummy["MORTGAGE"].value_counts()
```

Out[38]:
```
0    1353
1    1147
Name: MORTGAGE, dtype: int64
```

In [39]:
```
ld=pd.concat([ld,dummy],axis=1)
```

In [40]:
```
ld.head()
```

Out[40]:

| | ID | Amount.Requested | Amount.Funded.By.Investors | Interest.Rate | Loan.Length | Loan.Pu |
|---|---|---|---|---|---|---|
| **0** | 81174.0 | 20000.0 | 20000.0 | 8.90 | 36.0 | debt_consoli |
| **1** | 99592.0 | 19200.0 | 19200.0 | 12.12 | 36.0 | debt_consoli |
| **2** | 80059.0 | 35000.0 | 35000.0 | 21.98 | 60.0 | debt_consoli |
| **3** | 15825.0 | 10000.0 | 9975.0 | 9.99 | 36.0 | debt_consoli |
| **4** | 33182.0 | 12000.0 | 12000.0 | 11.71 | 36.0 | credi |

In [41]:
```
ld=ld.drop(["Home.Ownership"],axis=1)
```

In [42]:
```
ld.head()
```

Out[42]:

| | ID | Amount.Requested | Amount.Funded.By.Investors | Interest.Rate | Loan.Length | Loan.Pu |
|---|---|---|---|---|---|---|
| **0** | 81174.0 | 20000.0 | 20000.0 | 8.90 | 36.0 | debt_consoli |
| **1** | 99592.0 | 19200.0 | 19200.0 | 12.12 | 36.0 | debt_consoli |
| **2** | 80059.0 | 35000.0 | 35000.0 | 21.98 | 60.0 | debt_consoli |
| **3** | 15825.0 | 10000.0 | 9975.0 | 9.99 | 36.0 | debt_consoli |
| **4** | 33182.0 | 12000.0 | 12000.0 | 11.71 | 36.0 | credi |

In [43]:
```
ld["State"].value_counts()
```

Out[43]:
```
CA    433
NY    255
TX    174
FL    169
IL    101
GA     97
PA     96
NJ     94
VA     78
MA     73
OH     71
MD     68
NC     64
CO     61
WA     58
CT     50
AZ     46
MI     45
AL     38
MN     38
MO     33
NV     32
OR     30
SC     28
WI     26
KY     23
LA     22
KS     21
OK     21
UT     16
NH     15
RI     15
WV     14
AR     13
NM     13
HI     12
DC     11
AK     11
DE      8
MT      7
VT      5
SD      4
WY      4
IN      3
MS      1
IA      1
.       1
Name: State, dtype: int64
```

In [44]:
```
ld=ld.drop(["State"],axis=1)
ld.head()
```

Out[44]:

| | ID | Amount.Requested | Amount.Funded.By.Investors | Interest.Rate | Loan.Length | Loan.Pu |
|---|---|---|---|---|---|---|
| **0** | 81174.0 | 20000.0 | 20000.0 | 8.90 | 36.0 | debt_consoli |
| **1** | 99592.0 | 19200.0 | 19200.0 | 12.12 | 36.0 | debt_consoli |
| **2** | 80059.0 | 35000.0 | 35000.0 | 21.98 | 60.0 | debt_consoli |
| **3** | 15825.0 | 10000.0 | 9975.0 | 9.99 | 36.0 | debt_consoli |
| **4** | 33182.0 | 12000.0 | 12000.0 | 11.71 | 36.0 | credi |

In [45]:
```
ld.dtypes
```

Out[45]:
```
ID                              float64
Amount.Requested                float64
Amount.Funded.By.Investors      float64
Interest.Rate                   float64
Loan.Length                     float64
Loan.Purpose                     object
Debt.To.Income.Ratio            float64
Monthly.Income                  float64
Open.CREDIT.Lines               float64
Revolving.CREDIT.Balance         object
Inquiries.in.the.Last.6.Months  float64
Employment.Length               float64
F3                              float64
MORTGAGE                          uint8
OTHER                             uint8
OWN                               uint8
RENT                              uint8
dtype: object
```

In [46]:
```
ld["Loan.Purpose"].value_counts()
```

Out[46]:
```
debt_consolidation    1307
credit_card            444
other                  200
home_improvement       152
major_purchase         101
small_business          87
car                     50
wedding                 39
medical                 30
moving                  29
vacation                21
house                   20
educational             15
renewable_energy         4
Name: Loan.Purpose, dtype: int64
```

In [47]:
```
dummy2=pd.get_dummies(ld["Loan.Purpose"])
```

In [48]:
```
dummy2.head()
```

Out[48]:

| | car | credit_card | debt_consolidation | educational | home_improvement | house | major_purchase | m |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| **1** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| **2** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| **3** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | |
| **4** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |

In [ ]:

In [63]:
```
ld1 = pd.concat([ld,dummy2[["debt_consolidation","credit_card","other","home_impr
```

In [64]:
```
ld1["Revolving.CREDIT.Balance"]=pd.to_numeric(ld["Revolving.CREDIT.Balance"],erro
```

In [65]:
```
ld1
```

Out[65]:

| | ID | Amount.Requested | Amount.Funded.By.Investors | Interest.Rate | Loan.Length | Lo |
|---|---|---|---|---|---|---|
| **0** | 81174.0 | 20000.0 | 20000.00 | 8.90 | 36.0 | debt_c |
| **1** | 99592.0 | 19200.0 | 19200.00 | 12.12 | 36.0 | debt_c |
| **2** | 80059.0 | 35000.0 | 35000.00 | 21.98 | 60.0 | debt_c |
| **3** | 15825.0 | 10000.0 | 9975.00 | 9.99 | 36.0 | debt_c |
| **4** | 33182.0 | 12000.0 | 12000.00 | 11.71 | 36.0 | |
| **5** | 62403.0 | 6000.0 | 6000.00 | 15.31 | 36.0 | |
| **6** | 48808.0 | 10000.0 | 10000.00 | 7.90 | 36.0 | debt_c |
| **7** | 22090.0 | 33500.0 | 33450.00 | 17.14 | 60.0 | |
| **8** | 76404.0 | 14675.0 | 14675.00 | 14.33 | 36.0 | |
| **9** | 15867.0 | NaN | 7000.00 | 6.91 | 36.0 | |
| **10** | 94971.0 | 2000.0 | 2000.00 | 19.72 | 36.0 | |
| **11** | 36911.0 | 10625.0 | 10625.00 | 14.27 | 36.0 | debt_c |
| **12** | 41200.0 | 28000.0 | 27975.00 | 21.67 | 60.0 | debt_c |
| **13** | 83869.0 | 35000.0 | 34950.00 | 8.90 | 36.0 | debt_c |
| **14** | 53853.0 | 9600.0 | 9600.00 | 7.62 | 36.0 | debt_c |
| **15** | 21399.0 | 25000.0 | 24975.00 | 15.65 | 60.0 | debt_c |
| **16** | 62127.0 | 10000.0 | 10000.00 | 12.12 | 36.0 | debt_c |
| **17** | 23446.0 | 14000.0 | 13900.25 | 10.37 | 60.0 | debt_c |
| **18** | 44987.0 | 10000.0 | 10000.00 | 9.76 | 36.0 | |
| **19** | 17977.0 | 5200.0 | 5175.00 | 9.99 | 60.0 | debt_c |
| **20** | 86099.0 | 22000.0 | 21975.00 | 21.98 | 36.0 | debt_c |
| **21** | 99483.0 | 30000.0 | 30000.00 | 19.05 | 60.0 | |
| **22** | 28798.0 | 6500.0 | 6500.00 | 17.99 | 60.0 | |
| **23** | 24168.0 | 17400.0 | 17400.00 | 11.99 | 36.0 | |
| **24** | 10356.0 | 4000.0 | 4000.00 | 16.82 | 60.0 | |
| **25** | 46027.0 | 7200.0 | 7200.00 | 7.90 | 36.0 | debt_c |
| **26** | 2238.0 | 8000.0 | 8000.00 | 14.42 | 36.0 | debt_c |
| **27** | 65278.0 | 8000.0 | 8000.00 | 15.31 | 36.0 | debt_c |
| **28** | 4227.0 | 3000.0 | NaN | 8.59 | 36.0 | |
| **29** | 50182.0 | 14500.0 | 14500.00 | 7.90 | 36.0 | debt_c |
| **...** | ... | ... | ... | ... | ... | |
| **2470** | 84265.0 | 20000.0 | 20000.00 | 22.95 | 60.0 | debt_c |
| **2471** | 80231.0 | 19000.0 | 19000.00 | 7.90 | 36.0 | debt_c |
| **2472** | 49533.0 | 17300.0 | 17250.00 | 22.45 | 60.0 | |

| | ID | Amount.Requested | Amount.Funded.By.Investors | Interest.Rate | Loan.Length | Lo |
|---|---|---|---|---|---|---|
| **2473** | 102514.0 | 7000.0 | 711.54 | 15.13 | 36.0 | maj |
| **2474** | 78618.0 | 7200.0 | 7200.00 | 18.75 | 36.0 | debt_c |
| **2475** | 86953.0 | 10000.0 | 10000.00 | 14.09 | 36.0 | maj |
| **2476** | 80129.0 | 4000.0 | 3925.00 | 14.09 | 36.0 | |
| **2477** | 85216.0 | 17500.0 | 17500.00 | 8.90 | 36.0 | debt_c |
| **2478** | 38247.0 | 20000.0 | 20000.00 | 11.71 | 36.0 | |
| **2479** | 91245.0 | 16200.0 | 16200.00 | 15.80 | 60.0 | debt_c |
| **2480** | 53041.0 | 10000.0 | 10000.00 | 6.03 | 36.0 | sm |
| **2481** | 63051.0 | 27000.0 | 27000.00 | 6.62 | 36.0 | debt_c |
| **2482** | 14446.0 | 4500.0 | 4475.00 | 7.51 | 36.0 | sm |
| **2483** | 68628.0 | NaN | 15875.00 | 14.33 | 36.0 | sm |
| **2484** | 98758.0 | 15000.0 | 15000.00 | 10.16 | 36.0 | |
| **2485** | 13070.0 | 25000.0 | 24950.00 | 10.75 | 36.0 | debt_c |
| **2486** | 45836.0 | 7000.0 | 7000.00 | 17.27 | 36.0 | |
| **2487** | 52330.0 | 15000.0 | 15000.00 | 19.99 | 36.0 | |
| **2488** | 48243.0 | 17000.0 | 17000.00 | 15.81 | 36.0 | debt_c |
| **2489** | 63256.0 | 19075.0 | 19075.00 | 18.75 | 36.0 | debt_c |
| **2490** | 42124.0 | 10000.0 | 10000.00 | 11.71 | 36.0 | debt_c |
| **2491** | 78043.0 | 8475.0 | 8475.00 | 7.62 | 36.0 | debt_c |
| **2492** | 925.0 | 6400.0 | 6350.00 | 10.08 | 36.0 | debt_c |
| **2493** | 74047.0 | 30000.0 | 30000.00 | 23.28 | 60.0 | |
| **2494** | 49957.0 | 24000.0 | 23975.00 | 14.65 | 36.0 | debt_c |
| **2495** | 23735.0 | 30000.0 | 29950.00 | 16.77 | 60.0 | debt_c |
| **2496** | 65882.0 | 16000.0 | 16000.00 | 14.09 | 60.0 | home_i |
| **2497** | 55610.0 | 10000.0 | 10000.00 | 13.99 | 36.0 | debt_c |
| **2498** | 38576.0 | 6000.0 | 6000.00 | 12.42 | 36.0 | maj |
| **2499** | 3116.0 | 9000.0 | 5242.75 | 13.79 | 36.0 | debt_c |

2500 rows × 21 columns

In [66]:
```
ld1=ld1.drop(["Loan.Purpose"],axis=1)
```

In [67]:
```
ld1.dtypes
```

Out[67]:
```
ID                                 float64
Amount.Requested                   float64
Amount.Funded.By.Investors         float64
Interest.Rate                      float64
Loan.Length                        float64
Debt.To.Income.Ratio               float64
Monthly.Income                     float64
Open.CREDIT.Lines                  float64
Revolving.CREDIT.Balance           float64
Inquiries.in.the.Last.6.Months     float64
Employment.Length                  float64
F3                                 float64
MORTGAGE                             uint8
OTHER                                uint8
OWN                                  uint8
RENT                                 uint8
debt_consolidation                   uint8
credit_card                          uint8
other                                uint8
home_improvement                     uint8
dtype: object
```

In [78]:
```
ld1=ld.dropna()
```

In [69]:
```
#To deleting the NaN values from the table
ld1=ld1.dropna()
```

In [79]:
```
ld1.isnull().sum()
```

Out[79]:
```
ID                                 0
Amount.Requested                   0
Amount.Funded.By.Investors         0
Interest.Rate                      0
Loan.Length                        0
Loan.Purpose                       0
Debt.To.Income.Ratio               0
Monthly.Income                     0
Open.CREDIT.Lines                  0
Revolving.CREDIT.Balance           0
Inquiries.in.the.Last.6.Months     0
Employment.Length                  0
F3                                 0
MORTGAGE                           0
OTHER                              0
OWN                                0
RENT                               0
dtype: int64
```

In [80]:
```
ld1.dtypes
ld1=ld1.drop(["Loan.Purpose","Revolving.CREDIT.Balance"],axis=1)
```

In [72]:
```
#sklearn module -> train_test_split

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [82]:
```
#ld1=ld1.drop(["z"],axis=1)
#ld1.dtypes
```

In [83]:
```
ld1.dtypes
```

Out[83]:
```
ID                                float64
Amount.Requested                  float64
Amount.Funded.By.Investors        float64
Interest.Rate                     float64
Loan.Length                       float64
Debt.To.Income.Ratio              float64
Monthly.Income                    float64
Open.CREDIT.Lines                 float64
Inquiries.in.the.Last.6.Months    float64
Employment.Length                 float64
F3                                float64
MORTGAGE                            uint8
OTHER                               uint8
OWN                                 uint8
RENT                                uint8
dtype: object
```

In [ ]:
```

```

In [84]:
```
x= ld1.drop(["Interest.Rate","ID"],axis=1)
y= ld1["Interest.Rate"]
```

In [85]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 1/3,random_state=0)
```

In [ ]:
```
#xtrain,xtest,ytrain,ytest = train_test_split(x,y,train_size = 1/3,random_state=0,
```

In [86]:
```
LR = LinearRegression() #Storing Linear Regression function into var to use multi
```

In [87]:
```
LR.fit(xtrain,ytrain)
```

Out[87]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [88]:
```
yprediction=LR.predict(xtest)
```

In [89]:
```
ypred=LR.predict(xtest)
```

In [90]:
```
ypred
```

Out[90]:
```
array([14.48808559, 13.48765184, 17.95978345,  7.69693303, 15.04468916,
       10.7443494 , 12.06251739, 13.68461121, 18.87113524,  6.15254873,
       14.95691383,  9.69631468, 15.75210821, 14.92458522, 10.77675455,
        6.64758602, 14.5911197 ,  7.32208395,  8.17609713, 13.83092889,
       13.04124941, 14.22692924, 14.94373201, 15.2695556 , 15.30337707,
       10.86110868, 13.83263192, 14.42597651, 10.29514321, 12.3321702 ,
       15.75005724, 12.38568244, 13.39344383,  9.60096016, 13.57178983,
        8.93484071,  8.56971593, 18.20930142, 14.04724334,  9.69062188,
       11.23430867, 11.70499626, 14.22091174, 10.28738897, 14.32986162,
       11.06558764, 13.6550816 , 17.19921243, 12.74083446, 13.00456066,
       16.51177296,  6.23696289, 10.21827662, 13.60543981,  8.45776004,
       15.67230686, 15.57554332, 13.25274415, 15.81142866, 14.16383374,
        4.04024222,  6.83302199,  7.30747011, 20.32637258,  9.21149876,
       15.27182108, 13.93541695,  6.41700887, 13.31524881,  9.46180807,
       13.44142013, 16.09153691,  7.11476954, 11.43658688, 16.07464453,
        5.73495891, 17.80492823, 12.03393791, 17.48818261, 10.65885018,
       14.18917795, 18.14214117,  6.8229581 ,  7.70154364, 15.40516748,
       10.67267572, 10.1715663 , 17.32008187, 15.87422258, 11.3321291 ,
       16.23365451, 13.5845811 , 15.20765763, 18.98569338, 13.33292506,
```

In [ ]:

In [ ]: