dashboard

July 27, 2022

```
[1]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
```

```
[2]: df_tn = pd.read_csv("train_mz1.csv")
     df_tn.head()
```

```
[2]:       UID  BLOCKID  SUMLEVEL  COUNTYID  STATEID        state state_ab  \
     0  267822      NaN       140        53       36     New York       NY
     1  246444      NaN       140       141       18      Indiana       IN
     2  245683      NaN       140        63       18      Indiana       IN
     3  279653      NaN       140       127       72  Puerto Rico       PR
     4  247218      NaN       140       161       20       Kansas       KS

             city           place   type  … female_age_mean  female_age_median  \
     0   Hamilton        Hamilton   City  …        44.48629           45.33333
     1 South Bend        Roseland   City  …        36.48391           37.58333
     2   Danville        Danville   City  …        42.15810           42.83333
     3   San Juan        Guaynabo  Urban  …        47.77526           50.58333
     4  Manhattan  Manhattan City   City  …        24.17693           21.58333

        female_age_stdev  female_age_sample_weight  female_age_samples  pct_own  \
     0          22.51276                 685.33845              2618.0  0.79046
     1          23.43353                 267.23367              1284.0  0.52483
     2          23.94119                 707.01963              3238.0  0.85331
     3          24.32015                 362.20193              1559.0  0.65037
     4          11.10484                1854.48652              3051.0  0.13046

        married  married_snp  separated  divorced
     0  0.57851      0.01882    0.01240   0.08770
     1  0.34886      0.01426    0.01426   0.09030
     2  0.64745      0.02830    0.01607   0.10657
     3  0.47257      0.02021    0.02021   0.10106
     4  0.12356      0.00000    0.00000   0.03109

     [5 rows x 80 columns]
```

Missing value treatment

```
[3]: df_te = pd.read_csv("test_mz.csv")
     df_te.head()
```

```
[3]:       UID  BLOCKID  SUMLEVEL  COUNTYID  STATEID        state state_ab  \
     0  255504      NaN       140       163       26     Michigan       MI
     1  252676      NaN       140         1       23        Maine       ME
     2  276314      NaN       140        15       42  Pennsylvania       PA
     3  248614      NaN       140       231       21     Kentucky       KY
     4  286865      NaN       140       355       48        Texas       TX

                  city                 place     type  … female_age_mean  \
     0         Detroit  Dearborn Heights City      CDP  …        34.78682
     1          Auburn           Auburn City     City  …        44.23451
     2       Pine City            Millerton  Borough  …        41.62426
     3      Monticello       Monticello City     City  …        44.81200
     4  Corpus Christi                 Edroy     Town  …        40.66618

        female_age_median  female_age_stdev  female_age_sample_weight  \
     0           33.75000          21.58531                 416.48097
     1           46.66667          22.37036                 532.03505
     2           44.50000          22.86213                 453.11959
     3           48.00000          21.03155                 263.94320
     4           42.66667          21.30900                 709.90829

        female_age_samples  pct_own  married  married_snp  separated  divorced
     0              1938.0  0.70252  0.28217      0.05910    0.03813   0.14299
     1              1950.0  0.85128  0.64221      0.02338    0.00000   0.13377
     2              1879.0  0.81897  0.59961      0.01746    0.01358   0.10026
     3              1081.0  0.84609  0.56953      0.05492    0.04694   0.12489
     4              2956.0  0.79077  0.57620      0.01726    0.00588   0.16379

     [5 rows x 80 columns]
```

```
[4]:  ####Figure out the primary key and look for the requirement of indexing.
     ## in train  and test data set UID is prrimar key so we kept UID as index.
```

```
[5]: df_tn.set_index('UID',inplace=True)
```

```
[6]: df_tn.head()
```

```
[6]:         BLOCKID  SUMLEVEL  COUNTYID  STATEID        state state_ab  \
     UID
     267822      NaN       140        53       36     New York       NY
     246444      NaN       140       141       18      Indiana       IN
     245683      NaN       140        63       18      Indiana       IN
     279653      NaN       140       127       72  Puerto Rico       PR
     247218      NaN       140       161       20       Kansas       KS
```

```
              city           place   type primary  …  female_age_mean  \
UID                                                 …
267822    Hamilton        Hamilton   City   tract  …         44.48629
246444  South Bend        Roseland   City   tract  …         36.48391
245683    Danville        Danville   City   tract  …         42.15810
279653    San Juan        Guaynabo  Urban   tract  …         47.77526
247218   Manhattan  Manhattan City   City   tract  …         24.17693

        female_age_median  female_age_stdev  female_age_sample_weight  \
UID
267822           45.33333          22.51276                 685.33845
246444           37.58333          23.43353                 267.23367
245683           42.83333          23.94119                 707.01963
279653           50.58333          24.32015                 362.20193
247218           21.58333          11.10484                1854.48652

        female_age_samples  pct_own  married  married_snp  separated  divorced
UID
267822              2618.0  0.79046  0.57851      0.01882    0.01240   0.08770
246444              1284.0  0.52483  0.34886      0.01426    0.01426   0.09030
245683              3238.0  0.85331  0.64745      0.02830    0.01607   0.10657
279653              1559.0  0.65037  0.47257      0.02021    0.02021   0.10106
247218              3051.0  0.13046  0.12356      0.00000    0.00000   0.03109

[5 rows x 79 columns]
```

[7]:
```python
df_te.set_index('UID',inplace=True)
df_te
```

[7]:
```
        BLOCKID  SUMLEVEL  COUNTYID  STATEID          state state_ab  \
UID
255504      NaN       140       163       26       Michigan       MI
252676      NaN       140         1       23          Maine       ME
276314      NaN       140        15       42   Pennsylvania       PA
248614      NaN       140       231       21       Kentucky       KY
286865      NaN       140       355       48          Texas       TX
…             …         …         …        …            …        …
238088      NaN       140       105       12        Florida       FL
242811      NaN       140        31       17       Illinois       IL
250127      NaN       140         9       25  Massachusetts       MA
241096      NaN       140        27       19           Iowa       IA
287763      NaN       140       453       48          Texas       TX

                    city                  place     type primary  …  \
UID                                                                …
255504           Detroit  Dearborn Heights City      CDP   tract  …
```

```
252676          Auburn          Auburn City     City   tract  …
276314       Pine City            Millerton  Borough   tract  …
248614      Monticello      Monticello City     City   tract  …
286865  Corpus Christi                Edroy     Town   tract  …
…                  …                    …        …       …  …
238088        Lakeland      Crystal Springs     City   tract  …
242811         Chicago         Chicago City  Village   tract  …
250127        Lawrence    Methuen Town City     City   tract  …
241096         Carroll         Carroll City     City   tract  …
287763          Austin   Sunset Valley City     Town   tract  …

        female_age_mean  female_age_median  female_age_stdev  \
UID
255504         34.78682           33.75000          21.58531
252676         44.23451           46.66667          22.37036
276314         41.62426           44.50000          22.86213
248614         44.81200           48.00000          21.03155
286865         40.66618           42.66667          21.30900
…                   …                  …                 …
238088         53.51255           59.58333          23.23426
242811         33.14169           32.83333          20.24698
250127         43.53905           43.66667          23.17995
241096         45.63179           48.16667          24.84209
287763         35.99955           35.41667          20.68049

        female_age_sample_weight  female_age_samples  pct_own  married  \
UID
255504                 416.48097              1938.0  0.70252  0.28217
252676                 532.03505              1950.0  0.85128  0.64221
276314                 453.11959              1879.0  0.81897  0.59961
248614                 263.94320              1081.0  0.84609  0.56953
286865                 709.90829              2956.0  0.79077  0.57620
…                             …                  …       …       …
238088                 699.33353              2914.0  0.93121  0.65969
242811                 306.63915              1191.0  0.33122  0.42882
250127                 900.13903              3723.0  0.84372  0.50269
241096                 693.82905              3213.0  0.83330  0.66699
287763                 559.30291              2047.0  0.52587  0.51922

        married_snp  separated  divorced
UID
255504      0.05910    0.03813   0.14299
252676      0.02338    0.00000   0.13377
276314      0.01746    0.01358   0.10026
248614      0.05492    0.04694   0.12489
286865      0.01726    0.00588   0.16379
…              …          …         …
```

```
238088      0.02135    0.02135   0.08780
242811      0.07781    0.02829   0.05305
250127      0.00108    0.00108   0.07294
241096      0.02738    0.00000   0.04694
287763      0.08066    0.02520   0.10586

[11709 rows x 79 columns]
```

[8]:
```
##Gauge the fill rate of the variables and devise plans for missing value␣
↪treatment.
##Please explain explicitly the reason for the treatment chosen for each␣
↪variable.
## first check null values and percentage of null values for each column
```

[9]:
```
#for train data missing values
qw=df_tn.isnull().sum()
qw[qw>0]
```

[9]:
```
BLOCKID                        27321
rent_mean                        314
rent_median                      314
rent_stdev                       314
rent_sample_weight               314
rent_samples                     314
rent_gt_10                       314
rent_gt_15                       314
rent_gt_20                       314
rent_gt_25                       314
rent_gt_30                       314
rent_gt_35                       314
rent_gt_40                       314
rent_gt_50                       314
hi_mean                          268
hi_median                        268
hi_stdev                         268
hi_sample_weight                 268
hi_samples                       268
family_mean                      298
family_median                    298
family_stdev                     298
family_sample_weight             298
family_samples                   298
hc_mortgage_mean                 573
hc_mortgage_median               573
hc_mortgage_stdev                573
hc_mortgage_sample_weight        573
hc_mortgage_samples              573
```

```
hc_mean                          600
hc_median                        600
hc_stdev                         600
hc_samples                       600
hc_sample_weight                 600
home_equity_second_mortgage      457
second_mortgage                  457
home_equity                      457
debt                             457
second_mortgage_cdf              457
home_equity_cdf                  457
debt_cdf                         457
hs_degree                        190
hs_degree_male                   200
hs_degree_female                 223
male_age_mean                    189
male_age_median                  189
male_age_stdev                   189
male_age_sample_weight           189
male_age_samples                 189
female_age_mean                  206
female_age_median                206
female_age_stdev                 206
female_age_sample_weight         206
female_age_samples               206
pct_own                          268
married                          191
married_snp                      191
separated                        191
divorced                         191
dtype: int64
```

[10]:
```
#percentage of missing values for train data
df_tn_null= df_tn.isnull().sum()*100/len(df_tn)
print(df_tn_null[df_tn_null>0].sort_values(ascending=False))
#since BLockId has 100%miising value in train data so we drop it.
```

```
BLOCKID                        100.000000
hc_sample_weight                 2.196113
hc_median                        2.196113
hc_stdev                         2.196113
hc_samples                       2.196113
hc_mean                          2.196113
hc_mortgage_stdev                2.097288
hc_mortgage_mean                 2.097288
hc_mortgage_median               2.097288
hc_mortgage_sample_weight        2.097288
```

```
hc_mortgage_samples            2.097288
debt_cdf                       1.672706
second_mortgage                1.672706
home_equity                    1.672706
debt                           1.672706
second_mortgage_cdf            1.672706
home_equity_cdf                1.672706
home_equity_second_mortgage    1.672706
rent_gt_25                     1.149299
rent_gt_35                     1.149299
rent_mean                      1.149299
rent_median                    1.149299
rent_stdev                     1.149299
rent_gt_20                     1.149299
rent_gt_50                     1.149299
rent_gt_40                     1.149299
rent_sample_weight             1.149299
rent_gt_30                     1.149299
rent_samples                   1.149299
rent_gt_10                     1.149299
rent_gt_15                     1.149299
family_samples                 1.090736
family_sample_weight           1.090736
family_stdev                   1.090736
family_median                  1.090736
family_mean                    1.090736
pct_own                        0.980930
hi_median                      0.980930
hi_mean                        0.980930
hi_samples                     0.980930
hi_sample_weight               0.980930
hi_stdev                       0.980930
hs_degree_female               0.816222
female_age_sample_weight       0.753999
female_age_stdev               0.753999
female_age_samples             0.753999
female_age_mean                0.753999
female_age_median              0.753999
hs_degree_male                 0.732038
separated                      0.699096
married_snp                    0.699096
married                        0.699096
divorced                       0.699096
hs_degree                      0.695436
male_age_mean                  0.691776
male_age_median                0.691776
male_age_stdev                 0.691776
male_age_sample_weight         0.691776
```

```
male_age_samples                  0.691776
dtype: float64
```

[11]:
```
df_tn.drop(columns=['BLOCKID'],inplace = True)
```

[12]:
```
##missing values in test data
df_te_null= df_te.isnull().sum()*100/len(df_tn)
print(df_te_null[df_te_null>0].sort_values(ascending=False))
##since blockid has 43% missing values so we drop it.
```

```
BLOCKID                         42.857143
hc_sample_weight                 1.061455
hc_median                        1.061455
hc_stdev                         1.061455
hc_samples                       1.061455
hc_mean                          1.061455
hc_mortgage_stdev                0.980930
hc_mortgage_mean                 0.980930
hc_mortgage_median               0.980930
hc_mortgage_sample_weight        0.980930
hc_mortgage_samples              0.980930
debt_cdf                         0.805241
second_mortgage                  0.805241
home_equity                      0.805241
debt                             0.805241
second_mortgage_cdf              0.805241
home_equity_cdf                  0.805241
home_equity_second_mortgage      0.805241
rent_gt_25                       0.545368
rent_gt_50                       0.545368
rent_gt_15                       0.545368
rent_gt_10                       0.545368
rent_gt_30                       0.545368
rent_gt_20                       0.545368
rent_gt_35                       0.545368
rent_gt_40                       0.545368
rent_mean                        0.541708
rent_median                      0.541708
rent_stdev                       0.541708
rent_sample_weight               0.541708
rent_samples                     0.541708
family_samples                   0.497786
family_sample_weight             0.497786
family_stdev                     0.497786
family_median                    0.497786
family_mean                      0.497786
pct_own                          0.446543
hi_median                        0.446543
```

```
hi_mean                       0.446543
hi_samples                    0.446543
hi_sample_weight              0.446543
hi_stdev                      0.446543
hs_degree_female              0.384320
female_age_sample_weight      0.351378
female_age_stdev              0.351378
female_age_samples            0.351378
female_age_mean               0.351378
female_age_median             0.351378
hs_degree_male                0.325757
hs_degree                     0.311116
separated                     0.307456
married_snp                   0.307456
married                       0.307456
male_age_mean                 0.307456
male_age_median               0.307456
male_age_stdev                0.307456
male_age_sample_weight        0.307456
male_age_samples              0.307456
divorced                      0.307456
dtype: float64
```

[13]:
```
df_te.drop(columns=['BLOCKID','SUMLEVEL'],inplace = True)##we drop sumlevel␣
↪doesn't give any prediction so we drop it
```

[14]:
```
df_tn.drop(columns=['SUMLEVEL'],inplace=True)#we drop sumlevel doesn't give any␣
↪prediction so we drop it
```

[15]:
```
##imputing missing values with mean
missing_train_cols=[]
for col in df_tn.columns:
    if df_tn[col].isna().sum() !=0:
         missing_train_cols.append(col)
print(missing_train_cols)
```

```
['rent_mean', 'rent_median', 'rent_stdev', 'rent_sample_weight', 'rent_samples',
'rent_gt_10', 'rent_gt_15', 'rent_gt_20', 'rent_gt_25', 'rent_gt_30',
'rent_gt_35', 'rent_gt_40', 'rent_gt_50', 'hi_mean', 'hi_median', 'hi_stdev',
'hi_sample_weight', 'hi_samples', 'family_mean', 'family_median',
'family_stdev', 'family_sample_weight', 'family_samples', 'hc_mortgage_mean',
'hc_mortgage_median', 'hc_mortgage_stdev', 'hc_mortgage_sample_weight',
'hc_mortgage_samples', 'hc_mean', 'hc_median', 'hc_stdev', 'hc_samples',
'hc_sample_weight', 'home_equity_second_mortgage', 'second_mortgage',
'home_equity', 'debt', 'second_mortgage_cdf', 'home_equity_cdf', 'debt_cdf',
'hs_degree', 'hs_degree_male', 'hs_degree_female', 'male_age_mean',
'male_age_median', 'male_age_stdev', 'male_age_sample_weight',
'male_age_samples', 'female_age_mean', 'female_age_median', 'female_age_stdev',
'female_age_sample_weight', 'female_age_samples', 'pct_own', 'married',
'married_snp', 'separated', 'divorced']
```

[16]:
```
missing_test_cols=[]
for col in df_te.columns:
    if df_te[col].isna().sum() !=0:
         missing_test_cols.append(col)
print(missing_test_cols)
```

```
['rent_mean', 'rent_median', 'rent_stdev', 'rent_sample_weight', 'rent_samples',
'rent_gt_10', 'rent_gt_15', 'rent_gt_20', 'rent_gt_25', 'rent_gt_30',
'rent_gt_35', 'rent_gt_40', 'rent_gt_50', 'hi_mean', 'hi_median', 'hi_stdev',
'hi_sample_weight', 'hi_samples', 'family_mean', 'family_median',
'family_stdev', 'family_sample_weight', 'family_samples', 'hc_mortgage_mean',
'hc_mortgage_median', 'hc_mortgage_stdev', 'hc_mortgage_sample_weight',
'hc_mortgage_samples', 'hc_mean', 'hc_median', 'hc_stdev', 'hc_samples',
'hc_sample_weight', 'home_equity_second_mortgage', 'second_mortgage',
'home_equity', 'debt', 'second_mortgage_cdf', 'home_equity_cdf', 'debt_cdf',
'hs_degree', 'hs_degree_male', 'hs_degree_female', 'male_age_mean',
'male_age_median', 'male_age_stdev', 'male_age_sample_weight',
'male_age_samples', 'female_age_mean', 'female_age_median', 'female_age_stdev',
'female_age_sample_weight', 'female_age_samples', 'pct_own', 'married',
'married_snp', 'separated', 'divorced']
```

[17]:
```
for col in df_tn.columns:
    if col in (missing_train_cols):
        df_tn[col].replace(np.nan, df_tn[col].mean(),inplace=True)
```

[18]:
```
for col in df_te.columns:
    if col in (missing_test_cols):
        df_te[col].replace(np.nan, df_te[col].mean(),inplace=True)
```

[19]:
```
df_tn.isnull().sum().sum()
```

[19]:
```
0
```

[20]:
```
df_te.isnull().sum().sum()
```

[20]:
```
0
```

[21]:
```
/#### Exploratory data analysis
```

[22]:
```
df_col=df_tn[['place','lat','lng']]
df_col
```

[22]:
```
                   place         lat          lng
UID
```

```
267822          Hamilton  42.840812  -75.501524
246444          Roseland  41.701441  -86.266614
245683          Danville  39.792202  -86.515246
279653          Guaynabo  18.396103  -66.104169
247218    Manhattan City  39.195573  -96.569366
…                    …          …           …
279212             Coamo  18.076060  -66.358379
277856         Blue Bell  40.158138  -75.307271
233000      Saddle Ridge  40.410316 -103.814003
287425  Colleyville City  32.904866  -97.162151
265371          Paradise  36.064754 -115.152237

[27321 rows x 3 columns]
```

[23]:
```
df_1=df_tn[(df_tn["pct_own"] >0.10) & (df_tn["second_mortgage"] < 0.5)]
df_tn_location_mort_pct=df_1[['pct_own','second_mortgage','place','lat','lng']].
 ↪sort_values(by='second_mortgage',ascending=False)
df_tn_location_mort_pct.head()
```

[23]:
```
        pct_own  second_mortgage            place        lat        lng
UID
251185  0.20247          0.43363   Worcester City  42.254262 -71.800347
269323  0.15618          0.31818     Harbor Hills  40.751809 -73.853582
251324  0.22380          0.30212      Glen Burnie  39.127273 -76.635265
235788  0.11618          0.28972  Egypt Lake-leto  28.029063 -82.495395
242304  0.14228          0.28899       Lincolnwood  41.967289 -87.652434
```

[24]:
```
import plotly.express as px
import plotly.graph_objects as go
```

[25]:
```
##Use the following bad debt equation: Bad Debt = P (Second Mortgage  Home␣
 ↪Equity Loan) Bad Debt = second_mortgage + home_equity -␣
 ↪home_equity_second_mortgage c)
##Create pie charts to show overall debt and bad debt
```

[26]:
```
df_tn['Bad_Debt']=df_tn['second_mortgage']+df_tn['home_equity']-df_tn['home_equity_second_mort
df_tn['Bad_Debt'].head()
```

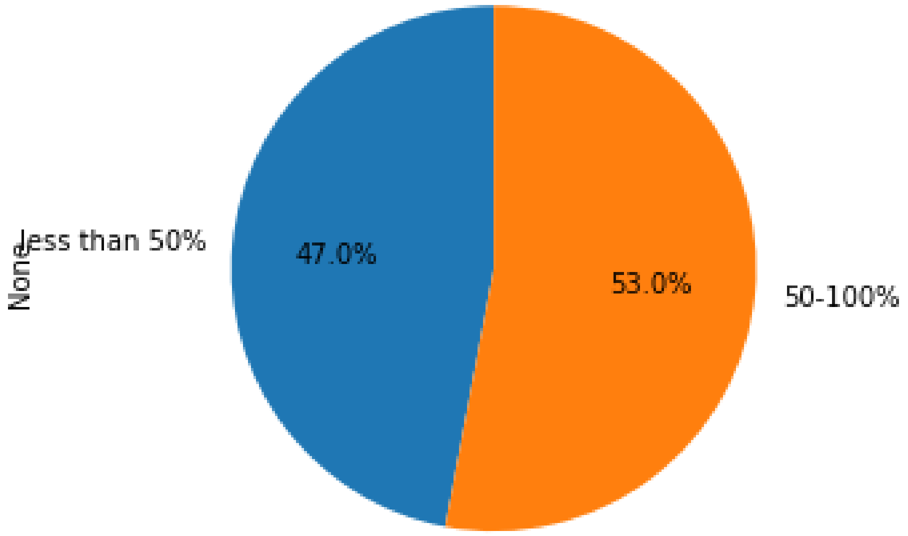
[26]:
```
UID
267822    0.09408
246444    0.04274
245683    0.09512
279653    0.01086
247218    0.05426
Name: Bad_Debt, dtype: float64
```

```
[27]: debt=df_tn['debt']
      debt.head()
```

```
[27]: UID
      267822    0.52963
      246444    0.60855
      245683    0.73484
      279653    0.52714
      247218    0.51938
      Name: debt, dtype: float64
```

```
[97]: df_tn['bins'] = pd.cut(df_tn['Bad_Debt'],bins=[0,0.10,1], labels=["less than␣
       ↪50%","50-100%"])
      df_tn.groupby(['bins']).size().plot(kind='pie',subplots=True,startangle=90,␣
       ↪autopct='%1.1f%%')
      plt.axis('equal')
```

```
[97]: (-1.1201881996052996,
       1.1099751547716121,
       -1.108903086418464,
       1.1004239564961174)
```



```
[28]: ##Create Box and whisker plot and analyze the distribution for
      ##2nd mortgage, home equity, good debt, and bad debt for different cities
```

```
[29]: #Lets take Las Vegas,Hamilton city for analysis
      df_tn_hamilton = df_tn.loc[df_tn['city']=='Hamilton']
```

```
df_tn_las = df_tn.loc[df_tn['city']=='Danville']
df_tn_Coamo=df_tn.loc[df_tn['city']=='Coamo']
```

[ ]:

```
```

[30]:

```
df_tn_city=pd.concat([df_tn_hamilton,df_tn_las])
df_tn_city.head()
```

[30]:

```
        COUNTYID  STATEID        state state_ab      city           place  \
UID
267822        53       36     New York       NY  Hamilton        Hamilton
263797        21       34   New Jersey       NJ  Hamilton       Yardville
270979        17       39         Ohio       OH  Hamilton  Hamilton City
259028        95       28  Mississippi       MS  Hamilton        Hamilton
270984        17       39         Ohio       OH  Hamilton       New Miami

           type primary  zip_code  area_code  …  female_age_median  \
UID                                           …
267822     City   tract     13346        315  …           45.33333
263797     City   tract      8610        609  …           55.00000
270979  Village   tract     45015        513  …           31.66667
259028      CDP   tract     39746        662  …           35.91667
270984  Village   tract     45013        513  …           52.33333

        female_age_stdev  female_age_sample_weight  female_age_samples  \
UID
267822          22.51276                 685.33845              2618.0
263797          24.05831                 732.58443              3124.0
270979          22.66500                 565.32725              2528.0
259028          22.79602                 483.01311              1954.0
270984          24.55724                 682.81171              2912.0

        pct_own  married  married_snp  separated  divorced  Bad_Debt
UID
267822  0.79046  0.57851      0.01882    0.01240   0.08770   0.09408
263797  0.64400  0.56377      0.01980    0.00990   0.04892   0.18071
270979  0.61278  0.47397      0.04419    0.02663   0.13741   0.15005
259028  0.83241  0.58678      0.01052    0.00000   0.11721   0.02130
270984  0.63194  0.55697      0.01322    0.00000   0.15209   0.15651

[5 rows x 78 columns]
```

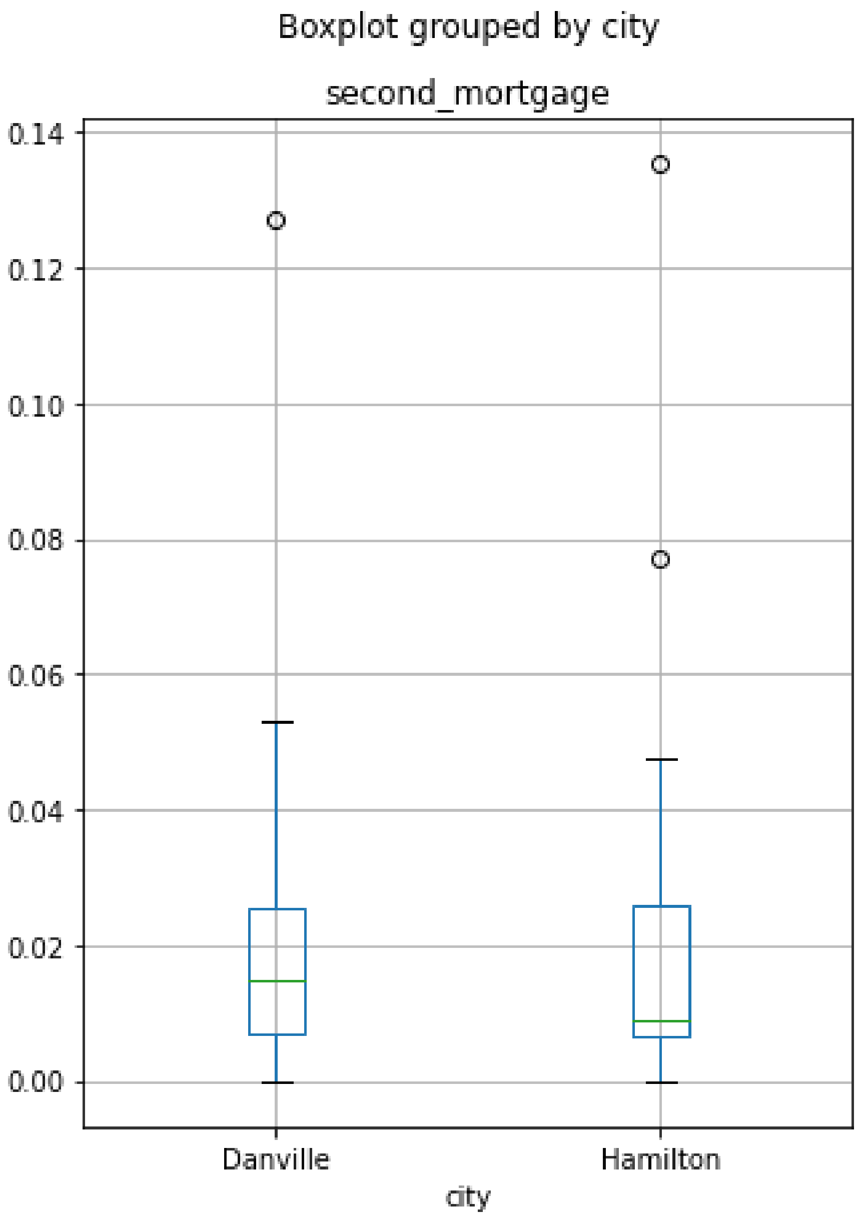
[31]:

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

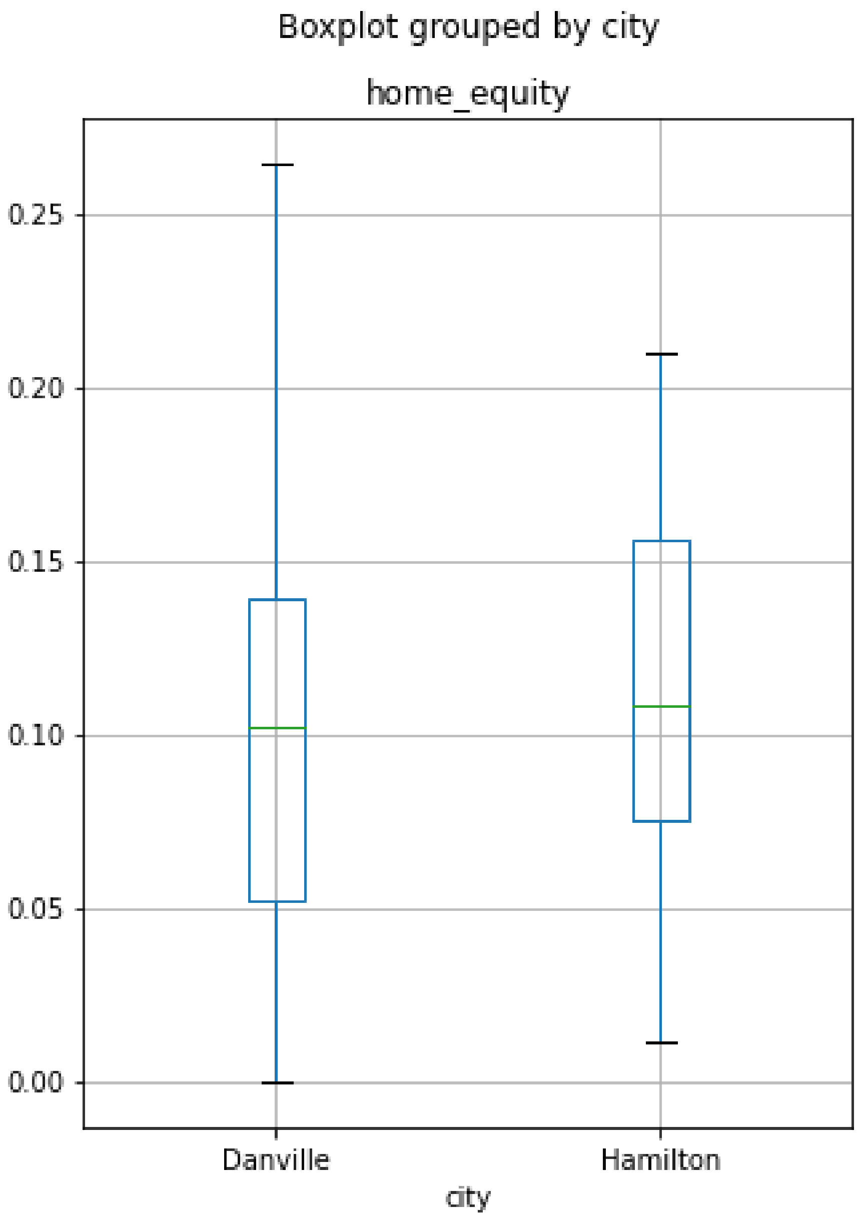
[32]:

```
df_tn_city.boxplot(by ='city', column =['second_mortgage'], grid =␣
 ↪True,figsize=(5,7))
```

```
plt.show()
```

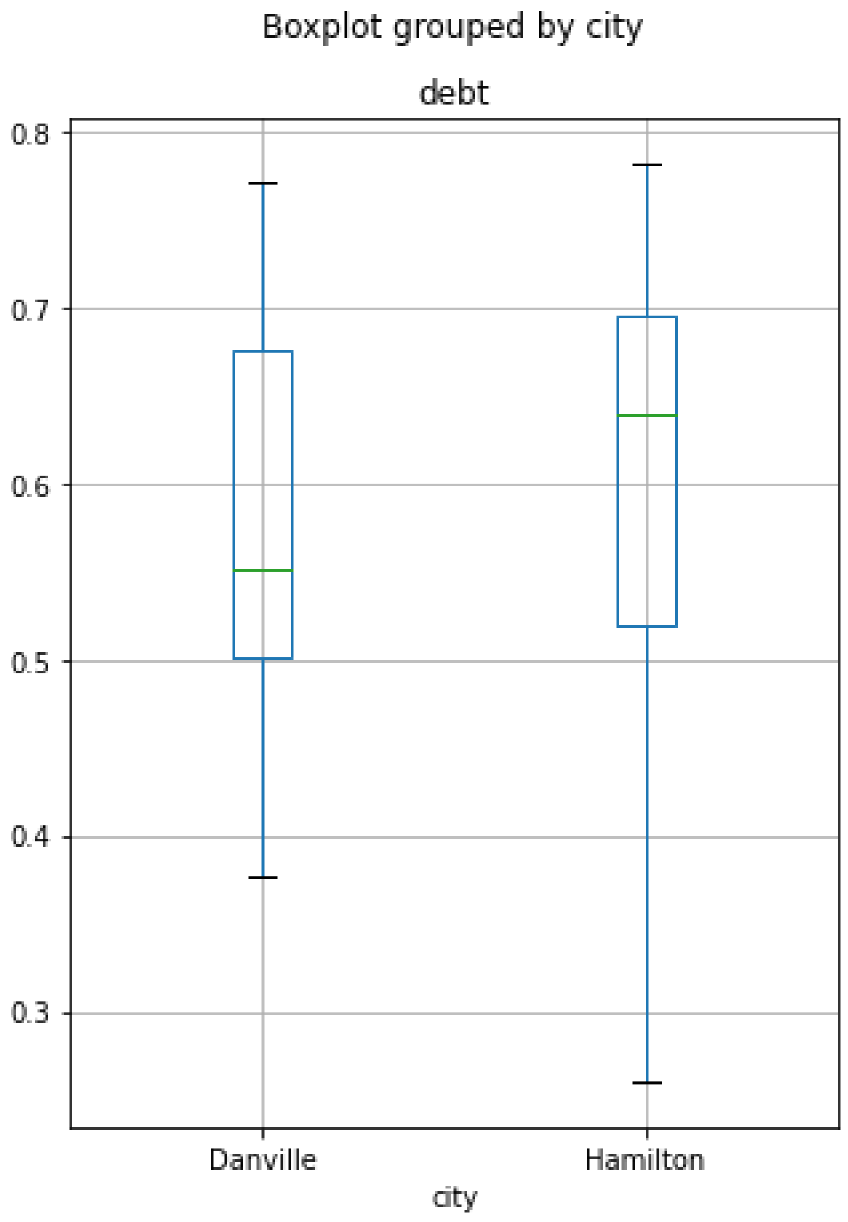


[33]:
```
df_tn_city.boxplot(by ='city', column =['home_equity'], grid =␣
↪True,figsize=(5,7))
plt.show()
```



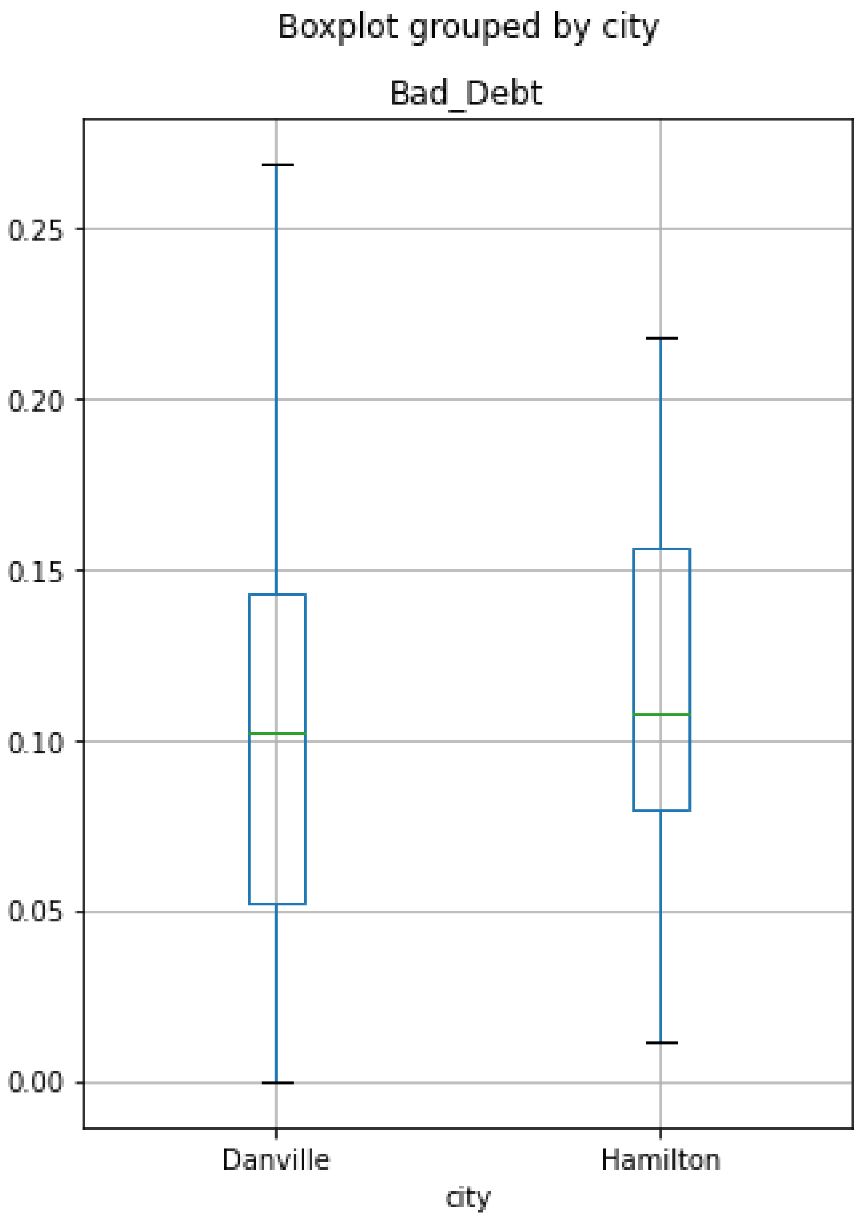
```
[34]: df_tn_city.boxplot(by ='city', column =['debt'], grid = True,figsize=(5,7))
```

```
[34]: <AxesSubplot:title={'center':'debt'}, xlabel='city'>
```



```
[35]: df_tn_city.boxplot(by ='city', column =['Bad_Debt'], grid = True,figsize=(5,7))
```

```
[35]: <AxesSubplot:title={'center':'Bad_Debt'}, xlabel='city'>
```



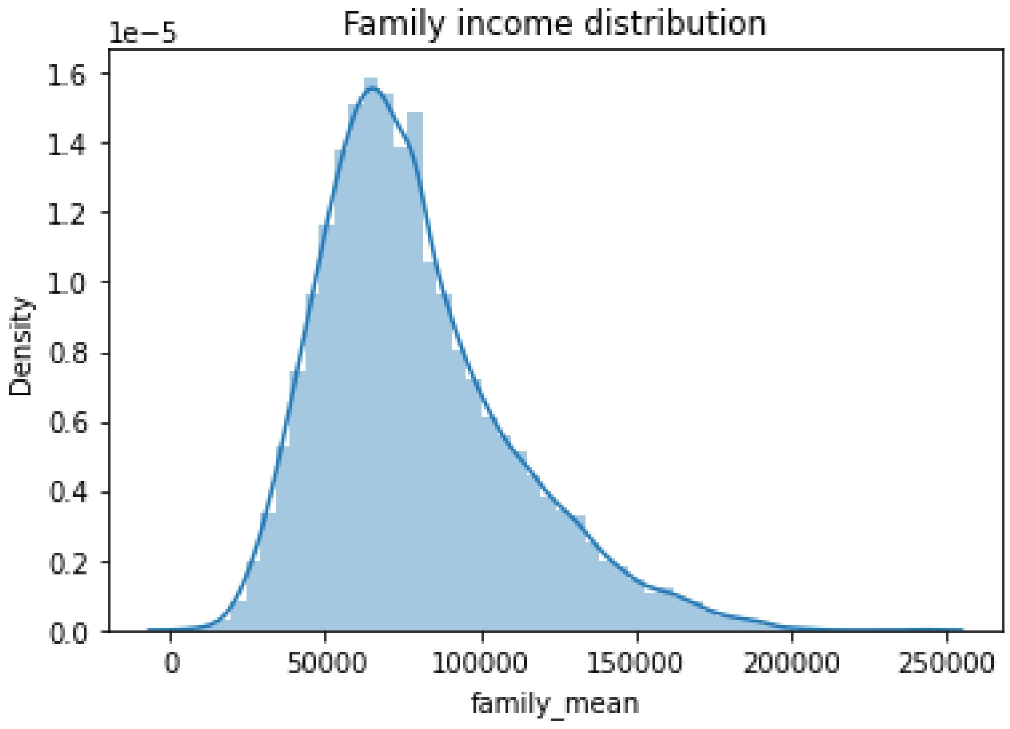
[36]:
```
##inoverall Danvile city has maximum matrices than Hamilton
```

[37]:
```
##Create a collated income distribution chart for family income, house hold␣
↪income, and remaining income
```

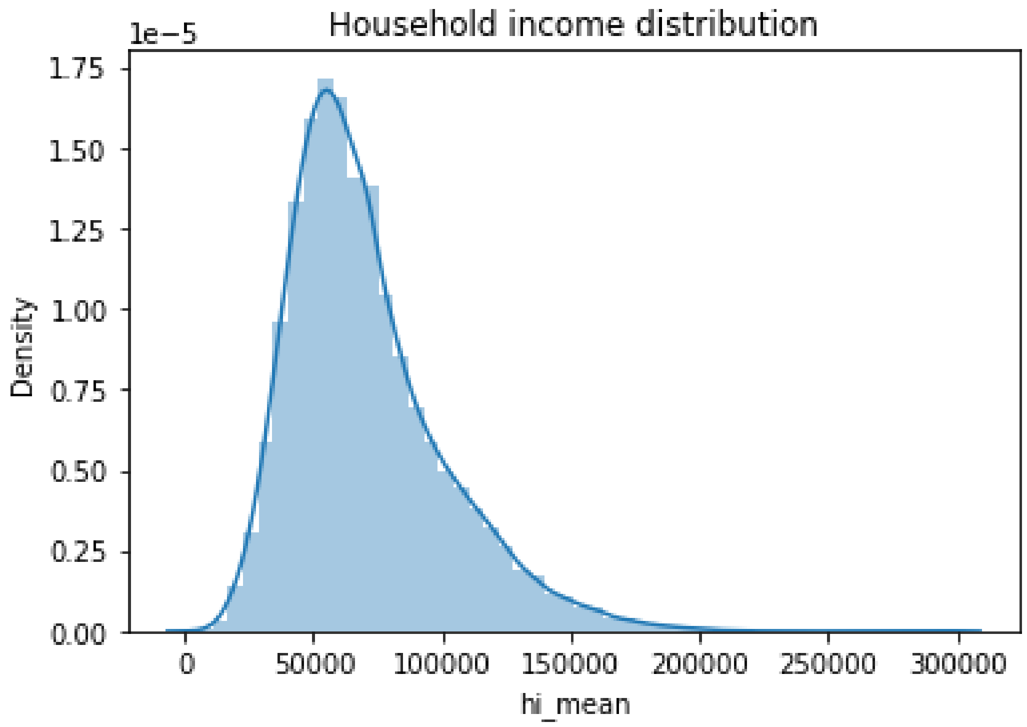
[38]:
```
import seaborn as sns
sns.distplot(df_tn['family_mean'])
plt.title("Family income distribution")
plt.show()
import warnings
```

```
warnings.filterwarnings(action= 'ignore')
```

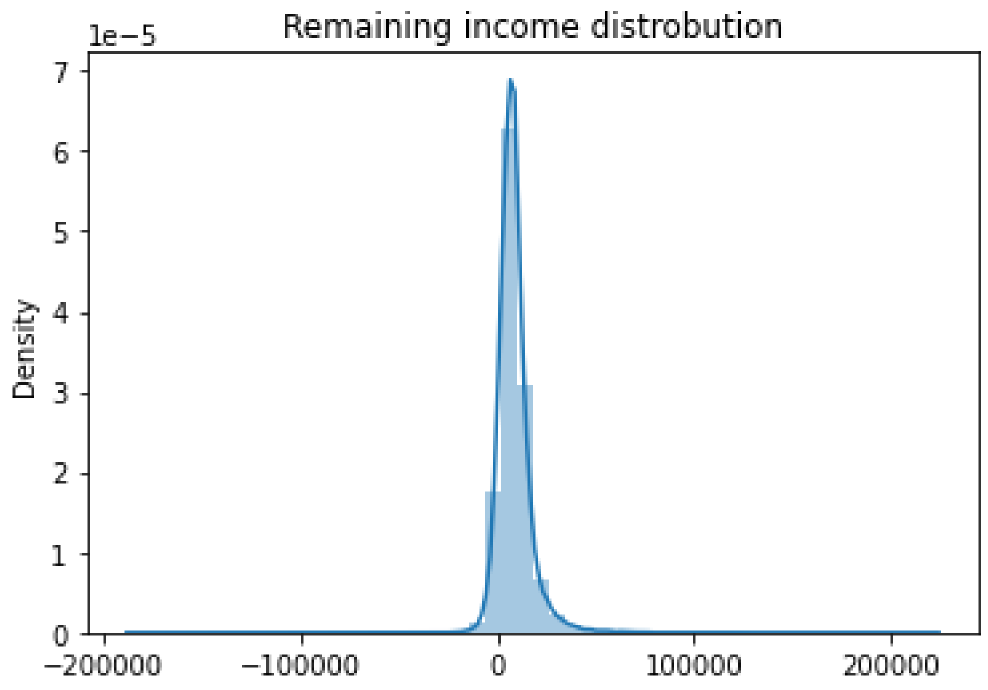
```
/usr/local/lib/python3.7/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```



[39]:
```
sns.distplot(df_tn['hi_mean'])
plt.title("Household income distribution")
plt.show()
import warnings
warnings.filterwarnings(action= 'ignore')
```



[40]:
```
sns.distplot(df_tn['family_mean']-df_tn['hi_mean'])
plt.title("Remaining income distrobution")
plt.show()
import warnings
warnings.filterwarnings(action= 'ignore')
```



Exploratory Data Analysis (EDA): Perform debt analysis. You may take the following steps:

Explore the top 2,500 locations where the percentage of households with a second mortgage is the highest and percent ownership is above 10 percent. Visualize using geo-map. You may keep the upper limit for the percent of households with a second mortgage to 50 percent

Use the following bad debt equation:

Bad Debt = P (Second Mortgage   Home Equity Loan) Bad Debt = second_mortgage + home_equity - home_equity_second_mortgage Create pie charts to show overall debt and bad debt

Create Box and whisker plot and analyze the distribution for 2nd mortgage, home equity, good debt, and bad debt for different cities

Create a collated income distribution chart for family income, house hold income, and remaining income

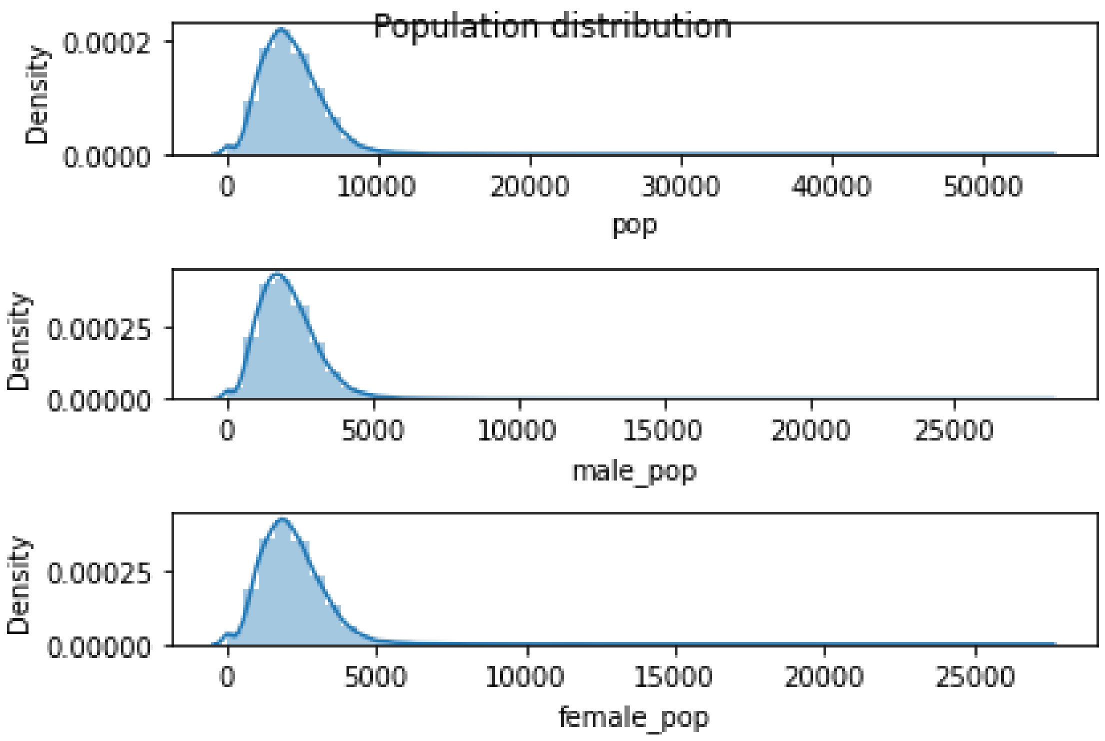
[41]:
```
##Perform EDA and come out with insights into population density and age. You␣
↪may have to derive new fields (make sure to weight averages for accurate␣
↪measurements):

##Use pop and ALand variables to create a new field called population density

##Use male_age_median, female_age_median, male_pop, and female_pop to create a␣
↪new field called median age
```

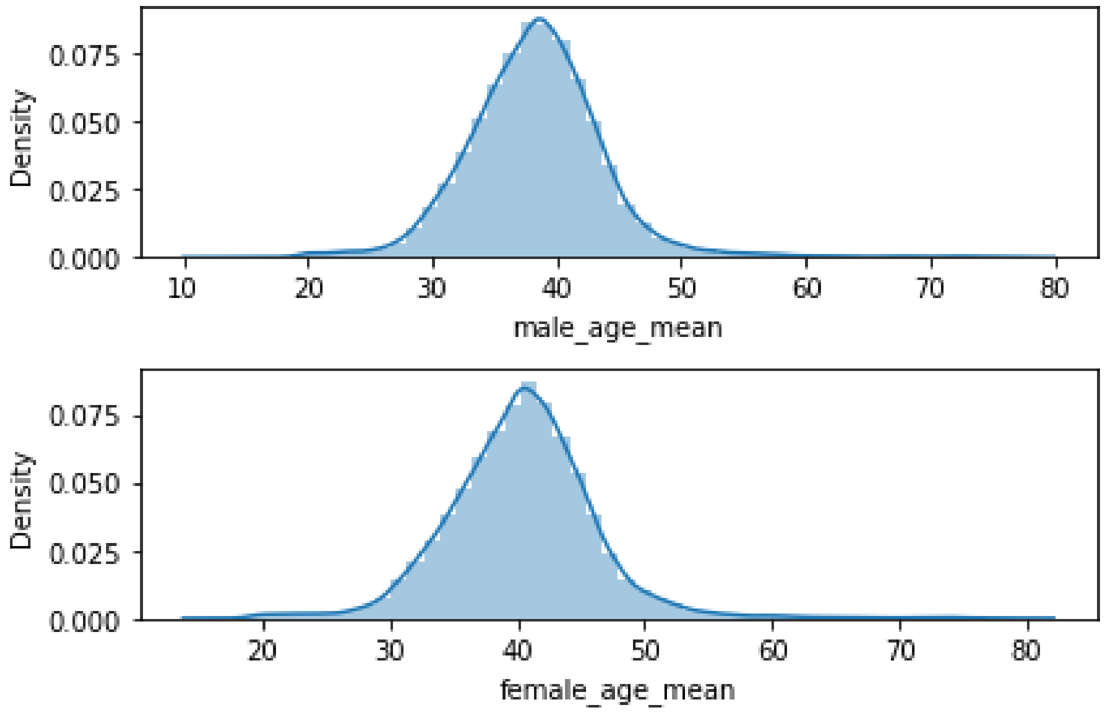
```
##Visualize the findings using appropriate chart type
```

[42]:
```
import seaborn as sns
figure,(ax1,ax2,ax3) = plt.subplots(3,1)
sns.distplot(df_tn['pop'],ax=ax1)
sns.distplot(df_tn['male_pop'],ax=ax2)
sns.distplot(df_tn['female_pop'],ax=ax3)
plt.subplots_adjust(wspace=0.8,hspace=0.8)
plt.tight_layout()
figure.suptitle('Population distribution')
plt.show()
import warnings
warnings.filterwarnings(action= 'ignore')
import warnings
warnings.filterwarnings(action= 'ignore')
```



[43]:
```
igure,(ax1,ax2) = plt.subplots(2,1)
sns.distplot(df_tn['male_age_mean'],ax=ax1)
sns.distplot(df_tn['female_age_mean'],ax=ax2)
plt.subplots_adjust(wspace=0.8,hspace=0.8)
plt.tight_layout()
plt.show()
import warnings
```

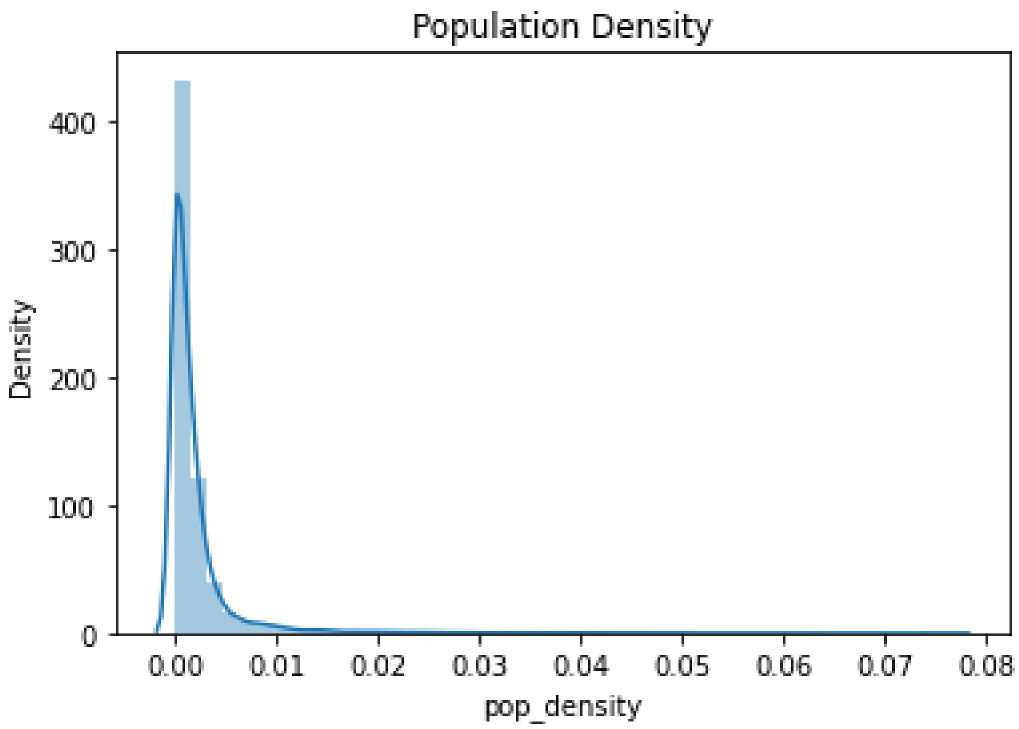
```
warnings.filterwarnings(action= 'ignore')
```



[44]:
```
df_tn["pop_density"]=df_tn["pop"]/df_tn["ALand"]
```

[45]:
```
df_te["pop_density"]=df_te["pop"]/df_te["ALand"]
```

[46]:
```
sns.distplot(df_tn['pop_density'])
plt.title('Population Density')
plt.show()
##we observe very less distribution
```

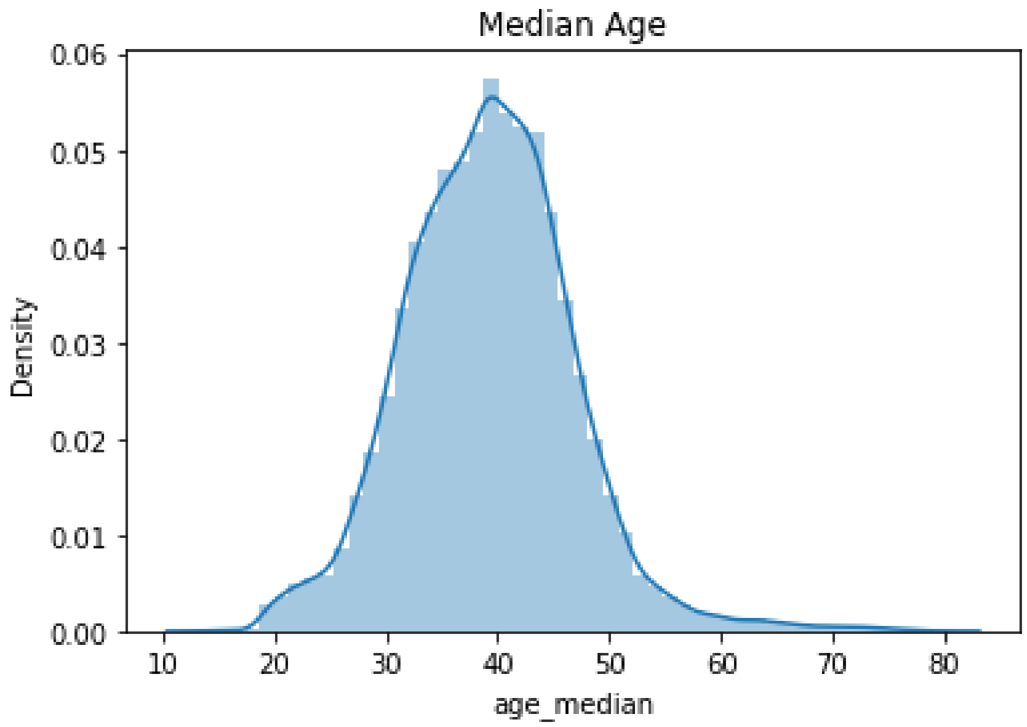


```
[47]: df_tn['age_median']=(df_tn['male_age_median']+df_tn['female_age_median'])/2
      df_te['age_median']=(df_te['male_age_median']+df_te['female_age_median'])/2
```

```
[48]: df_tn[['male_age_median','female_age_median','male_pop','female_pop','age_median']].
       ↪head()
```

```
[48]:         male_age_median  female_age_median  male_pop  female_pop  age_median
      UID
      267822         44.00000           45.33333      2612        2618   44.666665
      246444         32.00000           37.58333      1349        1284   34.791665
      245683         40.83333           42.83333      3643        3238   41.833330
      279653         48.91667           50.58333      1141        1559   49.750000
      247218         22.41667           21.58333      2586        3051   22.000000
```

```
[49]: sns.distplot(df_tn["age_median"])
      plt.title("Median Age")
      plt.show()
      #mamximum age from 30 to 50
      #average age of people around of 40
      ## right skewness is there
```



[50]:
```
##Create bins for population into a new variable by selecting appropriate class␣
↪interval
##so that the number of categories don't exceed 5 for the ease of analysis.

##Analyze the married, separated, and divorced population for these population␣
↪brackets

##Visualize using appropriate chart type
```

[51]:
```
df_tn["pop"].describe()
```

[51]:
```
count    27321.000000
mean      4316.032685
std       2169.226173
min          0.000000
25%       2885.000000
50%       4042.000000
75%       5430.000000
max      53812.000000
Name: pop, dtype: float64
```

[52]:
```
df_tn["pop"].value_counts().head(4)
```

[52]:
```
0       182
2872     15
4824     14
3706     14
Name: pop, dtype: int64
```

[53]:
```
df_tn['pop_bins']=pd.cut(df_tn['pop'],bins=5,labels=['very␣
 ↪low','low','medium','high','very high'])
```

[54]:
```
df_tn[["pop_bins","pop"]]
```

[54]:
```
        pop_bins    pop
UID
267822  very low   5230
246444  very low   2633
245683  very low   6881
279653  very low   2700
247218  very low   5637
…            …      …
279212  very low   1847
277856  very low   4155
233000  very low   2829
287425       low  11542
265371  very low   3726

[27321 rows x 2 columns]
```

[55]:
```
df_tn['pop_bins'].value_counts()
```

[55]:
```
very low     27058
low            246
medium           9
high             7
very high        1
Name: pop_bins, dtype: int64
```

[56]:
```
df_tn.groupby(by='pop_bins')[['married','separated','divorced']].count()
```

[56]:
```
           married  separated  divorced
pop_bins
very low     27058      27058     27058
low            246        246       246
medium           9          9         9
high             7          7         7
very high        1          1         1
```

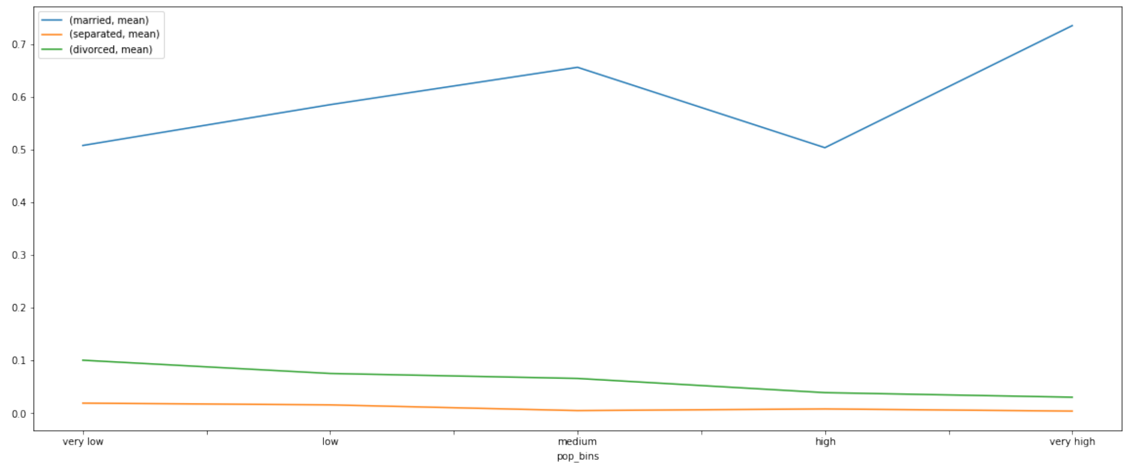
[57]:
```
df_tn.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean",␣
↪"median"])
```

[57]:
```
             married             separated             divorced
                mean    median        mean    median       mean    median
pop_bins
very low    0.507548  0.524680    0.019126  0.013650   0.100504  0.096020
low         0.584894  0.593135    0.015833  0.011195   0.075348  0.070045
medium      0.655737  0.618710    0.005003  0.004120   0.065927  0.064890
high        0.503359  0.335660    0.008141  0.002500   0.039030  0.010320
very high   0.734740  0.734740    0.004050  0.004050   0.030360  0.030360
```

very high popuation has high married couple and less seprated and divorced in verylow population group has high divorced

[58]:
```
plt.figure(figsize=(10,5))
pop_bin_married=df_tn.
↪groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean"])
pop_bin_married.plot(figsize=(20,8))
plt.legend(loc='best')
plt.show()
```

```
<Figure size 720x360 with 0 Axes>
```



[59]:
```
##Please detail your observations for rent as a percentage of income at an␣
↪overall level, and for different states.
```

[60]:
```
df_tn_rent = df_tn.groupby(by='state')['rent_mean'].agg(["mean"])
df_tn_rent.head()
```

```
[60]:                  mean
      state
      Alabama      774.004927
      Alaska      1185.763570
      Arizona     1097.753511
      Arkansas     720.918575
      California  1471.133857
```

```
[61]: df_tn_income = df_tn.groupby(by='state')['family_mean'].agg(["mean"])
      df_tn_income.head()
```

```
[61]:                   mean
      state
      Alabama     67030.064213
      Alaska      92136.545109
      Arizona     73328.238798
      Arkansas    64765.377850
      California  87655.470820
```

```
[62]: df_tn_rent_income=df_tn_rent["mean"]/df_tn_income["mean"]
      df_tn_rent_income.head()
```

```
[62]: state
      Alabama       0.011547
      Alaska        0.012870
      Arizona       0.014970
      Arkansas      0.011131
      California    0.016783
      Name: mean, dtype: float64
```

```
[63]: ##overall rent percent of income
      print("overall rent percent of income ")
      sum(df_tn['rent_mean'])/sum(df_tn['family_mean'])*100
```

```
     overall rent percent of income
```

```
[63]: 1.3358170721473863
```

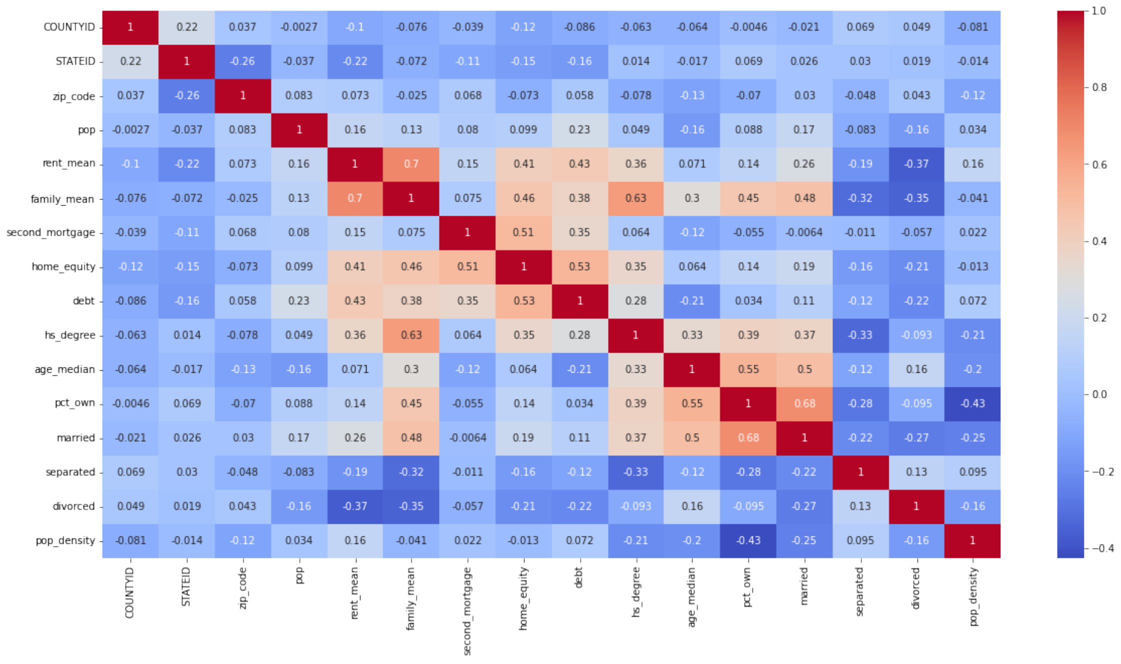
```
[64]: ##Perform correlation analysis for all the relevant variables by creating a␣
       ↪heatmap. Describe your findings.
```

```
[65]: df_tn.columns
```

```
[65]: Index(['COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'type',
             'primary', 'zip_code', 'area_code', 'lat', 'lng', 'ALand', 'AWater',
             'pop', 'male_pop', 'female_pop', 'rent_mean', 'rent_median',
             'rent_stdev', 'rent_sample_weight', 'rent_samples', 'rent_gt_10',
             'rent_gt_15', 'rent_gt_20', 'rent_gt_25', 'rent_gt_30', 'rent_gt_35',
             'rent_gt_40', 'rent_gt_50', 'universe_samples', 'used_samples',
             'hi_mean', 'hi_median', 'hi_stdev', 'hi_sample_weight', 'hi_samples',
             'family_mean', 'family_median', 'family_stdev', 'family_sample_weight',
             'family_samples', 'hc_mortgage_mean', 'hc_mortgage_median',
             'hc_mortgage_stdev', 'hc_mortgage_sample_weight', 'hc_mortgage_samples',
             'hc_mean', 'hc_median', 'hc_stdev', 'hc_samples', 'hc_sample_weight',
             'home_equity_second_mortgage', 'second_mortgage', 'home_equity', 'debt',
             'second_mortgage_cdf', 'home_equity_cdf', 'debt_cdf', 'hs_degree',
             'hs_degree_male', 'hs_degree_female', 'male_age_mean',
             'male_age_median', 'male_age_stdev', 'male_age_sample_weight',
             'male_age_samples', 'female_age_mean', 'female_age_median',
             'female_age_stdev', 'female_age_sample_weight', 'female_age_samples',
             'pct_own', 'married', 'married_snp', 'separated', 'divorced',
             'Bad_Debt', 'pop_density', 'age_median', 'pop_bins'],
            dtype='object')
```

[66]:
```
df_tn_co=df_tn[['COUNTYID','STATEID','zip_code','type','pop','rent_mean','family_mean','second
               'home_equity', 'debt','hs_degree','age_median','pct_own',␣
↪'married','separated', 'divorced','pop_density']].corr()
```

[67]:
```
plt.figure(figsize=(20,10))
sns.heatmap(df_tn_co,annot=True,cmap='coolwarm')
plt.show()
```



1.High positive correaltion is noticed between pop, male_pop and female_pop

2.High positive correaltion is noticed between rent_mean,hi_mean, family_mean,hc_mean

[68]:
```
# #1. The economic multivariate data has a significant number of measured␣
↪variables.
# #The goal is to find where the measured variables depend on a number of␣
↪smaller unobserved common factors or latent variables.
# #2. Each variable is assumed to be dependent upon a linear combination of␣
↪the common factors, and the coefficients are known as loadings.
# #Each measured variable also includes a component due to independent random␣
↪variability, known as "specific variance" because it is specific to one␣
↪variable. Obtain the common factors and then plot the loadings.
# #Use factor analysis to find latent variables in our dataset and gain␣
↪insight into the linear relationships in the data.
# #Following are the list of latent variables:
# #• Highschool graduation rates • Median population age • Second mortgage␣
↪statistics • Percent own • Bad debt expense
```

[69]:
```
from sklearn.decomposition import FactorAnalysis
```

[70]:
```
transformer = FactorAnalysis(n_components=7, random_state=0)
X_transformed = transformer.fit_transform(df_tn.select_dtypes(exclude=␣
 ↪('object','category')))
```

[71]:
```
X_transformed.shape
```

[71]:
```
(27321, 7)
```

[72]:
```
##Data Modeling : Linear Regression
##Build a linear Regression model to predict the total monthly expenditure for␣
 ↪home mortgages loan.
##Please refer ‘deplotment_RE.xlsx’. Column hc_mortgage_mean is predicted␣
 ↪variable.
##This is the mean monthly mortgage and owner costs of specified geographical␣
 ↪location.
##Note: Exclude loans from prediction model which have NaN (Not a Number)␣
 ↪values for hc_mortgage_mean.
```

[73]:
```
df_tn.columns
```

[73]:
```
Index(['COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'type',
       'primary', 'zip_code', 'area_code', 'lat', 'lng', 'ALand', 'AWater',
       'pop', 'male_pop', 'female_pop', 'rent_mean', 'rent_median',
       'rent_stdev', 'rent_sample_weight', 'rent_samples', 'rent_gt_10',
       'rent_gt_15', 'rent_gt_20', 'rent_gt_25', 'rent_gt_30', 'rent_gt_35',
       'rent_gt_40', 'rent_gt_50', 'universe_samples', 'used_samples',
       'hi_mean', 'hi_median', 'hi_stdev', 'hi_sample_weight', 'hi_samples',
       'family_mean', 'family_median', 'family_stdev', 'family_sample_weight',
       'family_samples', 'hc_mortgage_mean', 'hc_mortgage_median',
       'hc_mortgage_stdev', 'hc_mortgage_sample_weight', 'hc_mortgage_samples',
       'hc_mean', 'hc_median', 'hc_stdev', 'hc_samples', 'hc_sample_weight',
       'home_equity_second_mortgage', 'second_mortgage', 'home_equity', 'debt',
       'second_mortgage_cdf', 'home_equity_cdf', 'debt_cdf', 'hs_degree',
       'hs_degree_male', 'hs_degree_female', 'male_age_mean',
       'male_age_median', 'male_age_stdev', 'male_age_sample_weight',
       'male_age_samples', 'female_age_mean', 'female_age_median',
       'female_age_stdev', 'female_age_sample_weight', 'female_age_samples',
       'pct_own', 'married', 'married_snp', 'separated', 'divorced',
       'Bad_Debt', 'pop_density', 'age_median', 'pop_bins'],
      dtype='object')
```

[74]:
```
df_tn['type'].unique()
type_dict={'type':{'City':1,
                   'Urban':2,
                   'Town':3,
                   'CDP':4,
                   'Village':5,
                   'Borough':6}
          }
df_tn.replace(type_dict,inplace=True)
```

[75]:
```
df_tn['type'].unique()
```

[75]:
```
array([1, 2, 3, 4, 5, 6])
```

[76]:
```
df_te.replace(type_dict,inplace=True)
```

[77]:
```
df_te['type'].unique()
```

[77]:
```
array([4, 1, 6, 3, 5, 2])
```

[78]:
```
feature_cols=['COUNTYID','STATEID','zip_code','type','pop', 'family_mean',
         'second_mortgage', 'home_equity', 'debt','hs_degree',
           'age_median','pct_own', 'married','separated', 'divorced']
```

[79]:
```
x_train=df_tn[feature_cols]
y_train=df_tn['hc_mortgage_mean']
```

[80]:
```
x_test=df_te[feature_cols]
y_test=df_te['hc_mortgage_mean']
```

[81]:
```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score,␣
 ↪mean_absolute_error,mean_squared_error,accuracy_score
```

[82]:
```
scaler=StandardScaler()
x_train_sc=scaler.fit_transform(x_train)
x_test_sc=scaler.fit_transform(x_test)
```

[83]:
```
regg = LinearRegression()
regg.fit(x_train_sc,y_train)
```

[83]:
```
LinearRegression()
```

[84]:
```
y_pred=regg.predict(x_test_sc)
```

[85]:
```
y_pred
```

[85]:
```
array([ 857.27639218, 1603.19565511, 1063.78265054, …, 1919.0665886 ,
       1513.12412525, 1146.41314792])
```

[86]:
```
print("r2 score of model")
r2_score(y_test,y_pred)
```

```
r2 score of model
```

[86]:
```
0.7348210754610929
```

[87]:
```
print("Overall RMSE of linear regression model", np.
 ↪sqrt(mean_squared_error(y_test,y_pred)))
```

```
Overall RMSE of linear regression model 323.1018894984635
```

Model has high RMSE and high r2 score which shows model perform well.

[88]:
```
#Run another model at State level. There are 52 states in USA.
```

[89]:
```
state=df_tn["STATEID"].unique()
state[0:6]
```

[89]:
```
array([36, 18, 72, 20,  1, 48])
```

[90]:
```
for i in [20,1,45]:
    print("State ID-",i)

    x_train_nation=df_tn[df_tn['COUNTYID']==i][feature_cols]
    y_train_nation=df_tn[df_tn['COUNTYID']==i]['hc_mortgage_mean']

    x_test_nation=df_te[df_te['COUNTYID']==i][feature_cols]
    y_test_nation=df_te[df_te['COUNTYID']==i]['hc_mortgage_mean']

    x_train_scaled_nation=scaler.fit_transform(x_train_nation)
    x_test_scaled_nation=scaler.fit_transform(x_test_nation)
    regg.fit(x_train_scaled_nation,y_train_nation)
    y_pred_nation=regg.predict(x_test_scaled_nation)

    print("Overall R2 score of linear regression model for state,",i,":-"␣
↪,r2_score(y_test_nation,y_pred_nation))
    print("Overall RMSE of linear regression model for state,",i,":-" ,np.
↪sqrt(mean_squared_error(y_test_nation,y_pred_nation)))
    print("\n")
```

```
State ID- 20
Overall R2 score of linear regression model for state, 20 :- 0.6046603766461811
Overall RMSE of linear regression model for state, 20 :- 307.9718899931471


State ID- 1
Overall R2 score of linear regression model for state, 1 :- 0.8104382475484617
Overall RMSE of linear regression model for state, 1 :- 307.8275861848434


State ID- 45
Overall R2 score of linear regression model for state, 45 :- 0.7887446497855253
Overall RMSE of linear regression model for state, 45 :- 225.69615420724125
```

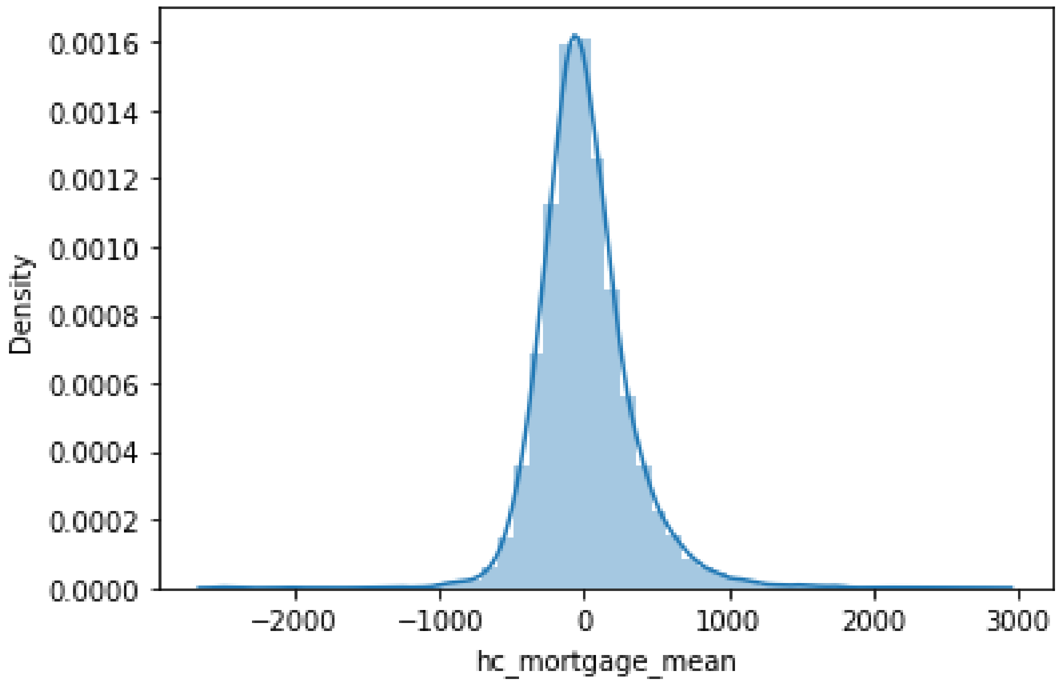
[91]:
```
#To check the residuals
```

[92]:
```
residual= y_test-y_pred
residual
```

[92]:
```
UID
255504    281.969088
252676    -69.935775
276314    190.761969
248614   -157.290627
286865     -9.887017
             …
238088    -67.541646
242811    -41.578757
250127   -127.427569
241096   -330.820475
287763    217.760642
Name: hc_mortgage_mean, Length: 11709, dtype: float64
```

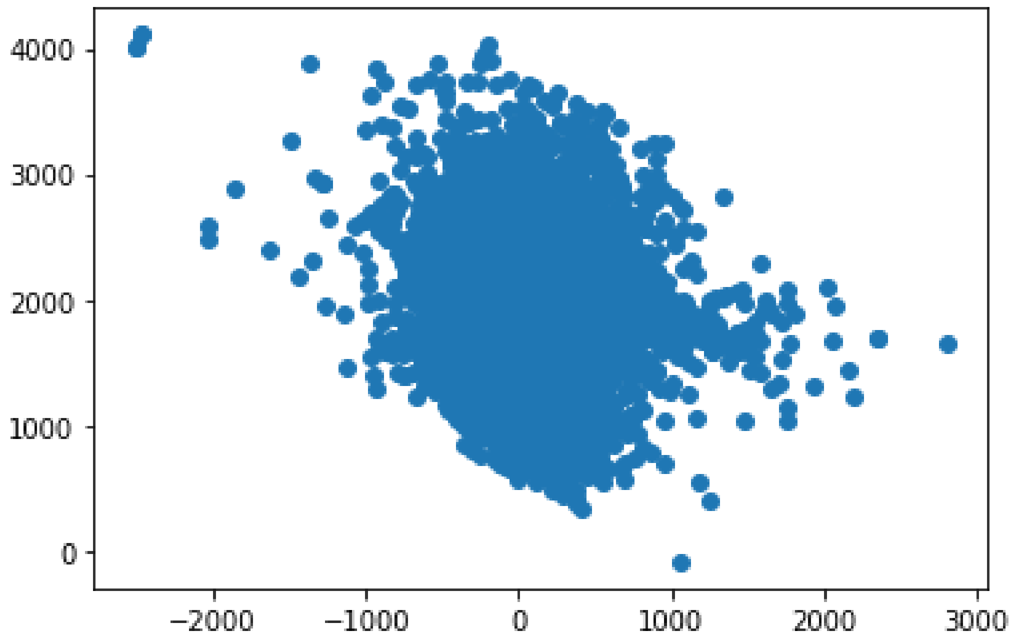
[93]:
```
sns.distplot(residual)
```

```
[93]: <AxesSubplot:xlabel='hc_mortgage_mean', ylabel='Density'>
```



```
[94]: plt.scatter(residual,y_pred)
      # Independance of residuals
```

```
[94]: <matplotlib.collections.PathCollection at 0x7fd4ae77f250>
```



```
[95]: ##End
```