Real Estate Course-end Project 1 By Pavan Lande

February 24, 2023

0.0.1 PROJECT BY:-> PAVAN LANDE

0.0.2 Real Estate.

Course-end Project 1

DESCRIPTION

Problem Statement

A banking institution requires actionable insights into mortgage-backed securities, geographic business investment, and real estate analysis. The mortgage bank would like to identify potential monthly mortgage expenses for each region based on monthly family income and rental of the real estate. A statistical model needs to be created to predict the potential demand in dollars amount of loan for each of the region in the USA. Also, there is a need to create a dashboard which would refresh periodically post data retrieval from the agencies. The dashboard must demonstrate relationships and trends for the key metrics as follows: number of loans, average rental income, monthly mortgage and owner's cost, family income vs mortgage cost comparison across different regions. The metrics described here do not limit the dashboard to these few. Dataset Description

Variables

Description Second mortgage Households with a second mortgage statistics Home equity Households with a home equity loan statistics Debt Households with any type of debt statistics Mortgage Costs Statistics regarding mortgage payments, home equity loans, utilities, and property taxes Home Owner Costs Sum of utilities, and property taxes statistics Gross Rent Contract rent plus the estimated average monthly cost of utility features High school Graduation High school graduation statistics Population Demographics Population demographics statistics Age Demographics Age demographic statistics Household Income Total income of people residing in the household Family Income Total income of people related to the householder

1 Project Task: Week 1

1.1 1. Data Import and Preparation:

[1]:
```
import time
import random
from math import *
import operator
```

```python
import pandas as pd
import numpy as np
# import plotting libraries
import matplotlib
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
%matplotlib inline
import seaborn as sns
sns.set(style="white", color_codes=True)
sns.set(font_scale=1.5)
```

[2]:
```python
df_train=pd.read_csv(r'C:\DATA SCIENCE CLASS\Online Class Lectures\DATA SCIENCE␣
↪CLASS SEP-22 COHORT\Data Science Job Guarantee Bootcamp Capstone\Simplilearn␣
↪DATA Set\Project_1\Project 1\train.csv')
```

[3]:
```python
df_test=pd.read_csv(r'C:\DATA SCIENCE CLASS\Online Class Lectures\DATA SCIENCE␣
↪CLASS SEP-22 COHORT\Data Science Job Guarantee Bootcamp Capstone\Simplilearn␣
↪DATA Set\Project_1\Project 1\test.csv')
```

[4]:
```python
df_train.columns
```

[4]:
```
Index(['UID', 'BLOCKID', 'SUMLEVEL', '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'],
      dtype='object')
```

[5]:
```python
df_test.columns
```

[5]:
```
Index(['UID', 'BLOCKID', 'SUMLEVEL', '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'],
      dtype='object')
```

[6]:
```
len(df_train)
```

[6]:
```
27321
```

[7]:
```
len(df_test)
```

[7]:
```
11709
```

[8]:
```
df_train.head()
```

[8]:
```
      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]
```

[9]:
```
df_test.head()
```

[9]:
```
      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]
```

[10]:
```
df_train.describe()
```

[10]:
```
                UID  BLOCKID  SUMLEVEL      COUNTYID       STATEID  \
count  27321.000000      0.0   27321.0  27321.000000  27321.000000
```

mean	257331.996303	NaN	140.0	85.646426	28.271806
std	21343.859725	NaN	0.0	98.333097	16.392846
min	220342.000000	NaN	140.0	1.000000	1.000000
25%	238816.000000	NaN	140.0	29.000000	13.000000
50%	257220.000000	NaN	140.0	63.000000	28.000000
75%	275818.000000	NaN	140.0	109.000000	42.000000
max	294334.000000	NaN	140.0	840.000000	72.000000

	zip_code	area_code	lat	lng	ALand \
count	27321.000000	27321.000000	27321.000000	27321.000000	2.732100e+04
mean	50081.999524	596.507668	37.508813	-91.288394	1.295106e+08
std	29558.115660	232.497482	5.588268	16.343816	1.275531e+09
min	602.000000	201.000000	17.929085	-165.453872	4.113400e+04
25%	26554.000000	405.000000	33.899064	-97.816067	1.799408e+06
50%	47715.000000	614.000000	38.755183	-86.554374	4.866940e+06
75%	77093.000000	801.000000	41.380606	-79.782503	3.359820e+07
max	99925.000000	989.000000	67.074017	-65.379332	1.039510e+11

	…	female_age_mean	female_age_median	female_age_stdev \
count	…	27115.000000	27115.000000	27115.000000
mean	…	40.319803	40.355099	22.178745
std	…	5.886317	8.039585	2.540257
min	…	16.008330	13.250000	0.556780
25%	…	36.892050	34.916670	21.312135
50%	…	40.373320	40.583330	22.514410
75%	…	43.567120	45.416670	23.575260
max	…	79.837390	82.250000	30.241270

	female_age_sample_weight	female_age_samples	pct_own \
count	27115.000000	27115.000000	27053.000000
mean	544.238432	2208.761903	0.640434
std	283.546896	1089.316999	0.226640
min	0.664700	2.000000	0.000000
25%	355.995825	1471.000000	0.502780
50%	503.643890	2066.000000	0.690840
75%	680.275055	2772.000000	0.817460
max	6197.995200	27250.000000	1.000000

	married	married_snp	separated	divorced
count	27130.000000	27130.000000	27130.000000	27130.000000
mean	0.508300	0.047537	0.019089	0.100248
std	0.136860	0.037640	0.020796	0.049055
min	0.000000	0.000000	0.000000	0.000000
25%	0.425102	0.020810	0.004530	0.065800
50%	0.526665	0.038840	0.013460	0.095205
75%	0.605760	0.065100	0.027488	0.129000
max	1.000000	0.714290	0.714290	1.000000

```
[8 rows x 74 columns]
```

[11]:
```
df_test.describe()
```

[11]:
```
                 UID  BLOCKID  SUMLEVEL      COUNTYID       STATEID  \
count   11709.000000      0.0   11709.0  11709.000000  11709.000000
mean   257525.004783      NaN     140.0     85.710650     28.489196
std     21466.372658      NaN       0.0     99.304334     16.607262
min    220336.000000      NaN     140.0      1.000000      1.000000
25%    238819.000000      NaN     140.0     29.000000     13.000000
50%    257651.000000      NaN     140.0     61.000000     28.000000
75%    276300.000000      NaN     140.0    109.000000     42.000000
max    294333.000000      NaN     140.0    810.000000     72.000000

           zip_code     area_code           lat           lng         ALand  \
count  11709.000000  11709.000000  11709.000000  11709.000000  1.170900e+04
mean   50123.418396    593.598514     37.405491    -91.340229  1.095500e+08
std    29775.134038    232.074263      5.625904     16.407818  7.624940e+08
min      601.000000    201.000000     17.965835   -166.770979  8.299000e+03
25%    25570.000000    404.000000     33.919813    -97.816561  1.718660e+06
50%    47362.000000    612.000000     38.618093    -86.643344  4.835000e+06
75%    77406.000000    787.000000     41.232973    -79.697311  3.204540e+07
max    99929.000000    989.000000     64.804269    -65.695344  5.520166e+10

       …  female_age_mean  female_age_median  female_age_stdev  \
count  …     11613.000000       11613.000000      11613.000000
mean   …        40.111999          40.131864         22.148145
std    …         5.851192           7.972026          2.554907
min    …        15.360240          12.833330          0.737110
25%    …        36.729210          34.750000         21.270920
50%    …        40.196960          40.333330         22.472990
75%    …        43.496490          45.333330         23.549450
max    …        90.107940          90.166670         29.626680

       female_age_sample_weight  female_age_samples       pct_own  \
count              11613.000000        11613.000000  11587.000000
mean                 550.411243         2233.003186      0.634194
std                  280.992521         1072.017063      0.232232
min                    0.251910            3.000000      0.000000
25%                  363.225840         1499.000000      0.492500
50%                  509.103610         2099.000000      0.687640
75%                  685.883910         2800.000000      0.815235
max                 4145.557870        15466.000000      1.000000

            married   married_snp     separated      divorced
count  11625.000000  11625.000000  11625.000000  11625.000000
```

```
mean      0.505632    0.047960    0.019346    0.099191
std       0.139774    0.038693    0.021428    0.048525
min       0.000000    0.000000    0.000000    0.000000
25%       0.422020    0.020890    0.004500    0.064590
50%       0.525270    0.038680    0.013870    0.094350
75%       0.605660    0.065340    0.027910    0.128400
max       1.000000    0.714290    0.714290    0.362750

[8 rows x 74 columns]
```

[12]:

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27321 entries, 0 to 27320
Data columns (total 80 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   UID                          27321 non-null  int64
 1   BLOCKID                      0 non-null      float64
 2   SUMLEVEL                     27321 non-null  int64
 3   COUNTYID                     27321 non-null  int64
 4   STATEID                      27321 non-null  int64
 5   state                        27321 non-null  object
 6   state_ab                     27321 non-null  object
 7   city                         27321 non-null  object
 8   place                        27321 non-null  object
 9   type                         27321 non-null  object
 10  primary                      27321 non-null  object
 11  zip_code                     27321 non-null  int64
 12  area_code                    27321 non-null  int64
 13  lat                          27321 non-null  float64
 14  lng                          27321 non-null  float64
 15  ALand                        27321 non-null  float64
 16  AWater                       27321 non-null  int64
 17  pop                          27321 non-null  int64
 18  male_pop                     27321 non-null  int64
 19  female_pop                   27321 non-null  int64
 20  rent_mean                    27007 non-null  float64
 21  rent_median                  27007 non-null  float64
 22  rent_stdev                   27007 non-null  float64
 23  rent_sample_weight           27007 non-null  float64
 24  rent_samples                 27007 non-null  float64
 25  rent_gt_10                   27007 non-null  float64
 26  rent_gt_15                   27007 non-null  float64
 27  rent_gt_20                   27007 non-null  float64
 28  rent_gt_25                   27007 non-null  float64
 29  rent_gt_30                   27007 non-null  float64
```

```
30  rent_gt_35                   27007 non-null  float64
31  rent_gt_40                   27007 non-null  float64
32  rent_gt_50                   27007 non-null  float64
33  universe_samples             27321 non-null  int64
34  used_samples                 27321 non-null  int64
35  hi_mean                      27053 non-null  float64
36  hi_median                    27053 non-null  float64
37  hi_stdev                     27053 non-null  float64
38  hi_sample_weight             27053 non-null  float64
39  hi_samples                   27053 non-null  float64
40  family_mean                  27023 non-null  float64
41  family_median                27023 non-null  float64
42  family_stdev                 27023 non-null  float64
43  family_sample_weight         27023 non-null  float64
44  family_samples               27023 non-null  float64
45  hc_mortgage_mean             26748 non-null  float64
46  hc_mortgage_median           26748 non-null  float64
47  hc_mortgage_stdev            26748 non-null  float64
48  hc_mortgage_sample_weight    26748 non-null  float64
49  hc_mortgage_samples          26748 non-null  float64
50  hc_mean                      26721 non-null  float64
51  hc_median                    26721 non-null  float64
52  hc_stdev                     26721 non-null  float64
53  hc_samples                   26721 non-null  float64
54  hc_sample_weight             26721 non-null  float64
55  home_equity_second_mortgage  26864 non-null  float64
56  second_mortgage              26864 non-null  float64
57  home_equity                  26864 non-null  float64
58  debt                         26864 non-null  float64
59  second_mortgage_cdf          26864 non-null  float64
60  home_equity_cdf              26864 non-null  float64
61  debt_cdf                     26864 non-null  float64
62  hs_degree                    27131 non-null  float64
63  hs_degree_male               27121 non-null  float64
64  hs_degree_female             27098 non-null  float64
65  male_age_mean                27132 non-null  float64
66  male_age_median              27132 non-null  float64
67  male_age_stdev               27132 non-null  float64
68  male_age_sample_weight       27132 non-null  float64
69  male_age_samples             27132 non-null  float64
70  female_age_mean              27115 non-null  float64
71  female_age_median            27115 non-null  float64
72  female_age_stdev             27115 non-null  float64
73  female_age_sample_weight     27115 non-null  float64
74  female_age_samples           27115 non-null  float64
75  pct_own                      27053 non-null  float64
76  married                      27130 non-null  float64
77  married_snp                  27130 non-null  float64
```

```
 78  separated                    27130 non-null  float64
 79  divorced                     27130 non-null  float64
dtypes: float64(62), int64(12), object(6)
memory usage: 16.7+ MB
```

[13]:
```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11709 entries, 0 to 11708
Data columns (total 80 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   UID                          11709 non-null  int64
 1   BLOCKID                      0 non-null      float64
 2   SUMLEVEL                     11709 non-null  int64
 3   COUNTYID                     11709 non-null  int64
 4   STATEID                      11709 non-null  int64
 5   state                        11709 non-null  object
 6   state_ab                     11709 non-null  object
 7   city                         11709 non-null  object
 8   place                        11709 non-null  object
 9   type                         11709 non-null  object
 10  primary                      11709 non-null  object
 11  zip_code                     11709 non-null  int64
 12  area_code                    11709 non-null  int64
 13  lat                          11709 non-null  float64
 14  lng                          11709 non-null  float64
 15  ALand                        11709 non-null  int64
 16  AWater                       11709 non-null  int64
 17  pop                          11709 non-null  int64
 18  male_pop                     11709 non-null  int64
 19  female_pop                   11709 non-null  int64
 20  rent_mean                    11561 non-null  float64
 21  rent_median                  11561 non-null  float64
 22  rent_stdev                   11561 non-null  float64
 23  rent_sample_weight           11561 non-null  float64
 24  rent_samples                 11561 non-null  float64
 25  rent_gt_10                   11560 non-null  float64
 26  rent_gt_15                   11560 non-null  float64
 27  rent_gt_20                   11560 non-null  float64
 28  rent_gt_25                   11560 non-null  float64
 29  rent_gt_30                   11560 non-null  float64
 30  rent_gt_35                   11560 non-null  float64
 31  rent_gt_40                   11560 non-null  float64
 32  rent_gt_50                   11560 non-null  float64
 33  universe_samples             11709 non-null  int64
 34  used_samples                 11709 non-null  int64
 35  hi_mean                      11587 non-null  float64
```

```
 36  hi_median                    11587 non-null  float64
 37  hi_stdev                     11587 non-null  float64
 38  hi_sample_weight             11587 non-null  float64
 39  hi_samples                   11587 non-null  float64
 40  family_mean                  11573 non-null  float64
 41  family_median                11573 non-null  float64
 42  family_stdev                 11573 non-null  float64
 43  family_sample_weight         11573 non-null  float64
 44  family_samples               11573 non-null  float64
 45  hc_mortgage_mean             11441 non-null  float64
 46  hc_mortgage_median           11441 non-null  float64
 47  hc_mortgage_stdev            11441 non-null  float64
 48  hc_mortgage_sample_weight    11441 non-null  float64
 49  hc_mortgage_samples          11441 non-null  float64
 50  hc_mean                      11419 non-null  float64
 51  hc_median                    11419 non-null  float64
 52  hc_stdev                     11419 non-null  float64
 53  hc_samples                   11419 non-null  float64
 54  hc_sample_weight             11419 non-null  float64
 55  home_equity_second_mortgage  11489 non-null  float64
 56  second_mortgage              11489 non-null  float64
 57  home_equity                  11489 non-null  float64
 58  debt                         11489 non-null  float64
 59  second_mortgage_cdf          11489 non-null  float64
 60  home_equity_cdf              11489 non-null  float64
 61  debt_cdf                     11489 non-null  float64
 62  hs_degree                    11624 non-null  float64
 63  hs_degree_male               11620 non-null  float64
 64  hs_degree_female             11604 non-null  float64
 65  male_age_mean                11625 non-null  float64
 66  male_age_median              11625 non-null  float64
 67  male_age_stdev               11625 non-null  float64
 68  male_age_sample_weight       11625 non-null  float64
 69  male_age_samples             11625 non-null  float64
 70  female_age_mean              11613 non-null  float64
 71  female_age_median            11613 non-null  float64
 72  female_age_stdev             11613 non-null  float64
 73  female_age_sample_weight     11613 non-null  float64
 74  female_age_samples           11613 non-null  float64
 75  pct_own                      11587 non-null  float64
 76  married                      11625 non-null  float64
 77  married_snp                  11625 non-null  float64
 78  separated                    11625 non-null  float64
 79  divorced                     11625 non-null  float64
dtypes: float64(61), int64(13), object(6)
memory usage: 7.1+ MB
```

1.2 2. Figure out the primary key and look for the requirement of indexing

[14]:
```
#UID is unique userID value in the train and test dataset. So an index can be␣
↪created from the UID feature
df_train.set_index(keys=['UID'],inplace=True)#Set the DataFrame index using␣
↪existing columns.
df_test.set_index(keys=['UID'],inplace=True)
```

[15]:
```
df_train.head(2)
```

[15]:
```
        BLOCKID  SUMLEVEL  COUNTYID  STATEID     state state_ab        city  \
UID
267822      NaN       140        53       36  New York       NY    Hamilton
246444      NaN       140       141       18   Indiana       IN  South Bend

            place  type primary  …  female_age_mean  female_age_median  \
UID                              …
267822  Hamilton  City   tract  …         44.48629           45.33333
246444  Roseland  City   tract  …         36.48391           37.58333

        female_age_stdev  female_age_sample_weight  female_age_samples  \
UID
267822          22.51276                 685.33845              2618.0
246444          23.43353                 267.23367              1284.0

        pct_own  married  married_snp  separated  divorced
UID
267822  0.79046  0.57851      0.01882    0.01240    0.0877
246444  0.52483  0.34886      0.01426    0.01426    0.0903

[2 rows x 79 columns]
```

[16]:
```
df_test.head(2)
```

[16]:
```
        BLOCKID  SUMLEVEL  COUNTYID  STATEID     state state_ab     city  \
UID
255504      NaN       140       163       26  Michigan       MI  Detroit
252676      NaN       140         1       23     Maine       ME   Auburn

                          place  type primary  …  female_age_mean  \
UID                                            …
255504  Dearborn Heights City   CDP   tract  …         34.78682
252676            Auburn City  City   tract  …         44.23451

        female_age_median  female_age_stdev  female_age_sample_weight  \
UID
255504           33.75000          21.58531                 416.48097
```

```
252676           46.66667          22.37036                    532.03505

        female_age_samples  pct_own  married  married_snp  separated  divorced
UID
255504              1938.0  0.70252  0.28217      0.05910    0.03813   0.14299
252676              1950.0  0.85128  0.64221      0.02338    0.00000   0.13377

[2 rows x 79 columns]
```

1.3 3. 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.

[17]:

```
# Percentage of missing values in train set
missing_list_train=df_train.isnull().sum() *100/len(df_train)
missing_values_df_train=pd.DataFrame(missing_list_train,columns=['Percantage of␣
↪missing values'])
missing_values_df_train.sort_values(by=['Percantage of missing␣
↪values'],inplace=True,ascending=False)
missing_values_df_train[missing_values_df_train['Percantage of missing values']␣
↪>0][:10]
#BLOCKID can be dropped, since it is 100%missing values
```

[17]:

```
                              Percantage of missing values
BLOCKID                                         100.000000
hc_samples                                        2.196113
hc_mean                                           2.196113
hc_median                                         2.196113
hc_stdev                                          2.196113
hc_sample_weight                                  2.196113
hc_mortgage_mean                                  2.097288
hc_mortgage_stdev                                 2.097288
hc_mortgage_sample_weight                         2.097288
hc_mortgage_samples                               2.097288
```

[18]:

```
#percantage of missing values in test set
missing_list_test=df_test.isnull().sum() *100/len(df_train)
missing_values_df_test=pd.DataFrame(missing_list_test,columns=['Percantage of␣
↪missing values'])
missing_values_df_test.sort_values(by=['Percantage of missing␣
↪values'],inplace=True,ascending=False)
missing_values_df_test[missing_values_df_test['Percantage of missing values']␣
↪>0][:10]
#BLOCKID can be dropped, since it is 43%missing values
```

[18]:

	Percantage of missing values
BLOCKID	42.857143
hc_samples	1.061455
hc_mean	1.061455
hc_median	1.061455
hc_stdev	1.061455
hc_sample_weight	1.061455
hc_mortgage_mean	0.980930
hc_mortgage_stdev	0.980930
hc_mortgage_sample_weight	0.980930
hc_mortgage_samples	0.980930

[19]:
```
df_train .drop(columns=['BLOCKID','SUMLEVEL'],inplace=True) #SUMLEVEL doest not␣
↪have any predictive power and no variable
```

[20]:
```
df_test .drop(columns=['BLOCKID','SUMLEVEL'],inplace=True) #SUMLEVEL doest not␣
↪have any predictive power
```

[21]:
```
# Imputing missing values with mean
missing_train_cols=[]
for col in df_train.columns:
     if df_train[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']
```

[22]:
```
# Imputing missing values with mean
missing_test_cols=[]
for col in df_test.columns:
     if df_test[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']
```

[23]:
```
# Missing cols are all numerical variables
for col in df_train.columns:
     if col in (missing_train_cols):
             df_train[col].replace(np.nan, df_train[col].mean(),inplace=True)
```

[24]:
```
# Missing cols are all numerical variables
for col in df_test.columns:
     if col in (missing_test_cols):
             df_test[col].replace(np.nan, df_test[col].mean(),inplace=True)
```

[25]:
```
df_train.isna().sum().sum()
```

[25]: 0

[26]:
```
df_test.isna().sum().sum()
```

[26]: 0

1.4 Exploratory Data Analysis (EDA):

4.Perform debt analysis. You may take the following steps:

a) 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

[27]:
```
pip install pandasql
```

```
Requirement already satisfied: pandasql in c:\users\pavan
lande\anaconda3\lib\site-packages (0.7.3)Note: you may need to restart the
kernel to use updated packages.

Requirement already satisfied: numpy in c:\users\pavan lande\anaconda3\lib\site-
```

```
packages (from pandasql) (1.21.5)
Requirement already satisfied: sqlalchemy in c:\users\pavan
lande\anaconda3\lib\site-packages (from pandasql) (1.4.32)
Requirement already satisfied: pandas in c:\users\pavan
lande\anaconda3\lib\site-packages (from pandasql) (1.4.2)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\pavan
lande\anaconda3\lib\site-packages (from pandas->pandasql) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\pavan
lande\anaconda3\lib\site-packages (from pandas->pandasql) (2021.3)
Requirement already satisfied: six>=1.5 in c:\users\pavan
lande\anaconda3\lib\site-packages (from python-
dateutil>=2.8.1->pandas->pandasql) (1.16.0)
Requirement already satisfied: greenlet!=0.4.17 in c:\users\pavan
lande\anaconda3\lib\site-packages (from sqlalchemy->pandasql) (1.1.1)
```

[28]:
```
from pandasql import sqldf
q1 = "select place,pct_own,second_mortgage,lat,lng from df_train where pct_own␣
↪>0.10 and second_mortgage <0.5 order by second_mortgage DESC LIMIT 2500;"
pysqldf = lambda q: sqldf(q, globals())
df_train_location_mort_pct=pysqldf(q1)
```

[29]:
```
df_train_location_mort_pct.head()
```

[29]:

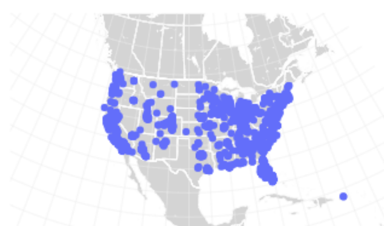
	place	pct_own	second_mortgage	lat	lng
0	Worcester City	0.20247	0.43363	42.254262	-71.800347
1	Harbor Hills	0.15618	0.31818	40.751809	-73.853582
2	Glen Burnie	0.22380	0.30212	39.127273	-76.635265
3	Egypt Lake-leto	0.11618	0.28972	28.029063	-82.495395
4	Lincolnwood	0.14228	0.28899	41.967289	-87.652434

[30]:
```
import plotly.express as px
import plotly.graph_objects as go
```

[31]:
```
fig = go.Figure(data=go.Scattergeo(
    lat = df_train_location_mort_pct['lat'],
    lon = df_train_location_mort_pct['lng']),
    )
fig.update_layout(
    geo=dict(
        scope = 'north america',
        showland = True,
        landcolor = "rgb(212, 212, 212)",
        subunitcolor = "rgb(255, 255, 255)",
        countrycolor = "rgb(255, 255, 255)",
        showlakes = True,
        lakecolor = "rgb(255, 255, 255)",
        showsubunits = True,
```

```
        showcountries = True,
        resolution = 50,
        projection = dict(
            type = 'conic conformal',
            rotation_lon = -100
        ),
        lonaxis = dict(
            showgrid = True,
            gridwidth = 0.5,
            range= [ -140.0, -55.0 ],
            dtick = 5
        ),
        lataxis = dict (
            showgrid = True,
            gridwidth = 0.5,
            range= [ 20.0, 60.0 ],
            dtick = 5
        )
    ),
    title='Top 2,500 locations with second mortgage is the highest and percent␣
 ↪ownership is above 10 percent')
fig.show()
```

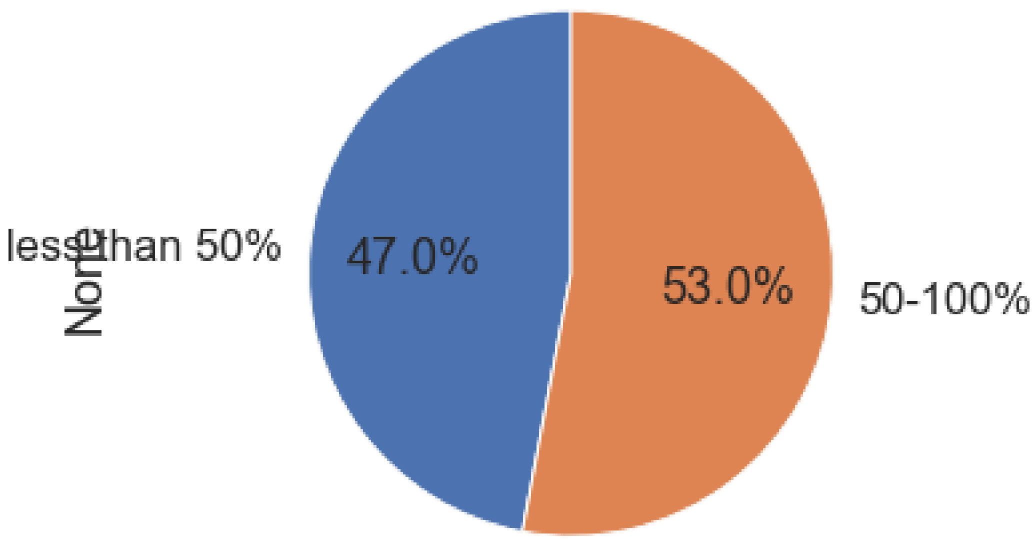
Top 2,500 locations with second mortgage is the highest and percent ownership is above 10 percent



1.4.1 b) 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

[32]:
```
df_train['bad_debt']=df_train['second_mortgage']+df_train['home_equity']-df_train['home_equity
```

[33]:
```
df_train['bins'] = pd.cut(df_train['bad_debt'],bins=[0,0.10,1], labels=["less␣
↪than 50%","50-100%"])
df_train.groupby(['bins']).size().plot(kind='pie',subplots=True,startangle=90,␣
↪autopct='%1.1f%%')
plt.axis('equal')

plt.show()
#df.plot.pie(subplots=True,figsize=(8, 3))
```



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

[34]:
```
cols=[]
df_train.columns
```

[34]:
```
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', 'bins'],
       dtype='object')
```

[35]:

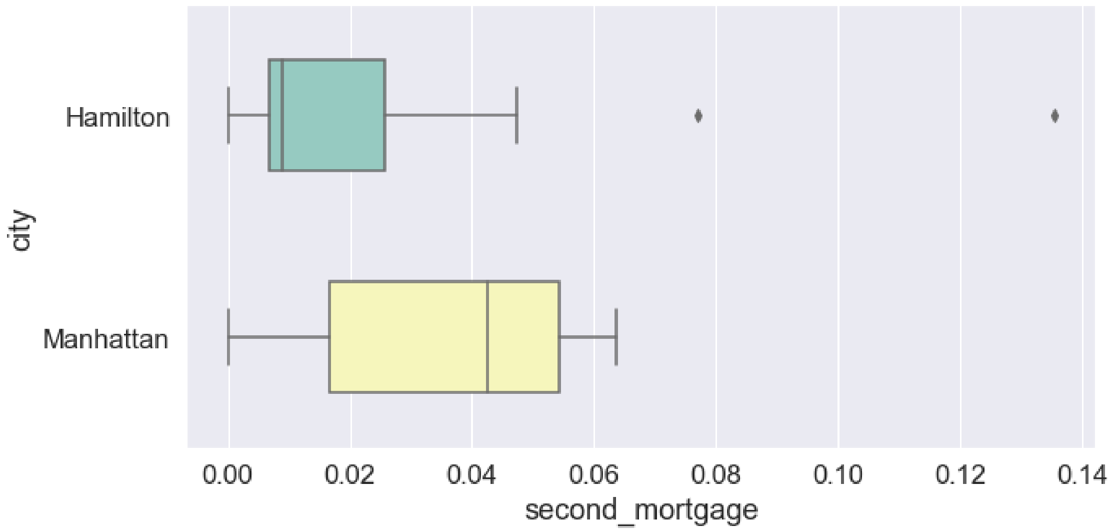
```python
#Taking Hamilton and Manhattan cities data
cols=['second_mortgage','home_equity','debt','bad_debt']
df_box_hamilton=df_train.loc[df_train['city'] == 'Hamilton']
df_box_manhattan=df_train.loc[df_train['city'] == 'Manhattan']
df_box_city=pd.concat([df_box_hamilton,df_box_manhattan])
df_box_city.head(4)
```

[35]:

```
        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

           type primary  zip_code  area_code  …  female_age_stdev  \
UID                                            …
267822     City   tract     13346        315  …          22.51276
263797     City   tract      8610        609  …          24.05831
270979  Village   tract     45015        513  …          22.66500
259028      CDP   tract     39746        662  …          22.79602

        female_age_sample_weight  female_age_samples  pct_own  married  \
UID
267822                 685.33845              2618.0  0.79046  0.57851
263797                 732.58443              3124.0  0.64400  0.56377
270979                 565.32725              2528.0  0.61278  0.47397
259028                 483.01311              1954.0  0.83241  0.58678

        married_snp  separated  divorced  bad_debt           bins
UID
267822      0.01882    0.01240   0.08770   0.09408  less than 50%
263797      0.01980    0.00990   0.04892   0.18071        50-100%
270979      0.04419    0.02663   0.13741   0.15005        50-100%
259028      0.01052    0.00000   0.11721   0.02130  less than 50%

[4 rows x 79 columns]
```

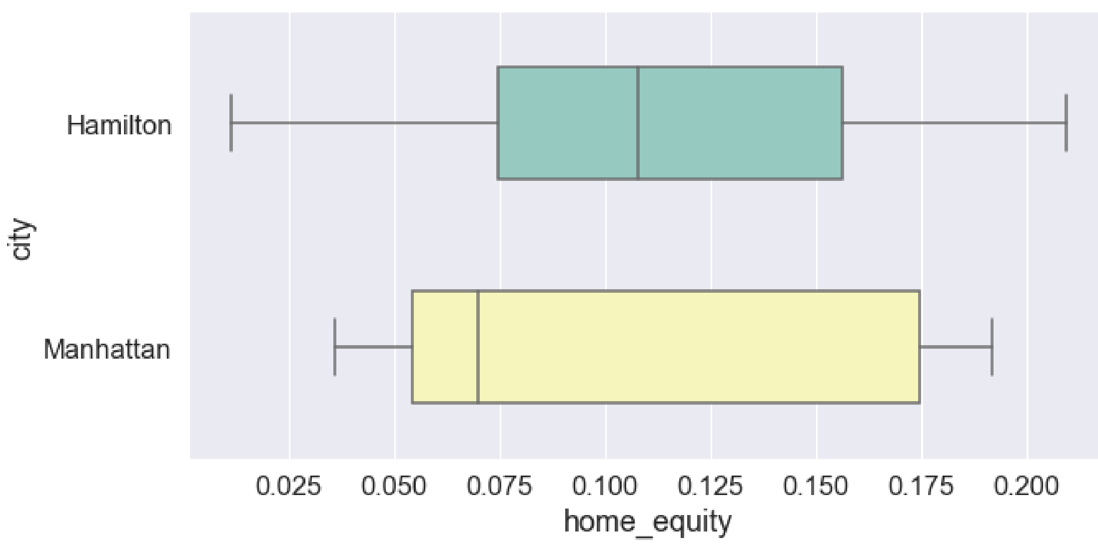
[36]:

```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='second_mortgage', y='city',width=0.
 ↪5,palette="Set3")
plt.show()
```

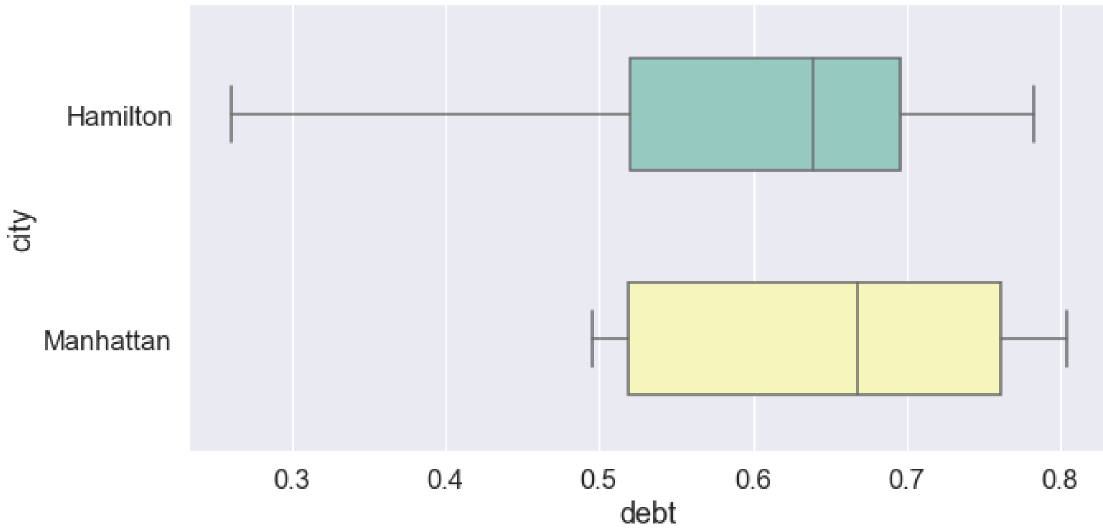


[37]:

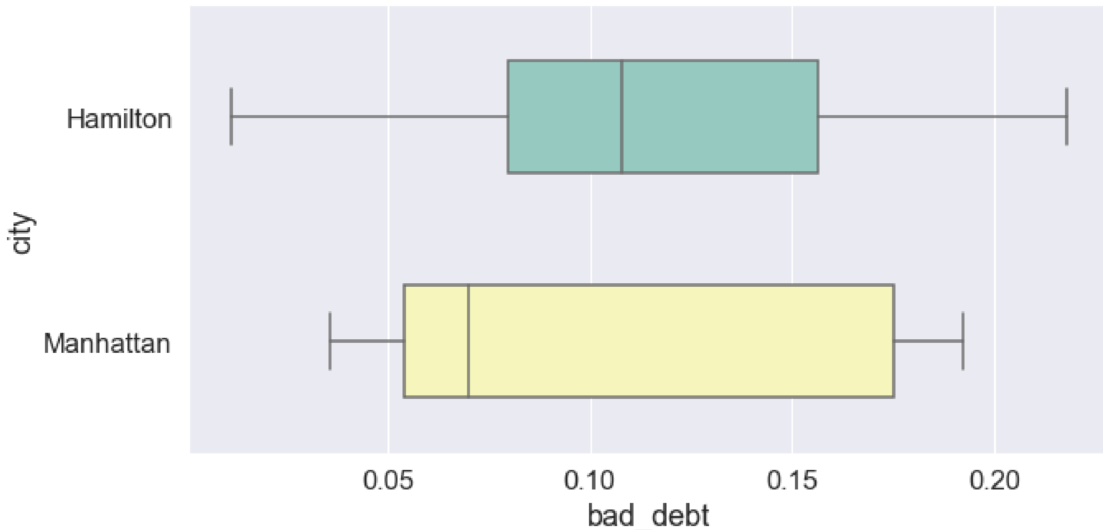
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='home_equity', y='city',width=0.5,palette="Set3")
plt.show()
```



[38]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='debt', y='city',width=0.5,palette="Set3")
plt.show()
```



[39]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='bad_debt', y='city',width=0.5,palette="Set3")
plt.show()
```

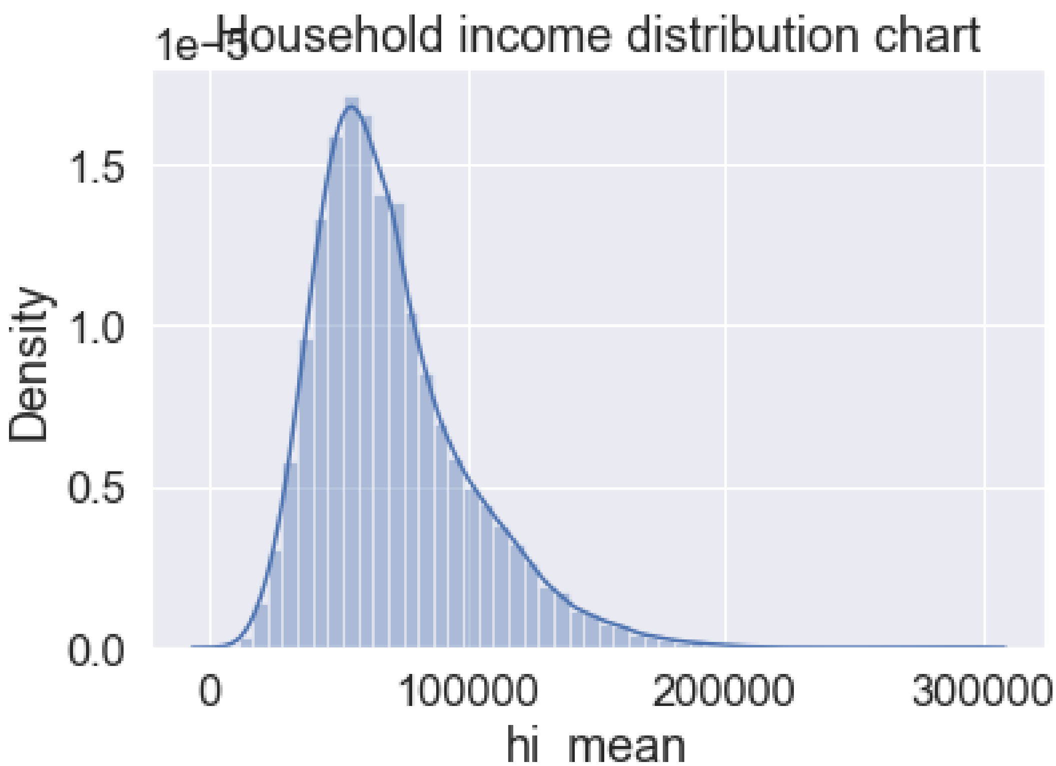


[40]:
```
## Manhattan has higher metrics compared to Hamilton
```

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

[41]:
```
sns.distplot(df_train['hi_mean'])
plt.title('Household income distribution chart')
plt.show()
```

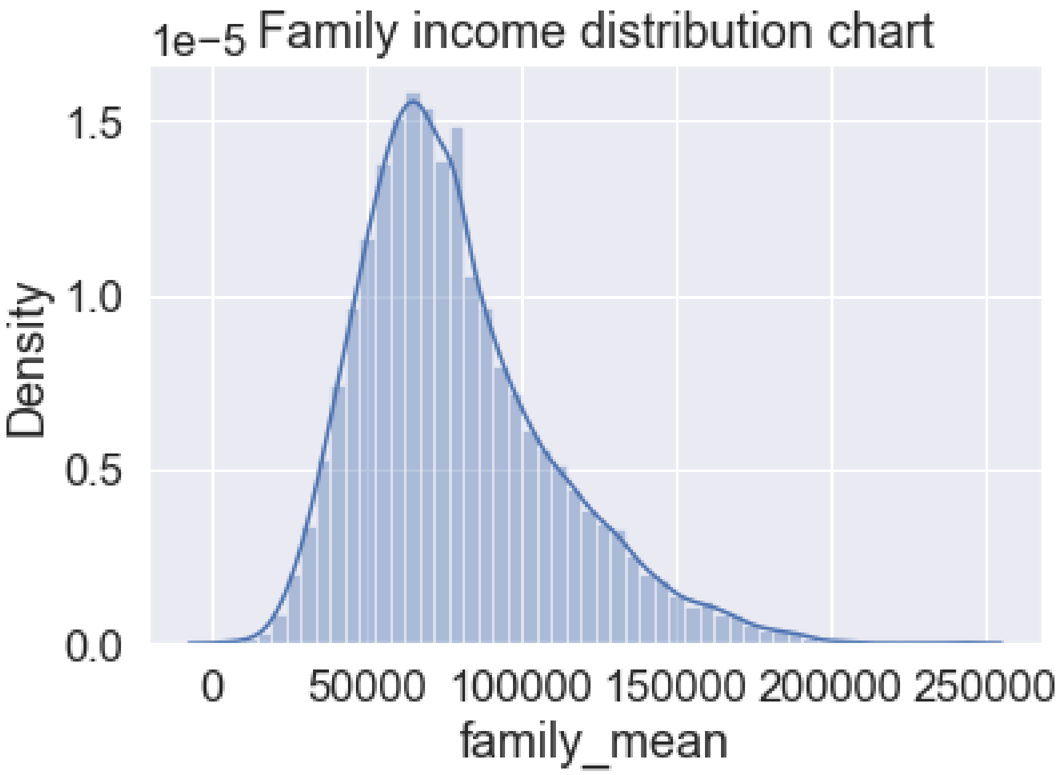
```
C:\Users\Pavan Lande\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning:

`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).
```



[42]:
```
sns.distplot(df_train['family_mean'])
plt.title('Family income distribution chart')
plt.show()
```

```
C:\Users\Pavan Lande\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning:
```

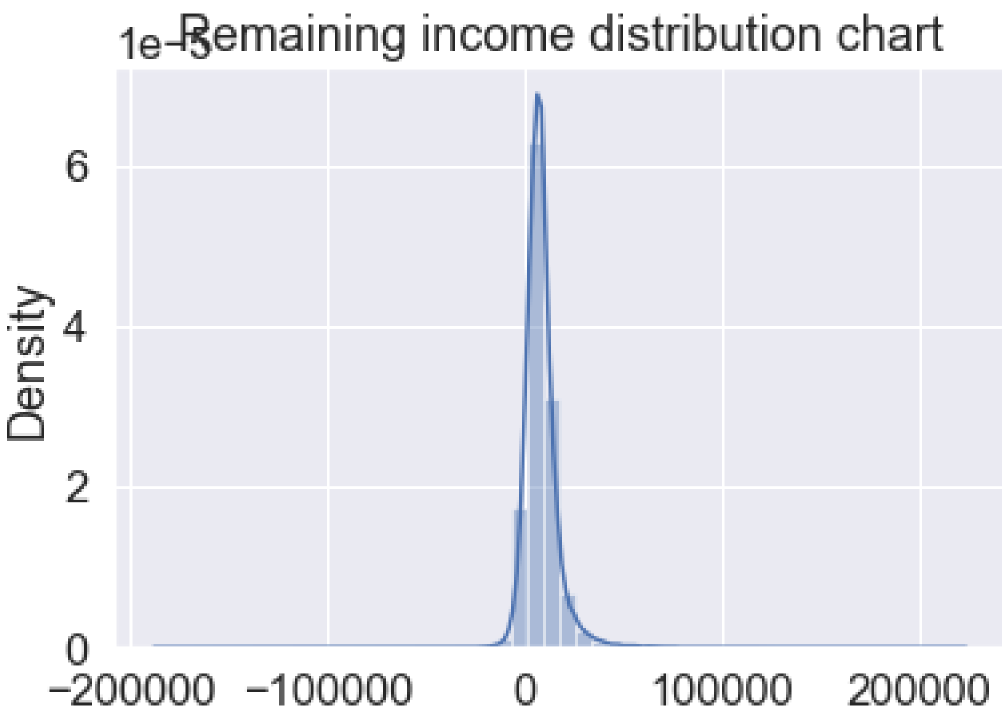
`distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).



[43]:
```
sns.distplot(df_train['family_mean']-df_train['hi_mean'])
plt.title('Remaining income distribution chart')
plt.show()
```

C:\Users\Pavan Lande\anaconda3\lib\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).



[44]:
```
## Income distribution almost has normality in its distrbution
```

1.6 Project Task: Week 2

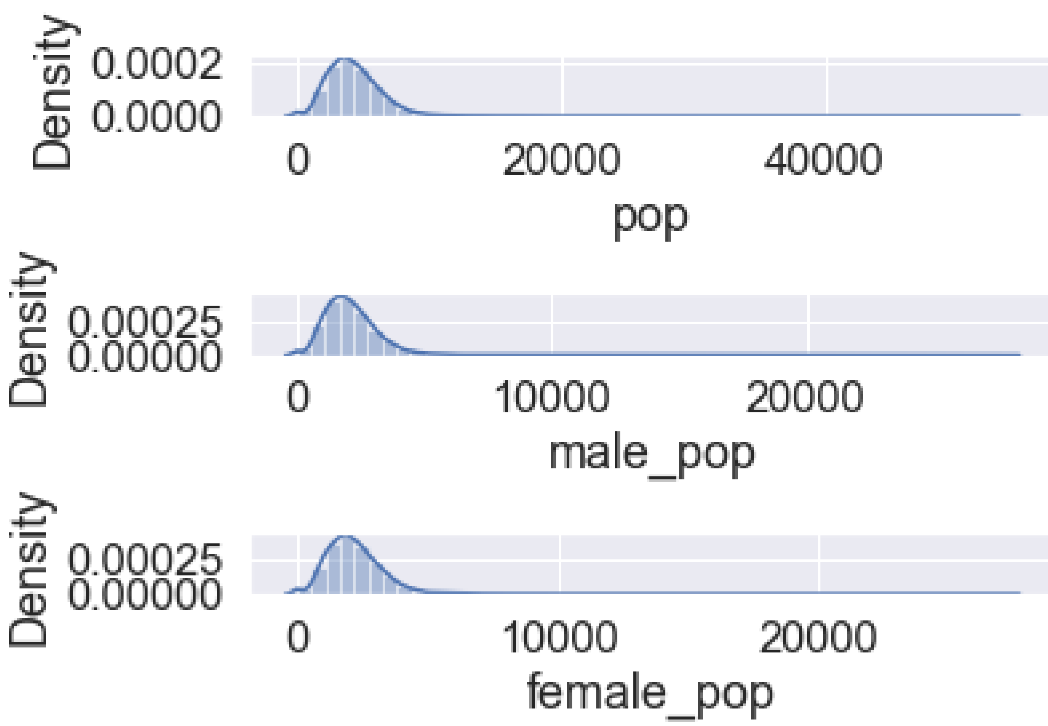
1.7 Exploratory Data Analysis (EDA):

1.8 1. 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):

[45]:
```
#plt.figure(figsize=(25,10))
fig,(ax1,ax2,ax3)=plt.subplots(3,1)
sns.distplot(df_train['pop'],ax=ax1)
sns.distplot(df_train['male_pop'],ax=ax2)
sns.distplot(df_train['female_pop'],ax=ax3)
plt.subplots_adjust(wspace=0.8,hspace=0.8)
plt.tight_layout()
plt.show()
```

```
C:\Users\Pavan Lande\anaconda3\lib\site-packages\seaborn\distributions.py:2619:
FutureWarning:
```

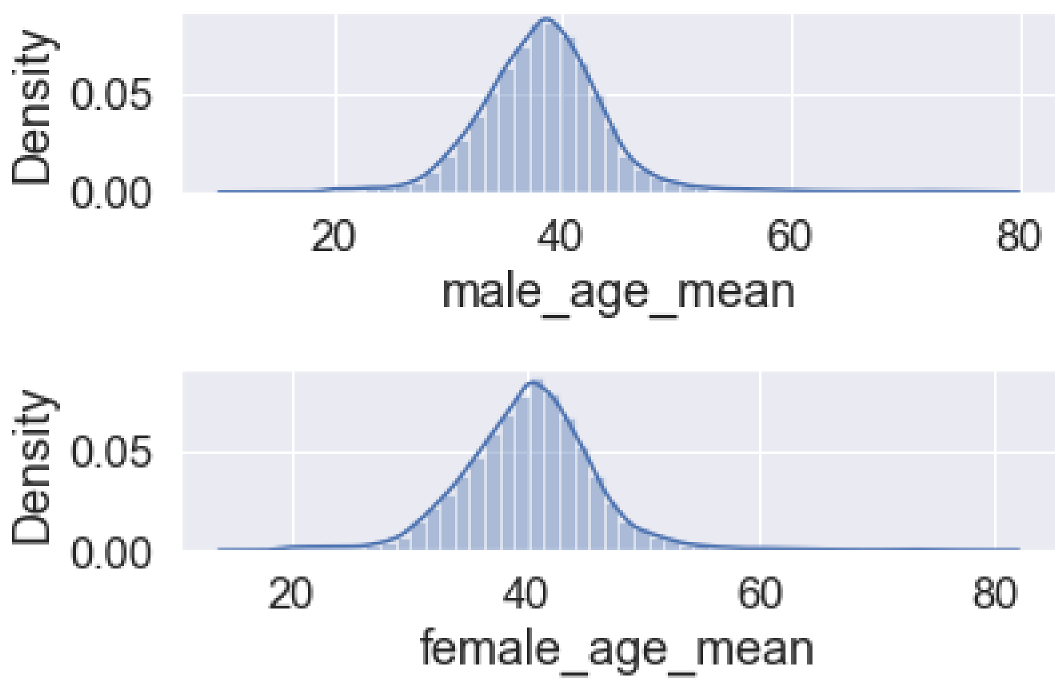
```
`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

C:\Users\Pavan Lande\anaconda3\lib\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).

C:\Users\Pavan Lande\anaconda3\lib\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).
```



[46]:
```
#plt.figure(figsize=(25,10))
fig,(ax1,ax2)=plt.subplots(2,1)
sns.distplot(df_train['male_age_mean'],ax=ax1)
sns.distplot(df_train['female_age_mean'],ax=ax2)
```

```
plt.subplots_adjust(wspace=0.8,hspace=0.8)
plt.tight_layout()
plt.show()
```

```
C:\Users\Pavan Lande\anaconda3\lib\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).

C:\Users\Pavan Lande\anaconda3\lib\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).
```



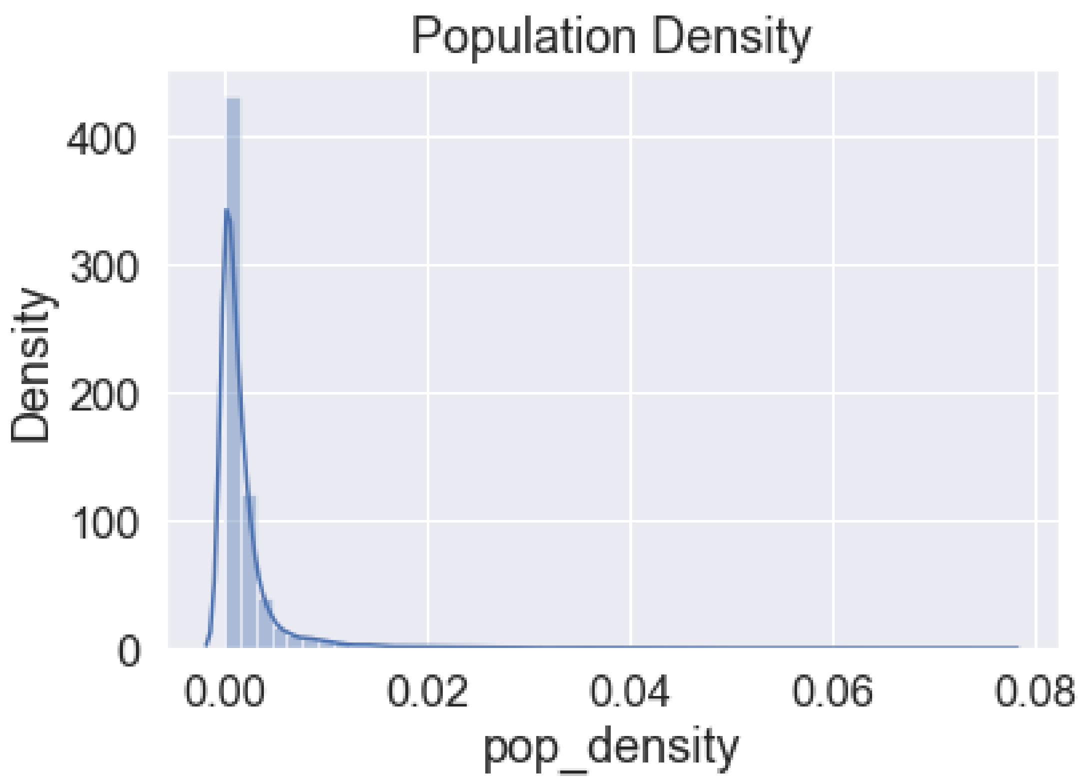
1.8.1 a) Use pop and ALand variables to create a new field called population density

[47]:
```
df_train['pop_density']=df_train['pop']/df_train['ALand']
```

[48]:
```
df_test['pop_density']=df_test['pop']/df_test['ALand']
```

[49]:
```
sns.distplot(df_train['pop_density'])
plt.title('Population Density')
plt.show() # Very less density is noticed
```

```
C:\Users\Pavan Lande\anaconda3\lib\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).
```



1.8.2 b) Use male_age_median, female_age_median, male_pop, and female_pop to create a new field called median age c) Visualize the findings using appropriate chart type

[50]:
```
df_train['age_median']=(df_train['male_age_median']+df_train['female_age_median'])/
↪2
df_test['age_median']=(df_test['male_age_median']+df_test['female_age_median'])/
↪2
```

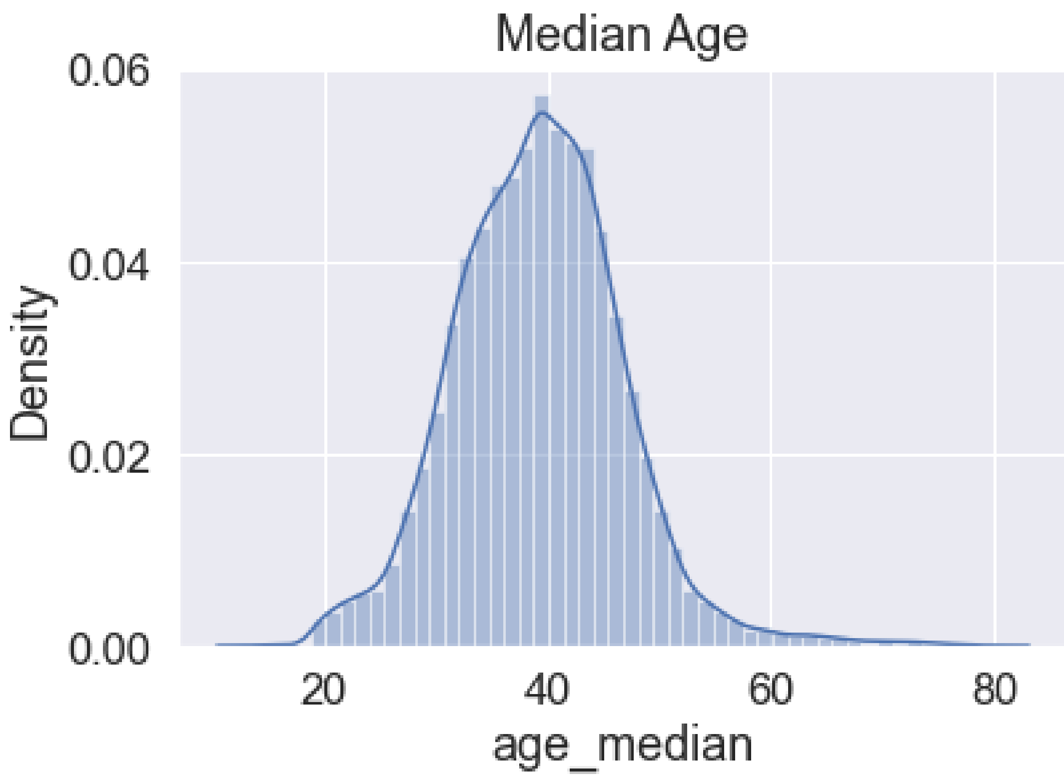
[51]:
```
df_train[['male_age_median','female_age_median','male_pop','female_pop','age_median']].
↪head()
```

[51]:

UID	male_age_median	female_age_median	male_pop	female_pop	age_median
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

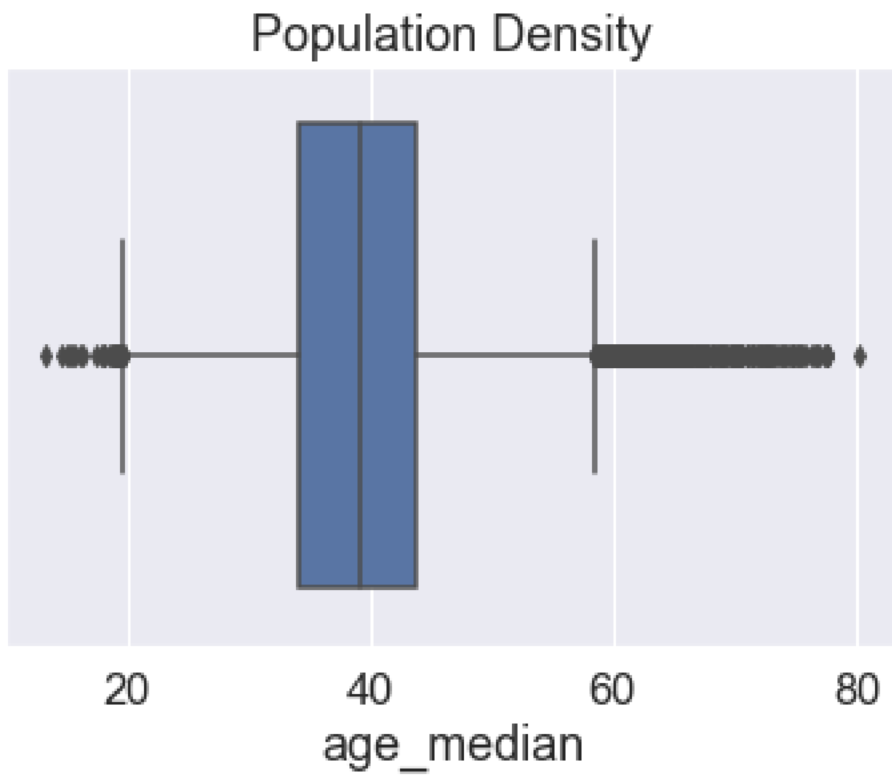
[52]:
```
sns.distplot(df_train['age_median'])
plt.title('Median Age')
plt.show()
# Age of population is mostly between 20 and 60
# Majority are of age around 40
# Median age distribution has a gaussian distribution
# Some right skewness is noticed
```

```
C:\Users\Pavan Lande\anaconda3\lib\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).
```



[53]:
```
sns.boxplot(df_train['age_median'])
plt.title('Population Density')
plt.show()
```

```
C:\Users\Pavan Lande\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
```



1.8.3 2. 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.

[54]:
```
df_train['pop'].describe()
```

[54]:
```
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
```

[55]:
```
df_train['pop_bins']=pd.cut(df_train['pop'],bins=5,labels=['very␣
↪low','low','medium','high','very high'])
```

[56]:
```
df_train[['pop','pop_bins']]
```

[56]:
```
          pop  pop_bins
UID
267822   5230  very low
246444   2633  very low
245683   6881  very low
279653   2700  very low
247218   5637  very low
…         …       …
279212   1847  very low
277856   4155  very low
233000   2829  very low
287425  11542       low
265371   3726  very low

[27321 rows x 2 columns]
```

[57]:
```python
df_train['pop_bins'].value_counts()
```

[57]:
```
very low     27058
low            246
medium           9
high             7
very high        1
Name: pop_bins, dtype: int64
```

1.8.4 a) Analyze the married, separated, and divorced population for these population brackets

[58]:
```python
df_train.groupby(by='pop_bins')[['married','separated','divorced']].count()
```

[58]:
```
           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
```

[59]:
```python
df_train.groupby(by='pop_bins')[['married','separated','divorced']].
↪agg(["mean", "median"])
```

[59]:
```
            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
```

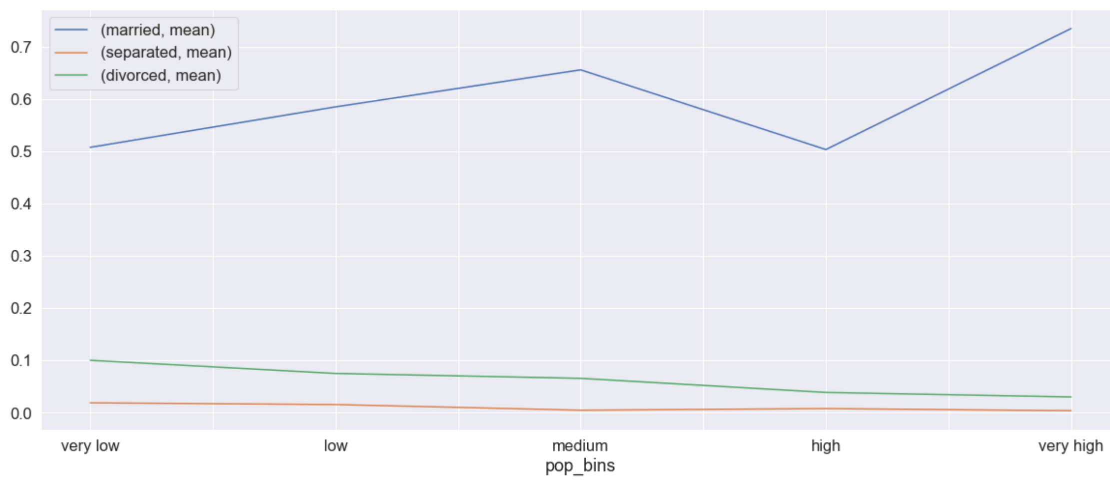
medium	0.655737	0.618710	0.005003	0.004120	0.065927	0.064890
high	0.503359	0.335660	0.008141	0.002500	0.039030	0.010320
very high	0.734740	0.734740	0.004050	0.004050	0.030360	0.030360

[60]:
```
## 1.Very high population group has more married people and less percantage of␣
↪separated and divorced couples
## 2.In very low population groups, there are more divorced people
```

1.8.5 b) Visualize using appropriate chart type

[61]:
```
plt.figure(figsize=(10,5))
pop_bin_married=df_train.
↪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>



1.8.6 3. Please detail your observations for rent as a percentage of income at an overall level, and for different states.

[62]:
```
rent_state_mean=df_train.groupby(by='state')['rent_mean'].agg(["mean"])
rent_state_mean.head()
```

[62]:

```
                          mean
state
```

```
Alabama       774.004927
Alaska       1185.763570
Arizona      1097.753511
Arkansas      720.918575
California   1471.133857
```

[63]:
```
income_state_mean=df_train.groupby(by='state')['family_mean'].agg(["mean"])
income_state_mean.head()
```

[63]:
```
                    mean
state
Alabama     67030.064213
Alaska      92136.545109
Arizona     73328.238798
Arkansas    64765.377850
California  87655.470820
```

[64]:
```
rent_perc_of_income=rent_state_mean['mean']/income_state_mean['mean']
rent_perc_of_income.head(10)
```

[64]:
```
state
Alabama                 0.011547
Alaska                  0.012870
Arizona                 0.014970
Arkansas                0.011131
California              0.016783
Colorado                0.013529
Connecticut             0.012637
Delaware                0.012929
District of Columbia    0.013198
Florida                 0.015772
Name: mean, dtype: float64
```

[65]:
```
#overall level rent as a percentage of income
sum(df_train['rent_mean'])/sum(df_train['family_mean'])
```

[65]:
```
0.013358170721473864
```

1.8.7 4. Perform correlation analysis for all the relevant variables by creating a heatmap. Describe your findings.

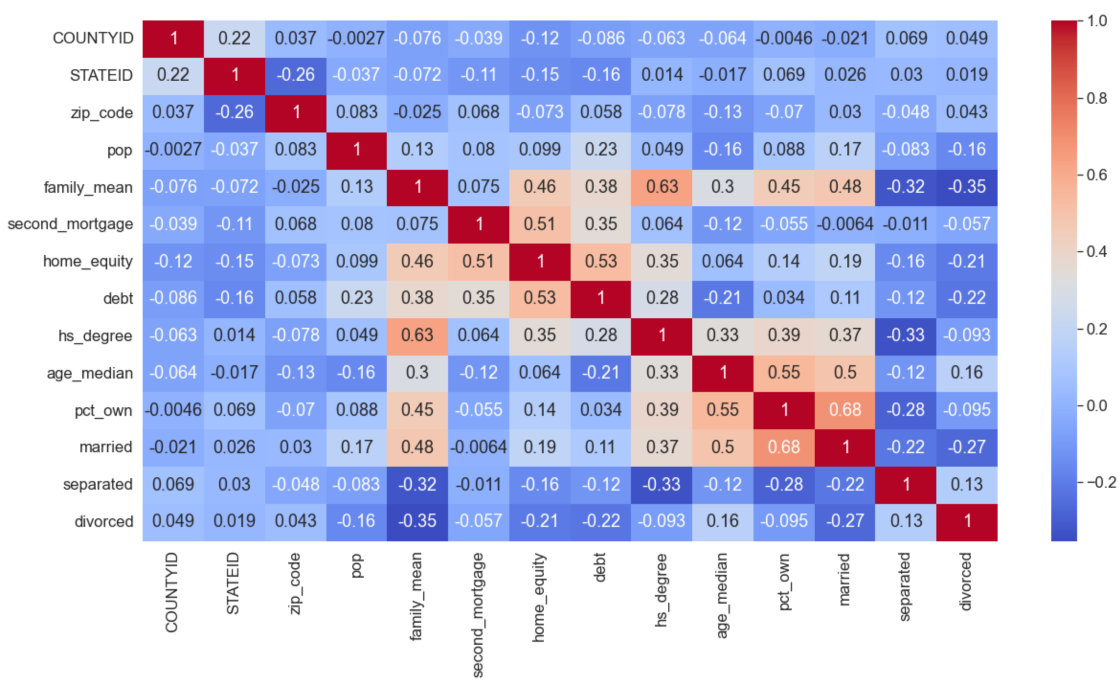
[66]:
```
df_train.columns
```

[66]:
```
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', 'bins', 'pop_density', 'age_median', 'pop_bins'],
dtype='object')
```

[67]:
```
cor=df_train[['COUNTYID','STATEID','zip_code','type','pop', 'family_mean',
         'second_mortgage', 'home_equity', 'debt','hs_degree',
           'age_median','pct_own', 'married','separated', 'divorced']].corr()
```

[68]:
```
plt.figure(figsize=(20,10))
sns.heatmap(cor,annot=True,cmap='coolwarm')
plt.show()
```



[69]:
```
## 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
```

1.9 Project Task: Week 3

1.9.1 Data Pre-processing:

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**

[73]:
```
pip install factor_analyzer
```

```
Collecting factor_analyzerNote: you may need to restart the kernel to use
updated packages.
  Downloading factor_analyzer-0.4.1.tar.gz (41 kB)
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
    Preparing wheel metadata: started
    Preparing wheel metadata: finished with status 'done'
Requirement already satisfied: scipy in c:\users\pavan lande\anaconda3\lib\site-
packages (from factor_analyzer) (1.7.3)
Requirement already satisfied: pandas in c:\users\pavan
lande\anaconda3\lib\site-packages (from factor_analyzer) (1.4.2)
Collecting pre-commit
```

```
  Downloading pre_commit-3.1.0-py2.py3-none-any.whl (202 kB)
Requirement already satisfied: scikit-learn in c:\users\pavan
lande\anaconda3\lib\site-packages (from factor_analyzer) (1.0.2)
Requirement already satisfied: numpy in c:\users\pavan lande\anaconda3\lib\site-
packages (from factor_analyzer) (1.21.5)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\pavan
lande\anaconda3\lib\site-packages (from pandas->factor_analyzer) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\pavan
lande\anaconda3\lib\site-packages (from pandas->factor_analyzer) (2021.3)
Requirement already satisfied: six>=1.5 in c:\users\pavan
lande\anaconda3\lib\site-packages (from python-
dateutil>=2.8.1->pandas->factor_analyzer) (1.16.0)
Requirement already satisfied: pyyaml>=5.1 in c:\users\pavan
lande\anaconda3\lib\site-packages (from pre-commit->factor_analyzer) (6.0)
Collecting identify>=1.0.0
  Downloading identify-2.5.18-py2.py3-none-any.whl (98 kB)
Collecting cfgv>=2.0.0
  Downloading cfgv-3.3.1-py2.py3-none-any.whl (7.3 kB)
Collecting nodeenv>=0.11.1
  Downloading nodeenv-1.7.0-py2.py3-none-any.whl (21 kB)
Collecting virtualenv>=20.10.0
  Downloading virtualenv-20.19.0-py3-none-any.whl (8.7 MB)
Requirement already satisfied: setuptools in c:\users\pavan
lande\anaconda3\lib\site-packages (from nodeenv>=0.11.1->pre-
commit->factor_analyzer) (61.2.0)

Collecting platformdirs<4,>=2.4
  Downloading platformdirs-3.0.0-py3-none-any.whl (14 kB)
Requirement already satisfied: filelock<4,>=3.4.1 in c:\users\pavan
lande\anaconda3\lib\site-packages (from virtualenv>=20.10.0->pre-
commit->factor_analyzer) (3.6.0)
Collecting distlib<1,>=0.3.6
  Downloading distlib-0.3.6-py2.py3-none-any.whl (468 kB)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\pavan
lande\anaconda3\lib\site-packages (from scikit-learn->factor_analyzer) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\pavan
lande\anaconda3\lib\site-packages (from scikit-learn->factor_analyzer) (1.1.0)
Building wheels for collected packages: factor-analyzer
  Building wheel for factor-analyzer (PEP 517): started
  Building wheel for factor-analyzer (PEP 517): finished with status 'done'
  Created wheel for factor-analyzer:
filename=factor_analyzer-0.4.1-py2.py3-none-any.whl size=42070
sha256=a8df995b025535c765e1b698c3485164a9b42199796963679f564a48d7644cd4
  Stored in directory: c:\users\pavan lande\appdata\local\pip\cache\wheels\6d\32
\bd\460a71becd83f7d77152f437c2fd451f5c87bc19cfcdbfcd24
Successfully built factor-analyzer
Installing collected packages: platformdirs, distlib, virtualenv, nodeenv,
identify, cfgv, pre-commit, factor-analyzer
```

```
Successfully installed cfgv-3.3.1 distlib-0.3.6 factor-analyzer-0.4.1
identify-2.5.18 nodeenv-1.7.0 platformdirs-3.0.0 pre-commit-3.1.0
virtualenv-20.19.0
```

[ ]:
```python
#pip install factor_analyzer
```

[85]:
```python
from sklearn.decomposition import FactorAnalysis
from factor_analyzer import FactorAnalyzer
```

[86]:
```python
fa=FactorAnalyzer(n_factors=5)
fa.fit_transform(df_train.select_dtypes(exclude= ('object','category')))
fa.loadings_
```

[86]:
```
array([[-1.13003333e-01,  2.05719967e-02, -2.56180655e-02,
        -6.41023715e-02,  4.32699935e-02],
       [-1.12981842e-01,  1.44962185e-02,  2.63002396e-02,
        -1.55460419e-01,  1.08699973e-01],
       [-1.30329217e-01, -3.58750675e-02,  4.50369897e-02,
        -9.58856411e-02, -5.21282677e-02],
       [-7.63666526e-02,  5.37354293e-02, -1.40479745e-01,
        -4.60011169e-02, -9.63398481e-02],
       [ 1.93211131e-02,  1.91263727e-02,  6.38895169e-03,
         2.79938985e-02, -4.98570662e-03],
       [ 9.11230628e-02, -9.62686827e-02, -7.00524147e-02,
        -1.36620317e-01, -1.51166208e-01],
       [-1.66259963e-02, -4.20576435e-02,  1.48210461e-01,
         4.88657934e-03,  1.01592880e-01],
       [-3.87971668e-02, -1.95136636e-02,  3.58355559e-02,
        -9.42111456e-02,  6.27513648e-02],
       [-6.77269397e-04, -1.49085756e-02, -3.02266534e-03,
        -4.53699545e-02,  2.53314974e-02],
       [ 6.99142620e-02,  9.56995396e-01, -8.49426462e-02,
        -1.26461993e-02, -4.48224753e-02],
       [ 6.55378988e-02,  9.18434053e-01, -1.07244320e-01,
        -3.40257996e-02, -4.35710778e-02],
       [ 7.09673445e-02,  9.47262361e-01, -5.81606384e-02,
         9.67512791e-03, -4.43896833e-02],
       [ 7.64127321e-01,  1.98361193e-03, -2.78394384e-02,
         1.07702953e-01, -1.35277893e-01],
       [ 7.13326490e-01, -9.89244802e-04, -3.77273019e-02,
         1.02603144e-01, -1.45864174e-01],
       [ 6.99131216e-01,  1.57217535e-02,  2.31589748e-03,
         9.52903540e-02,  6.56215552e-02],
       [-1.49781946e-01,  3.31323122e-01, -4.84989803e-01,
        -5.21607502e-02,  3.14493523e-01],
       [ 2.09878366e-01,  4.27069412e-01, -6.32065558e-01,
        -4.25040060e-02,  3.38367875e-01],
```

```
[-5.46488833e-02,  3.20305048e-02,  3.45859373e-02,
  4.51419047e-01, -1.70088305e-01],
[-4.02662086e-02,  1.10196806e-02,  5.38546908e-02,
  6.84871386e-01, -1.65772404e-01],
[-5.84254091e-02, -2.48620470e-02,  9.33300783e-02,
  8.46574105e-01, -1.04568452e-01],
[-7.32007166e-02, -4.50215584e-02,  1.25021273e-01,
  9.35912915e-01, -5.85047596e-02],
[-8.32222478e-02, -5.37499169e-02,  1.50851475e-01,
  9.64084468e-01, -3.64496819e-02],
[-6.81088890e-02, -6.22289357e-02,  1.56327619e-01,
  9.42924478e-01, -1.43864603e-02],
[-6.34245942e-02, -6.84197736e-02,  1.43598640e-01,
  8.96839189e-01, -3.18630090e-03],
[-4.42655319e-02, -8.17048662e-02,  1.07534461e-01,
  7.86808140e-01,  1.71262370e-02],
[ 1.91057689e-01,  4.55546817e-01, -6.06029977e-01,
 -4.18894657e-02,  3.57971951e-01],
[ 2.12022311e-01,  4.36535847e-01, -6.20126901e-01,
 -4.43586254e-02,  3.34546570e-01],
[ 7.88443048e-01,  4.65200095e-02,  1.48750290e-01,
 -2.14813165e-01, -1.62779033e-01],
[ 7.15327469e-01,  4.87452360e-02,  1.34103066e-01,
 -2.27209435e-01, -2.17134149e-01],
[ 8.59607738e-01,  3.74310514e-02,  1.76206972e-01,
 -1.31897681e-01,  2.01536368e-02],
[-2.35913965e-01,  8.45350862e-01, -4.14128757e-02,
  6.47882950e-02,  2.24477106e-01],
[ 1.33322411e-01,  9.51861953e-01,  2.82661013e-02,
 -5.43308940e-02,  9.35905800e-02],
[ 8.31516871e-01,  3.03677746e-02,  1.67472111e-01,
 -2.16060648e-01, -8.48864992e-02],
[ 7.96268282e-01,  2.50232881e-02,  1.56696750e-01,
 -2.18560300e-01, -1.00552494e-01],
[ 8.08207856e-01,  3.70115751e-02,  1.53989852e-01,
 -1.19804861e-01,  4.64019993e-02],
[-3.44199703e-01,  8.67678655e-01,  3.63064032e-02,
  9.25887129e-02,  4.54704507e-02],
[ 4.63702796e-02,  9.38042314e-01,  1.52749826e-01,
 -2.75865855e-02, -9.71709825e-02],
[ 9.68417670e-01, -4.48586972e-02, -9.10478399e-02,
  3.08715515e-02,  5.90236100e-02],
[ 9.49377997e-01, -5.05832535e-02, -1.06884565e-01,
  3.13147810e-02,  5.20767455e-02],
[ 8.08006341e-01, -6.51570187e-03,  9.02769332e-02,
  9.44598692e-03,  1.16064912e-01],
[-4.07669423e-01,  7.29254444e-01,  3.31091280e-01,
```

```
 -6.32032305e-02, -2.72821932e-01],
[ 7.97104874e-02,  7.30801450e-01,  2.70172196e-01,
 -4.62425318e-02, -3.55731491e-01],
[ 8.97463748e-01, -6.50750036e-02, -3.23973988e-02,
 -1.75339564e-02,  1.46482775e-01],
[ 8.60411442e-01, -6.40573773e-02, -4.52926250e-02,
 -1.78353051e-02,  1.35642250e-01],
[ 7.45517185e-01, -1.32709107e-02,  6.89506020e-02,
 -9.20061498e-03,  2.45218858e-01],
[-1.21253812e-01,  6.12058284e-01,  6.39942517e-01,
 -1.93121041e-02,  2.48210594e-01],
[-3.37575977e-01,  5.66387698e-01,  5.91355859e-01,
 -1.91367711e-02,  2.23333161e-01],
[-1.53307850e-01, -9.59396077e-03, -1.70909625e-01,
  1.20759470e-01, -6.58032010e-01],
[-1.30121743e-01, -1.63477700e-02, -1.71935572e-01,
  1.36666252e-01, -6.68510039e-01],
[ 2.48587550e-01, -2.31685998e-02, -3.48333037e-02,
  1.00381648e-01, -6.47889452e-01],
[ 2.05660542e-01,  7.99296854e-02, -3.12138301e-01,
  2.47865377e-02, -6.32441741e-01],
[ 9.95132414e-02, -7.05184740e-02, -2.07089949e-02,
 -1.07084332e-01,  6.78939658e-01],
[-2.67666446e-01, -9.10810647e-03, -2.81401396e-02,
 -9.90229427e-02,  6.53752096e-01],
[-2.17063350e-01, -7.47028387e-02,  3.60837403e-01,
 -2.10168609e-02,  6.39882157e-01],
[ 4.00279654e-01,  6.32486970e-02,  2.53998488e-01,
 -2.24675879e-01, -1.89316386e-01],
[ 4.12920658e-01,  6.43954282e-02,  2.23073033e-01,
 -2.14866145e-01, -1.77368536e-01],
[ 3.59723626e-01,  5.64244391e-02,  2.67920851e-01,
 -2.20427907e-01, -1.84360933e-01],
[ 2.37616985e-01, -4.90754649e-02,  8.28589543e-01,
  9.66538119e-02,  3.23270503e-01],
[ 2.46056868e-01, -3.26324605e-02,  8.43916689e-01,
  7.89928830e-02,  2.42802124e-01],
[-6.18904813e-02,  6.94603513e-02,  5.91071448e-01,
  9.39979816e-02,  9.15335951e-02],
[ 4.68518327e-02,  8.16906830e-01, -1.77327732e-01,
 -2.18455964e-02, -3.75667099e-02],
[ 6.33392347e-02,  9.23206345e-01, -1.05651914e-01,
 -3.38139050e-02, -4.77609268e-02],
[ 1.94836151e-01, -4.79014648e-02,  8.17679376e-01,
  1.47014893e-01,  3.29278649e-01],
[ 1.92286156e-01, -3.18766478e-02,  8.70978867e-01,
  1.36317638e-01,  2.51745577e-01],
```

```
[-9.85866541e-02,  6.33382613e-02,  4.77141914e-01,
  7.91177420e-02,  1.12800122e-01],
[ 5.08553413e-02,  8.76717092e-01, -1.48521246e-01,
  1.59920905e-02, -4.72935730e-02],
[ 6.87713537e-02,  9.54016647e-01, -5.65521229e-02,
  1.09296343e-02, -4.88907668e-02],
[-1.58063861e-02,  1.21323443e-01,  7.81017851e-01,
 -3.23270437e-02, -2.77799041e-01],
[ 1.88424297e-01,  1.96370886e-01,  5.61647961e-01,
 -1.16812481e-01, -1.31585929e-01],
[-7.14362045e-02, -7.51299192e-02, -2.64965586e-01,
  1.26583417e-01,  1.88388803e-01],
[-1.61918729e-01, -7.34385292e-02, -1.43626238e-01,
  1.24360039e-01,  1.46458330e-01],
[-3.53031847e-01, -4.96231769e-02,  1.46372388e-01,
  3.49935673e-02,  1.18785041e-01],
[ 2.45684674e-01, -2.63242706e-02, -4.09141295e-02,
  1.10429624e-01, -6.60731820e-01],
[ 3.36135215e-01, -2.07516448e-02, -3.85660965e-01,
  4.74768600e-02,  2.84198478e-01],
[ 2.31050215e-01, -3.31033501e-02,  9.06377823e-01,
  1.17057295e-01,  2.63072753e-01]])
```

1.10 Project Task: Week 4

1.10.1 Data Modeling :

1.10.2 1. 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.

[74]:
```python
df_train.columns
```

[74]:
```
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', 'bins', 'pop_density', 'age_median', 'pop_bins'],
      dtype='object')
```

[75]:
```
df_train['type'].unique()
type_dict={'type':{'City':1,
                   'Urban':2,
                   'Town':3,
                   'CDP':4,
                   'Village':5,
                   'Borough':6}
          }
df_train.replace(type_dict,inplace=True)
```

[76]:
```
df_train['type'].unique()
```

[76]:
```
array([1, 2, 3, 4, 5, 6], dtype=int64)
```

[77]:
```
df_test.replace(type_dict,inplace=True)
```

[78]:
```
df_test['type'].unique()
```

[78]:
```
array([4, 1, 6, 3, 5, 2], dtype=int64)
```

[79]:
```
feature_cols=['COUNTYID','STATEID','zip_code','type','pop', 'family_mean',
         'second_mortgage', 'home_equity', 'debt','hs_degree',
           'age_median','pct_own', 'married','separated', 'divorced']
```

[80]:
```
x_train=df_train[feature_cols]
y_train=df_train['hc_mortgage_mean']
```

[81]:
```
x_test=df_test[feature_cols]
y_test=df_test['hc_mortgage_mean']
```

[82]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,␣
↪mean_absolute_error,mean_squared_error,accuracy_score
```

[83]:
```
x_train.head()
```

[83]:
```
        COUNTYID  STATEID  zip_code  type   pop  family_mean  second_mortgage  \
UID
267822        53       36     13346     1  5230  67994.14790          0.02077
246444       141       18     46616     1  2633  50670.10337          0.02222
245683        63       18     46122     1  6881  95262.51431          0.00000
279653       127       72       927     2  2700  56401.68133          0.01086
247218       161       20     66502     1  5637  54053.42396          0.05426

        home_equity     debt  hs_degree  age_median  pct_own  married  \
UID
267822      0.08919  0.52963    0.89288   44.666665  0.79046  0.57851
246444      0.04274  0.60855    0.90487   34.791665  0.52483  0.34886
245683      0.09512  0.73484    0.94288   41.833330  0.85331  0.64745
279653      0.01086  0.52714    0.91500   49.750000  0.65037  0.47257
247218      0.05426  0.51938    1.00000   22.000000  0.13046  0.12356

        separated  divorced
UID
267822    0.01240   0.08770
246444    0.01426   0.09030
245683    0.01607   0.10657
279653    0.02021   0.10106
247218    0.00000   0.03109
```

[84]:
```
sc=StandardScaler()
x_train_scaled=sc.fit_transform(x_train)
x_test_scaled=sc.fit_transform(x_test)
```

1.10.3 a) Run a model at a Nation level. If the accuracy levels and R square are not satisfactory proceed to below step.

[87]:
```
linereg=LinearRegression()
linereg.fit(x_train_scaled,y_train)
```

[87]:
```
LinearRegression()
```

[88]:
```
y_pred=linereg.predict(x_test_scaled)
```

[89]:
```
print("Overall R2 score of linear regression model", r2_score(y_test,y_pred))
print("Overall RMSE of linear regression model", np.
 ↪sqrt(mean_squared_error(y_test,y_pred)))
```

```
Overall R2 score of linear regression model 0.7348210754610929
Overall RMSE of linear regression model 323.10188949846344
```

[ ]:
```
### The Accuracy and R2 score are good, but still will investigate the model␣
↪performance at state level
```

1.10.4 b) Run another model at State level. There are 52 states in USA.

[90]:
```
state=df_train['STATEID'].unique()
state[0:5]
#Picking a few iDs 20,1,45,6
```

[90]:
```
array([36, 18, 72, 20,  1], dtype=int64)
```

[91]:
```
for i in [20,1,45]:
    print("State ID-",i)

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

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

    x_train_scaled_nation=sc.fit_transform(x_train_nation)
    x_test_scaled_nation=sc.fit_transform(x_test_nation)

    linereg.fit(x_train_scaled_nation,y_train_nation)
    y_pred_nation=linereg.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 :- 0.6046603766461807
Overall RMSE of linear regression model for state, 20 :- 307.97188999314733


State ID- 1
Overall R2 score of linear regression model for state, 1 :- 0.8104382475484616
Overall RMSE of linear regression model for state, 1 :- 307.8275861848435


State ID- 45
Overall R2 score of linear regression model for state, 45 :- 0.7887446497855253
Overall RMSE of linear regression model for state, 45 :- 225.69615420724128
```

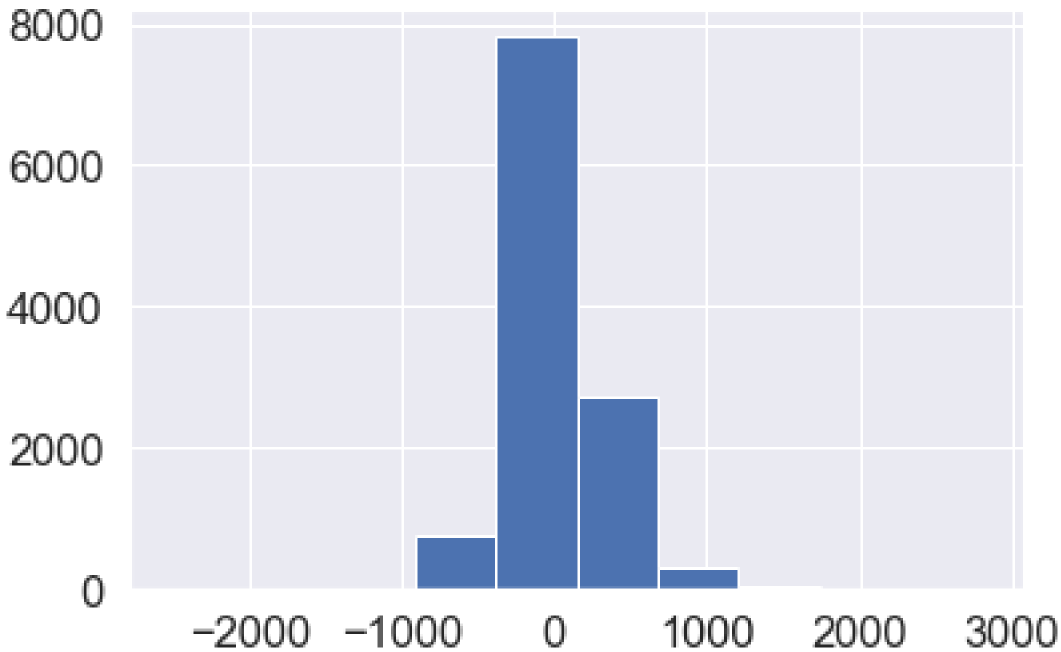
[92]:
```
# To check the residuals
```

[93]:
```
residuals=y_test-y_pred
residuals
```

[93]:
```
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
```

[94]:
```
plt.hist(residuals) # Normal distribution of residuals
```

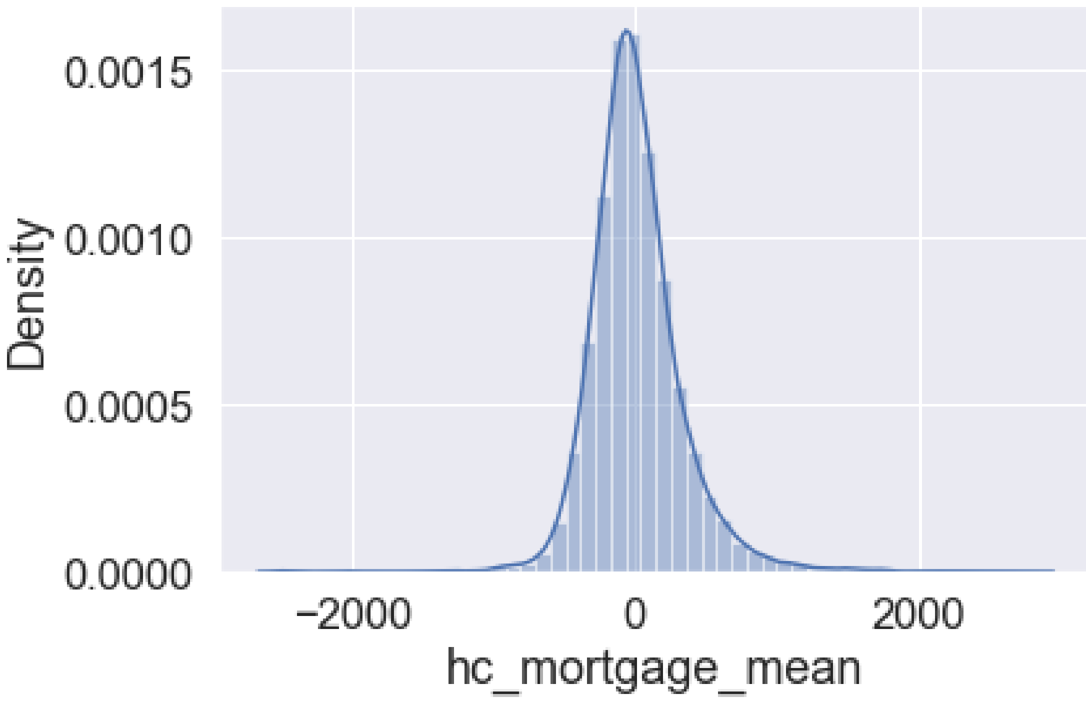
[94]:
```
(array([6.000e+00, 3.000e+00, 2.900e+01, 7.670e+02, 7.823e+03, 2.716e+03,
        3.010e+02, 4.900e+01, 1.200e+01, 3.000e+00]),
 array([-2515.04284233, -1982.92661329, -1450.81038425,  -918.69415521,
         -386.57792617,   145.53830287,   677.65453191,  1209.77076095,
         1741.88698999,  2274.00321903,  2806.11944807]),
 <BarContainer object of 10 artists>)
```



[95]:
```
sns.distplot(residuals)
```

```
C:\Users\Pavan Lande\anaconda3\lib\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).
```

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



[96]:
```
plt.scatter(residuals,y_pred) # Same variance and residuals does not have
 ↪correlation with predictor
# Independance of residuals
```

[96]: <matplotlib.collections.PathCollection at 0x222048c0bb0>


4000
3000
2000
1000
0
−2000
−1000
0
1000
2000
3000

[ ]: