DS Capstone Project 1 | Real Estate

A banking institution requires actionable insights from the perspective of Mortgage-Backed Securities, Geographic Business Investment and Real Estate Analysis.

The objective is to identify white spaces/potential business in the mortgage loan.

In [1]:
```
# Import required libraries and train, test datasets.
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
pd.options.display.max_columns = None
pd.options.display.max_rows = None
# To display all rows and columns
```

In [4]:
```
# import train data
df_train=pd.read_csv("D:\Bipasha Hambir\Capstone project\Project1\train.csv")
df_train.head(3)
```

Out[4]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	tract	46122	317
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	tract	927	787
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	tract	66502	785

In [5]:
```
# Data cleaning and understanding
```

In [6]:
```
df_train["type"].value_counts()
```

Out[6]:
```
City       15237
Town        3666
CDP         3658
Village     3216
Borough     1226
Urban        318
Name: type, dtype: int64
```

In [7]:
```
# Check dimension
df_train.shape
```

Out[7]: (27321, 80)

In [8]:
```
# import test data
df_test=pd.read_csv("D:\Simplilearn\Capstone project\Project1\test.csv")
df_test.head()
```

Out[8]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code
0	255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	tract	48239	
1	252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	tract	4210	
2	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	tract	14871	
3	248614	NaN	140	231	21	Kentucky	KY	Monticello	Monticello City	City	tract	42633	
4	286865	NaN	140	355	48	Texas	TX	Corpus Christi	Edroy	Town	tract	78410	

In [9]:
```
# Check dimension of data
df_test.shape
```

Out[9]: (11709, 80)

In [10]:
```
# See datatypes
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
```

In [11]:
```
# Check "UID" is unique or not
print(df_train.UID.nunique())
print(df_test.UID.nunique())
# Unique value is not equal to the train and test row length so it has duplicate value
```

```
27161
11677
```

In [12]:
```
# Check duplicate values
df_train.duplicated(["UID"]).sum()
```

Out[12]: 160

In [13]:
```
# Check missing values
df_train.isnull().sum().head(5)
```

Out[13]:
```
UID             0
BLOCKID     27321
SUMLEVEL        0
COUNTYID        0
STATEID         0
dtype: int64
```

In [14]:
```
# Check percentage of missing values in each column
(df_train.isnull().sum()/df_train.shape[0])*100
```

Out[14]:
```
[illegible]
dtype: float64
```

In [15]:
```
#Check unique values of column 'SUMLEVEL'
print(df_train['SUMLEVEL'].unique())
print(df_test['SUMLEVEL'].unique())
```

```
[140]
[140]
```

In [16]:
```
# "BLOCKID" has all null values and "SUMLEVEL" has all same values so we drop this as it isn't effect data.
df_train.drop(['BLOCKID','SUMLEVEL'],axis=1,inplace=True)
```

In [17]:
```
df_train.head(2)
```

Out[17]:

	UID	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	lat	lng
0	267822	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.840812	-75.501524
1	246444	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.701441	-86.266614

In [18]:
```
# we have enough rows so we drop null values and also it did not cause much data loss
df_tr=df_train.dropna()
df_tr.shape
```

Out[18]: (26585, 78)

In [19]:
```
(df_tr.shape[0]/df_train.shape[0])*100  # data loss 3%
```

Out[19]: 97.30610153361883

In [20]:
```
# Check percentage of missing values in each column
(df_test.isnull().sum()/df_test.shape[0])*100
```

Out[20]:
```
[illegible]
dtype: float64
```

In [21]:
```
# drop unnecessary column
df_test.drop(["BLOCKID","SUMLEVEL"],axis=1,inplace=True)
```

In [22]:
```
# we have enough rows so we drop null values and also it did not cause much data loss
df_te=df_test.dropna()
df_te.shape
```

Out[22]: (11355, 78)

In [23]:
```
(df_te.shape[0]/df_test.shape[0])*100     # data loss 3%
```

Out[23]: 96.97464061538552

In [24]:
```
df_tr.duplicated("UID").sum()
```

Out[24]: 0

In [25]:
```
df_te.duplicated("UID").sum()
```

Out[25]: 0

In [26]:
```
# there are no duplicate "UID", so we can make it index
```

In [27]:
```
df_tr.set_index(keys=("UID"),inplace=True)
```

In [28]:
```
df_te.set_index(keys=("UID"),inplace=True)
```

In [29]:
```
####################################################################
```

In [30]:
```
# we add an extra column of US country which will help create maps in tableau with this cleaned data.
df_tr["Country"]="United States"
df_tr.head(2)
```

Out[30]:

	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	lat	lng	ALand
UID													
267822	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	42.840812	-75.501524	202183361
246444	141	18	Indiana	IN	South Bend	Roseland	City	tract	46616	574	41.701441	-86.266614	1560828

In [31]:
```
df_tr.columns
```

Out[31]:
```
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',
       'Country'],
      dtype='object')
```

In [32]:
```
# Save these cleaned data in csv format
df_tr.to_csv("D:/Simplilearn/Capstone project/Project1/train_clean.csv")
df_te.to_csv("D:/Simplilearn/Capstone project/Project1/test_clean.csv")
```

In [33]:
```
df_tr.describe().T   # summary
```

Out[33]:

	count	mean	std	min	25%	50%	75%
[illegible]							

In [34]:
```
df_te.describe().T   # summary
```

Out[34]:

	count	mean	std	min	25%	50%	75%
[illegible]							

EDA

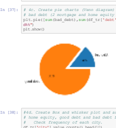
In [35]:
```
##a. top 2,500 locations where the percentage of households with a second mortgage is the highest
# and percent ownership is above 10%
a=df_tr[df_tr["pct_own"]>0.1]
a.sort_values(by=["second_mortgage"],ascending=False).head(2500)
```

Out[35]:

	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	lat	lng
289712	147	51	Virginia	VA	Farmville	Farmville	Town	tract	23901	434	37.297357	-78.396
251185	27	25	Massachusetts	MA	Worcester	Worcester City	City	tract	1610	508	42.254262	-71.800
269323	81	36	New York	NY	Corona	Harbor Hills	City	tract	11368	718	40.751809	-73.853
251324	3	24	Maryland	MD	Glen Burnie	Glen Burnie	CDP	tract	21061	410	39.127273	-76.635
235788	57	12	Florida	FL	Tampa	Egypt Lake-leto	CDP	tract	33614	813	28.029063	-82.495
242304	31	17	Illinois	IL	Chicago	Lincolnwood	Village	tract	60625	773	41.967289	-87.652
242456	31	17	Illinois	IL	Chicago	Chicago City	Village	tract	60623	773	41.906640	-87.698
292872	79	55	Wisconsin	WI	Milwaukee	Milwaukee City	City	tract	53210	414	43.067063	-87.952
226286	71	6	California	CA	Etiwanda	Rancho Cucamonga City	City	tract	91739	909	34.093184	-117.531
230597	81	6	California	CA	South San Francisco	San Bruno City	City	tract	94080	650	37.656229	-122.417
272149	61	39	Ohio	OH	Cincinnati	Cincinnati City	Village	tract	45219	513	39.121316	-84.511
282972	113	48	Texas	TX	Dallas	Dallas City	Town	tract	75207	214	32.806348	-96.773
232647	41	8	Colorado	CO	Colorado Springs	Colorado Springs City	City	tract	80916	719	38.816171	-104.702
232507	35	8	Colorado	CO	Littleton	Louviers	City	tract	80125	303	39.508438	-105.062
227668	55	6	California	CA	Napa	Napa City	City	tract	94559	707	38.297165	-122.297
271447	35	39	Ohio	OH	East Cleveland	East Cleveland City	City	tract	44112	216	41.527570	-81.557
255538	163	26	Michigan	MI	Highland Park	Highland Park City	CDP	tract	48203	313	42.397127	-83.105
231259	95	6	California	CA	Fairfield	Fairfield City	City	tract	94533	707	38.247535	-122.049
234204	1	11	District of Columbia	DC	Washington	Washington City	City	tract	20002	202	38.849942	-77.000
233686	3	9	Connecticut	CT	Hartford	Hartford City	City	tract	6105	860	41.767728	-72.706

In [36]:
```
# 4b. Calculate bad debt
bad_debt=df_tr["second_mortgage"]+df_tr["home_equity"]-df_tr["home_equity_second_mortgage"]
```

In [37]:
```
# 4c. Create pie charts (Venn diagram) to show overall debt (% bad debt and good debt) and
# bad debt (2 mortgage and home equity loan)
plt.pie([sum(bad_debt),sum(df_tr["debt"])],labels=["bad debt","good debt"],explode=[0.2,0],autopct="%3d%%")
plt.show()
```

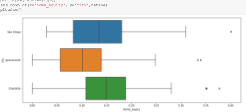


In [38]:
```
##d. Create Box and whisker plot and analyse the distribution for 2nd mortgage,
# home equity, good debt and bad debt for different cities.
# Check frequency of each city.
df_tr["city"].value_counts().head(10)
```

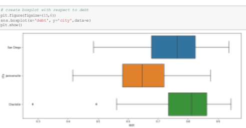
Out[38]:
```
Chicago          296
Brooklyn         261
Los Angeles      219
Houston          213
Philadelphia     160
San Antonio      128
Baltimore        128
Las Vegas        123
Phoenix          114
Miami            105
Name: city, dtype: int64
```

In [39]:
```
# Randomly pick three cities and create boxplot with respect to second_mortgage.
b=df_tr[df_tr["city"]=="San Diego"]
c=df_tr[df_tr["city"]=="Jacksonville"]
d=df_tr[df_tr["city"]=="Charlotte"]
e=pd.concat([b,c,d],axis=0)
plt.figure(figsize=(15,6))
sns.boxplot(x='second_mortgage', y='city',data=e)
plt.show()
```

In [40]:
```
# create boxplot with respect to home_equity.
plt.figure(figsize=(15,6))
sns.boxplot(x='home_equity', y='city',data=e)
plt.show()
```



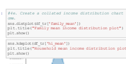
In [41]:
```
# create boxplot with respect to debt
plt.figure(figsize=(15,6))
sns.boxplot(x='debt', y='city',data=e)
plt.show()
```



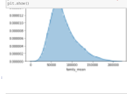
In [42]:
```
# create boxplot with respect to bad_debt
plt.figure(figsize=(15,6))
sns.boxplot(x=bad_debt, y=e["city"])
plt.xlabel("Bad debt")
plt.show()
```

In [43]:
```
#4e. Create a collated income distribution chart for family income, house hold income and remaining income.
sns.distplot(df_tr["family_mean"])
plt.title("Family mean income distribution plot")
plt.show()
```

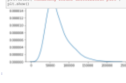
```
sns.kdeplot(df_tr['hi_mean'])
plt.title("Household mean income distribution plot")
plt.show()
```



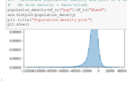
In [44]:
```
sns.distplot((df_tr['hi_mean']-df_tr['family_mean']))
plt.title("Remaining income distribution plot")
plt.show()
```



In [45]:
```
#5. Calculate population density and plot it's distribution
#   We know density = mass/volume
population_density=df_tr["pop"]/df_tr["ALand"]
sns.distplot(population_density)
plt.title("Population density plot")
plt.show()
```



In [46]:



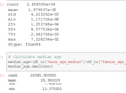
In [47]: `population_density.describe()  # summary of population density`

Out[47]:
```
count    2.658500e+04
mean     1.979937e-03
std      4.223262e-03
min      1.171076e-08
25%      1.253769e-04
50%      8.577534e-04
75%      2.042706e-03
max      7.228234e-02
dtype: float64
```

In [48]:
```
# Calculate median age
median_age=(df_tr['male_age_median']+df_tr['female_age_median'])/2
median_age.describe()
```

Out[48]:
```
count    26585.000000
mean        39.390229
std          7.463310
min         13.375000
25%         34.333333
50%         39.333330
75%         43.958330
max         80.166670
dtype: float64
```

In [49]:
```
sns.distplot(median_age,kde=False)
plt.title("Median age distribution plot")
plt.show()
```



In [50]:
```
#6. Create bins for population into a new variable by selecting appropriate class interval
#   so that the no of categories(bins) don't exceed 5 for the ease of analysis
df_tr["pop_bins"]=pd.cut(df_tr['pop'],bins=5,labels=['1','2','3','4','5'])
df_tr["pop_bins"].value_counts()
```

Out[50]:
```
1    26334
2      238
3        9
4        3
5        1
Name: pop_bins, dtype: int64
```

`df_tr[['pop_bins','pop']].head(2)`

Out[51]:

	pop_bins	pop
UID		
267822	1	5230
246444	1	2633

In [52]:
```
# Population bin wise analysis of married, separated, divorced column.
df_tr.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean", "median","count","std"])
```

Out[52]:

	married				separated				divorced			
	mean	median	count	std	mean	median	count	std	mean	median	count	std
pop_bins												
1	0.511868	0.528105	26334	0.130728	0.019017	0.013351	26334	0.020095	0.101046	0.094090	26334	0.047727
2	0.584690	0.592575	238	0.089282	0.015603	0.01108	238	0.015509	0.075749	0.070565	238	0.038176
3	0.655737	0.618710	9	0.066605	0.005002	0.00412	9	0.004739	0.065827	0.064890	9	0.029409
4	0.726957	0.736060	3	0.106582	0.015663	0.00916	3	0.014531	0.077310	0.063050	3	0.030074
5	0.734740	0.734740	1	NaN	0.004050	0.00405	1	NaN	0.030360	0.030360	1	NaN

In [53]:
```
#7. Statewise rent_mean statistics.
df_tr.groupby(by='state')[['rent_mean']].agg(["mean", "median","count","std","min","max"]).head(4)
```

Out[53]:

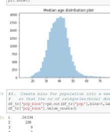
	mean	median	count	std	min	max
state						
Alabama	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Alaska	1172.690430	1161.377495	74	294.701055	533.60191	2347.69441
Arizona	1101.133796	1023.022040	523	426.761464	355.40366	3432.97803
Arkansas	715.387296	682.955320	264	179.199154	413.01673	1559.14885

In [54]:
```
# State wise family_mean statistics.
df_tr.groupby(by='state')['family_mean'].agg(["mean", "median","count","std","min","max"]).head(4)
```

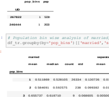
Out[54]:

	mean	median	count	std	min	max
state						
Alabama	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Alaska	93504.626703	90936.804995	74	21501.975204	51064.20258	149641.7083
Arizona	73546.551858	67420.986670	523	30141.703163	24223.38044	174136.6886
Arkansas	64046.416919	59955.383620	264	20587.910885	25969.78205	168048.8535

In [55]:
```
# percentage of rent mean of 5 states.
f=df_tr.groupby(by='state')['rent_mean'].mean()
plt.pie(f.head(5),autopct="%3d%%",labels=f.head(5).index)
plt.show()
```



In [56]:
```
# percentage of family mean of 5 states.
g=df_tr.groupby(by='state')['family_mean'].mean()
plt.pie(g.head(5),autopct="%3d%%",labels=g.head(5).index)
plt.show()
```



In [57]:
```
# Statewise overall rent as a percentage of income
rent_per_income=(f/g)*100
rent_per_income.head(5)
```

Out[57]:
```
state
Alabama       1.150641
Alaska        1.268940
Arizona       1.497193
Arkansas      1.116951
California    1.672761
dtype: float64
```

In [58]:
```
#8. Correlation analysis for all the relevant variables by creating a heatmap.
# relevant variables correlation
h=df_tr[['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']].corr()
h
```

Out[58]:

	ALand	AWater	pop	male_pop	female_pop	rent_mean	rent_median	rent_stdev	rent_sample...
ALand	1.000000	0.412784	-0.037642	-0.027673	-0.046990	-0.066063	-0.065349	-0.033645	[illegible]
AWater	0.412784	1.000000	0.016555	-0.013529	-0.019045	-0.002419	-0.006744	[illegible]	[illegible]
pop	-0.037642	0.016555	1.000000	0.965101	0.965804	[illegible]	[illegible]	[illegible]	[illegible]
male_pop	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
female_pop	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_mean	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_median	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_stdev	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_sample_weight	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_samples	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_gt_10	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_gt_15	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_gt_20	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_gt_25	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_gt_30	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_gt_35	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_gt_40	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
rent_gt_50	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
universe_samples	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
used_samples	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hi_mean	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hi_median	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hi_stdev	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hi_sample_weight	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hi_samples	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
family_mean	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
family_median	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
family_stdev	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
family_sample_weight	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
family_samples	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hc_mortgage_mean	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hc_mortgage_median	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hc_mortgage_stdev	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hc_mortgage_sample_weight	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hc_mortgage_samples	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hc_mean	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hc_median	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hc_stdev	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hc_samples	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hc_sample_weight	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
home_equity_second_mortgage	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
second_mortgage	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
home_equity	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
debt	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
second_mortgage_cdf	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
home_equity_cdf	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
debt_cdf	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hs_degree	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hs_degree_male	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
hs_degree_female	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
male_age_mean	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
male_age_median	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
male_age_stdev	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
male_age_sample_weight	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
male_age_samples	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
female_age_mean	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
female_age_median	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
female_age_stdev	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
female_age_sample_weight	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
female_age_samples	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
pct_own	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
married	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
married_snp	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
separated	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
divorced	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

In [59]:
```
# Heatmap
plt.figure(figsize=(15,5))
sns.heatmap(h,cmap="bwr")
plt.show()
```

FE

In [60]:
```
#9. Use factor analysis to find latent variables in our dataset.
# import libraries
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
from factor_analyzer import FactorAnalyzer
from sklearn.decomposition import FactorAnalysis
```

In [61]:
```
# relevant variables
train=df_tr[['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']]
train.shape
```

Out[61]: (26585, 65)

In [62]:
```
# Bartlett_sphericity test
# Bartlett's Test of Sphericity compares an observed correlation matrix to the identity matrix.
# Essentially it checks to see if there is a certain redundancy between the variables that we can summa
rize with
# a few number of factors.
chi_square_value,p_value=calculate_bartlett_sphericity(train)
chi_square_value,p_value
# Since p_value is less than 0.05 we reject null hypothesis which means observed correlation is not ide
ntity matrix.
# The test is statistically significant.
```

Out[62]: (8824144.012160782, 0.0)

In [63]:
```
fa=FactorAnalysis(n_components=5,random_state=1234)
fa.fit_transform(train)
```

Out[63]:
```
array([[ 0.05398456, -0.04928693, -0.37149127, -0.17226607, -0.57187931],
       [-0.10055339, 0.01508954, -0.89907798, -0.53976406, 0.87508589],
       [-0.04819296, -0.00850804, 0.49270066, -0.65254609, 0.37164669],
       ...,
       [ 0.91793109, -0.37455319, 0.42330274, -0.44996357, 0.22590949],
       [-0.08739231, 0.00921959, 3.15709743, -0.70988366, 0.36864706],
       [-0.09575864, 0.01234399, -0.74223958, -0.19824877, -0.42240218]])
```

In [64]:
```
fc=FactorAnalyzer(n_factors=5)
fc.fit_transform(train)
fc.loadings_
```

Out[64]:
```
array([[-0.0493062 , -0.0245062 , 0.03434916, -0.08479976, 0.06788041],
       [-0.00437269, -0.016404 , -0.00298266, -0.04008107, 0.02572808],
       [ 0.05638687, 0.9626342 , -0.14725594, 0.01912881, -0.08119301],
       [ 0.05624263, 0.93418697, -0.14997253, 0.00128653, -0.09205297],
       [ 0.07446931, 0.9626342 , -0.13989343, 0.03567281, -0.06780412],
       [ 0.78408365, -0.01135313, 0.00905801, 0.1661509 , -0.14741935],
       [ 0.72869811, -0.00465726, 0.00742849, 0.15780394, -0.16325702],
       [ 0.73022422, 0.00769071, -0.08680548, 0.13908288, 0.07204705],
       [-0.11770463, 0.32192314, -0.71384252, -0.11087457, 0.24698938],
       [ 0.26103281, 0.40029424, -0.91253111, -0.08746026, 0.23810338],
       [-0.01582813, 0.05067225, 0.03641507, 0.41791736, -0.13899302],
       [ 0.02428898, 0.05474536, 0.04016496, 0.64995232, -0.11374045],
       [ 0.02241097, -0.01388299, 0.05594352, 0.8172025 , -0.03567906],
       [ 0.01486968, -0.00104739, 0.07494925, 0.91609246, 0.01905638],
       [ 0.00618699, -0.00853108, 0.1011713 , 0.95453872, 0.04746954],
       [ 0.0182844 , 0.0182741 , 0.10476431, 0.94232116, 0.06873523],
       [ 0.0191146 , -0.02678515, 0.0935655 , 0.89726336, 0.07393322],
       [ 0.02773071, -0.04478673, 0.05821932, 0.78715451, 0.07931205],
       [ 0.2416936 , 0.42353603, -0.90349028, -0.08745045, 0.26227676],
       [ 0.2638032 , 0.40380551, -0.90771002, -0.09065663, 0.23490612],
       ...
       [ 0.05624263, 0.93418697, -0.14997253, 0.00128653, -0.09205297],
       ...
       [ 0.07446931, 0.95775696, -0.13989343, 0.03567281, -0.06780412],
       [ 0.04383769, 0.12820042, 0.91664601, -0.00112968, -0.14217986],
       [ 0.18044085, 0.19102561, 0.59442604, -0.09064547, -0.02683733],
       [-0.08220314, -0.06951903, -0.33450319, 0.13062949, 0.1061413 ],
       [-0.16692593, -0.07122753, -0.21436393, 0.1046549 , 0.09490569],
       [ 0.34585972, -0.05282302, 0.03392856, -0.0200651 , 0.17176314]])
```
[illegible]

In [65]:
```
l=pd.DataFrame(fc.get_factor_variance(),index=['SS loadings','Proportion var','Cumulative var'])
l
```

Out[65]:

	0	1	2	3	4
SS loadings	12.639124	12.027379	7.887294	5.818036	4.984010
Proportion var	0.197525	0.185037	0.121343	0.089513	0.076680
Cumulative var	0.197525	0.382562	0.503905	0.593417	0.670097

Model Building

In [66]:
```
#10. Model building
# State level model (California)
df_tr['state'].nunique()
```

Out[66]: 52

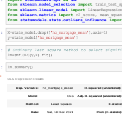
In [109]: `df_tr['state'].value_counts().head(5)  # frequency of state`

Out[109]:
```
California      2826
Texas           1876
New York        1683
Florida         1509
Pennsylvania    1203
Name: state, dtype: int64
```

In [68]:
```
state=df_tr[df_tr["state"]=="California"]
state_model=state.drop(['COUNTYID', 'STATEID', 'state', 'state_ab', 'city', 'place', 'type',
                        'primary', 'zip_code', 'area_code', 'lat', 'lng','pop_bins','Country'],axis
=1)
state_model.shape
```

Out[68]: (2826, 65)

In [69]:
```
sns.boxplot(state_model["hc_mortgage_mean"])
plt.show()
```



In [70]:
```
# remove outliers
state_model=state_model[state_model["hc_mortgage_mean"]<3900]
state_model.shape
```

Out[70]: (2813, 65)

In [71]:
```
import statsmodels.api as smf
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [72]:
```
X=state_model.drop(["hc_mortgage_mean"],axis=1)
y=state_model["hc_mortgage_mean"]
```

In [73]:
```
# Ordinary least square method to select significant variables
lm=smf.OLS(y,X).fit()
```

In [74]: `lm.summary()`

Out[74]:

OLS Regression Results

Dep. Variable:	hc_mortgage_mean	R-squared (uncentered):	0.999
Model:	OLS	Adj. R-squared (uncentered):	0.999
Method:	Least Squares	F-statistic:	2.744e+04
Date:	Sat, 18 Dec 2021	Prob (F-statistic):	0.00
Time:	19:55:36	Log-Likelihood:	-16335.
No. Observations:	2813	AIC:	3.278e+04
Df Residuals:	2752	BIC:	3.314e+04
Df Model:	61		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
ALand	-6.847e-09	4.95e-09	-1.407	0.160	-1.63e-08	2.68e-09
AWater	-6.816e-09	1.55e-07	-0.044	0.965	-3.1e-07	2.96e-07
pop	-0.0025	0.003	-1.138	0.255	-0.010	0.003
male_pop	-0.0010	0.005	-0.220	0.826	-0.010	0.008
female_pop	-0.0025	0.005	-0.537	0.591	-0.012	0.007
rent_mean	-0.0252	0.020	-1.285	0.199	-0.064	0.013
rent_median	0.0106	0.016	0.671	0.502	-0.020	0.042
rent_stdev	0.0452	0.013	3.572	0.000	0.020	0.070
rent_sample_weight	-0.0534	0.018	-2.992	0.003	-0.088	-0.018
rent_samples	-0.0259	0.094	-0.276	0.783	-0.210	0.158
rent_gt_10	-64.6959	40.502	-1.597	0.110	-144.113	14.721
rent_gt_15	78.8307	33.494	2.353	0.019	13.145	144.490
rent_gt_20	24.9603	29.740	0.839	0.401	-33.355	83.275
rent_gt_25	-42.8766	29.022	-1.477	0.140	-99.787	14.028
rent_gt_30	14.2457	29.847	0.477	0.633	-44.094	72.768
rent_gt_35	35.2790	33.249	1.061	0.289	-29.913	100.472
rent_gt_40	-10.8715	32.480	-0.329	0.743	-74.309	52.516
rent_gt_50	-22.4069	25.641	-0.874	0.382	-72.683	27.870
universe_samples	0.0492	0.048	1.019	0.308	-0.046	0.144
used_samples	-0.0183	0.072	-0.253	0.800	-0.160	0.123
hi_mean	-0.0009	0.001	-0.908	0.364	-0.002	0.001
hi_median	0.0010	0.000	2.140	0.032	7.54e-05	0.002
hi_stdev	-0.0006	0.001	-1.158	0.247	-0.002	0.000
hi_sample_weight	-0.0200	0.032	-0.616	0.538	-0.084	0.044
hi_samples	0.0246	0.024	1.029	0.303	-0.022	0.072
family_mean	0.0035	0.001	4.884	0.000	0.002	0.005
family_median	-0.0001	0.000	-0.510	0.610	-0.001	0.000
family_stdev	0.0004	0.000	0.907	0.365	-0.001	0.001
family_sample_weight	0.1156	0.039	2.926	0.003	0.038	0.193
family_samples	-0.0473	0.028	-0.881	0.379	-0.079	0.030
hc_mortgage_median	0.6936	0.006	119.958	0.000	0.682	0.705
hc_mortgage_stdev	0.3366	0.012	28.751	0.000	0.314	0.360
hc_mortgage_sample_weight	-0.8875	0.047	-19.065	0.000	-0.979	-0.796
hc_mortgage_samples	0.2062	0.026	7.869	0.000	0.155	0.257
hc_mean	0.1747	0.035	4.941	0.000	0.105	0.244
hc_median	-0.0725	0.028	-2.602	0.009	-0.128	-0.018
hc_stdev	-0.1282	0.024	-5.236	0.000	-0.176	-0.080
hc_samples	-0.2306	0.043	-5.355	0.000	-0.315	-0.146
hc_sample_weight	0.3195	0.078	4.118	0.000	0.167	0.472
home_equity_second_mortgage	64.7427	121.526	0.533	0.594	-173.547	303.033
second_mortgage	119.8265	124.317	0.964	0.335	-123.839	363.590
home_equity	32.0251	73.582	0.299	0.765	-122.257	166.307
debt	286.7948	47.727	6.009	0.000	193.210	380.379
second_mortgage_cdf	7.6329	10.098	0.756	0.450	-12.169	27.435
home_equity_cdf	5.8287	22.088	0.264	0.792	-37.481	49.138
debt_cdf	107.0585	26.802	3.994	0.000	54.505	159.612
hs_degree	277.3663	390.451	0.710	0.478	-488.240	1042.973
hs_degree_male	-85.5180	194.800	-0.439	0.661	-467.682	296.646
hs_degree_female	-182.1972	198.484	-0.918	0.359	-571.390	206.995
male_age_mean	3.6905	1.298	2.844	0.004	1.146	6.235
male_age_median	-0.4687	0.923	-0.528	0.598	-2.284	1.321
male_age_stdev	2.6719	1.333	2.006	0.045	0.060	5.283
male_age_sample_weight	-0.0143	0.032	-0.445	0.656	-0.049	0.077
male_age_samples	-0.0010	0.005	-0.220	0.826	-0.010	0.008
female_age_mean	-2.9926	1.285	-2.328	0.020	-5.512	-0.472
female_age_median	1.8607	0.925	2.027	0.042	0.071	3.667
female_age_stdev	-3.4923	1.252	-2.788	0.005	-5.946	-1.037
female_age_sample_weight	0.0285	0.030	0.951	0.342	-0.030	0.087
female_age_samples	-0.0025	0.005	-0.536	0.591	-0.012	0.007
pct_own	-46.5784	21.646	-2.150	0.032	-88.052	-4.105
married	-75.7769	29.151	-2.600	0.009	-132.941	-18.619
married_snp	65.7240	68.243	1.053	0.293	-60.167	198.615
separated	47.4662	110.776	0.428	0.668	-169.767	264.656
divorced	-180.5554	49.168	-3.659	0.000	-277.027	-84.084

Omnibus:	260.560	Durbin-Watson:	2.027
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1326.567
Skew:	0.297	Prob(JB):	8.70e-289
Kurtosis:	6.315	Cond. No.	1.00e+16

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 3.56e-12. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

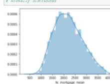
In [75]:
```
X=X[['rent_stdev','rent_sample_weight','rent_gt_15','hi_median','family_mean','family_median','family_s
ample_weight',
     'hc_mortgage_median','hc_mortgage_stdev','hc_mortgage_sample_weight','hc_mortgage_samples',
     'hc_mean','hc_median','hc_stdev','hc_samples','hc_sample_weight','debt','debt_cdf','male_age_mean'
,
     'male_age_stdev','female_age_mean','female_age_median','female_age_stdev','pct_own','married','div
orced']]
X.shape
# we take only those variables which p-value is less than 0.05 means these variables are statistically
 significant.
```
Out[75]: (2813, 26)

In [76]:
```
# Check multicollinearity between explanatory variables
# VIF can be calculated by the formula below: Where Ri2 represents the unadjusted coefficient of determ
ination
# for regressing the ith independent variable on the remaining ones. The reciprocal of VIF is known as
 tolerance.
vif=pd.DataFrame()
vif["VIF factor"]=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["features"]=X.columns
```

In [77]:
```
vif.round(1)
# there are maximum high multicollinearity but we do not remove it.
```
Out[77]:

	VIF factor	features
0	354	rent_stdev
1	6.8	rent_sample_weight
2	153.7	rent_gt_15
3	960	hi_median
4	314.6	family_mean
5	186.9	family_median
6	27.9	family_sample_weight
7	66.3	hc_mortgage_median
8	29.1	hc_mortgage_stdev
9	37.6	hc_mortgage_sample_weight
10	36.9	hc_mortgage_samples
11	185.2	hc_mean
12	102.4	hc_median
13	163	hc_stdev
14	566	hc_samples
15	85.7	hc_sample_weight
16	272.4	debt
17	210	debt_cdf
18	260.6	male_age_mean
19	306.9	male_age_stdev
20	793.6	female_age_mean
21	377.7	female_age_median
22	299.9	female_age_stdev
23	304	pct_own
24	629	married
25	73	divorced

In [78]:
```
sns.distplot(y)
plt.show()
# Normally distributed
```



In [79]:
```
# split data in train and validation set
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1234)
```

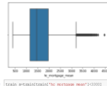
In [80]:
```
# linear regression model building
lr = LinearRegression()
lr.fit( X_train, y_train )
y_pred = lr.predict( X_test )
```

In [81]:
```
lr.coef_
```
Out[81]:
```
array([ 4.63870690e-02, -3.66275997e-02,  6.64644677e+01,  5.72477240e-04,
        3.42936096e-03, -2.53785846e-03,  5.83015675e-02,  6.95132265e-01,
        3.35444319e-01, -8.58468369e-01,  2.02124412e-01,  1.73551970e-01,
       -4.41829520e-02, -1.34020705e-01, -2.58764943e-01,  4.37134776e-01,
        1.73247295e+02,  2.66489469e+01,  2.60195164e+00,  8.45457988e-01,
       -1.50241227e+00,  9.59010004e-01, -2.81812141e+00, -7.40744681e+01,
       -9.07421774e+01, -1.79670842e+02])
```

In [82]:
```
r2_score(y_pred=y_pred, y_true=y_test)
# 98% variation explained by this model
```
Out[82]: 0.9816488244809747

In [83]:
```
rmse=mean_squared_error(y_pred=y_pred, y_true=y_test, squared=False)  # root mean square error
rmse
```
Out[83]: 87.57595151313595

In [84]:
```
# National level model
sns.boxplot(train['hc_mortgage_mean'])
plt.show()
```



In [85]:
```
train_s=train[train['hc_mortgage_mean']<3300]
train_s.shape
```
Out[85]: (26173, 65)

In [86]:
```
X=train_s.drop(['hc_mortgage_mean'],axis=1)
y=train_s['hc_mortgage_mean']
```

In [87]:
```
# In statistics, ordinary least squares (OLS) is a type of linear least squares method for estimating
# the unknown parameters in a linear regression model.
lm=smf.OLS(y,X).fit()
lm.summary()
```
Out[87]:

OLS Regression Results

Dep. Variable:	hc_mortgage_mean	R-squared (uncentered):	0.999
Model:	OLS	Adj. R-squared (uncentered):	0.999
Method:	Least Squares	F-statistic:	3.269e+05
Date:	Sat, 18 Dec 2021	Prob (F-statistic):	0.00
Time:	19:55:40	Log-Likelihood:	-1.4512e+05
No. Observations:	26173	AIC:	2.904e+05
Df Residuals:	26113	BIC:	2.909e+05
Df Model:	60		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
ALand	-6.425e-10	3.3e-10	-1.950	0.051	-1.29e-09	3.38e-12
AWater	-5.818e-10	1.89e-09	-0.308	0.758	-4.29e-09	3.12e-09
pop	-0.0010	0.001	-1.299	0.194	-0.003	0.001
male_pop	0.0003	0.001	0.222	0.824	-0.002	0.003
female_pop	-0.0012	0.001	-1.003	0.316	-0.004	0.001
rent_mean	0.0014	0.006	0.249	0.804	-0.010	0.012
rent_median	0.0050	0.005	1.073	0.283	-0.004	0.014
rent_stdev	0.0093	0.003	2.696	0.007	0.003	0.016
rent_sample_weight	-0.0221	0.004	-5.217	0.000	-0.030	-0.014
rent_samples	0.0362	0.022	1.648	0.099	-0.007	0.079
rent_gt_10	22.9772	7.928	2.898	0.004	7.438	38.517
rent_gt_15	-1.4326	6.070	-0.236	0.813	-13.331	10.466
rent_gt_20	13.4194	5.701	2.354	0.019	2.244	24.594
rent_gt_25	-7.0957	6.018	-1.179	0.238	-18.892	4.700
rent_gt_30	-6.6303	6.690	-0.991	0.322	-19.744	6.483
rent_gt_35	16.2987	7.725	2.110	0.035	1.158	31.440
rent_gt_40	-9.0299	7.725	-1.169	0.242	-24.171	6.111
rent_gt_50	-2.3855	5.831	-0.409	0.682	-13.814	9.043
universe_samples	-0.0089	0.009	-1.029	0.303	-0.026	0.008
used_samples	-0.0317	0.018	-1.724	0.085	-0.068	0.004
hi_mean	-6.638e-05	0.000	-0.372	0.710	-0.000	0.000
hi_median	0.0005	0.000	4.066	0.000	0.000	0.001
hi_stdev	7.183e-05	0.000	0.505	0.613	-0.000	0.000
hi_sample_weight	0.0341	0.008	4.083	0.000	0.018	0.051
hi_samples	-0.0111	0.005	-2.055	0.040	-0.022	-0.001
family_mean	0.0009	0.000	6.187	0.000	0.001	0.001
family_median	-0.0006	9.53e-05	-6.464	0.000	-0.001	-0.000
family_stdev	0.0002	0.000	1.896	0.058	-8.09e-06	0.000
family_sample_weight	0.0301	0.010	2.986	0.003	0.010	0.050
family_samples	-0.0116	0.007	-1.602	0.109	-0.026	0.003
hc_mortgage_median	0.7508	0.002	415.460	0.000	0.747	0.754
hc_mortgage_stdev	0.4125	0.003	138.379	0.000	0.407	0.418
hc_mortgage_sample_weight	-0.2270	0.007	-31.982	0.000	-0.241	-0.213
hc_mortgage_samples	0.0763	0.005	14.175	0.000	0.066	0.087
hc_mean	0.1365	0.011	12.276	0.000	0.115	0.158
hc_median	-0.0169	0.009	-1.810	0.070	-0.035	0.001
hc_stdev	-0.0476	0.007	-6.401	0.000	-0.062	-0.033
hc_samples	-0.0784	0.007	-10.820	0.000	-0.093	-0.064
hc_sample_weight	0.1194	0.012	10.259	0.000	0.097	0.142
home_equity_second_mortgage	-162.4076	37.631	-4.316	0.000	-236.165	-88.650
second_mortgage	186.9563	36.938	5.061	0.000	114.555	259.357
home_equity	120.9414	22.981	5.263	0.000	75.898	165.985
debt	96.1471	9.786	9.825	0.000	76.965	115.329
second_mortgage_cdf	-0.4256	2.469	-0.172	0.863	-5.265	4.414
home_equity_cdf	31.5924	6.070	5.205	0.000	19.695	43.489
debt_cdf	20.6400	5.903	3.497	0.000	9.070	32.209
hs_degree	61.1087	105.687	0.578	0.563	-146.043	268.261
hs_degree_male	-9.9872	51.645	-0.193	0.847	-111.214	91.240
hs_degree_female	-69.9029	54.851	-1.274	0.203	-177.414	37.608
male_age_mean	1.0461	0.311	3.367	0.001	0.437	1.655
male_age_median	-0.2515	0.206	-1.220	0.222	-0.656	0.153
male_age_stdev	-0.2727	0.318	-0.859	0.390	-0.895	0.350
male_age_sample_weight	0.0034	0.008	0.412	0.681	-0.013	0.019
male_age_samples	0.0003	0.001	0.221	0.825	-0.002	0.003
female_age_mean	-0.6835	0.307	-2.230	0.026	-1.284	-0.083
female_age_median	0.4733	0.211	2.238	0.025	0.059	0.888
female_age_stdev	-0.7810	0.300	-2.600	0.009	-1.370	-0.192
female_age_sample_weight	0.0180	0.008	2.333	0.020	0.003	0.033
female_age_samples	-0.0012	0.001	-0.973	0.331	-0.004	0.001
pct_own	-44.7079	5.377	-8.315	0.000	-55.247	-34.169
married	1.7393	6.313	0.276	0.783	-10.634	14.113
married_snp	52.3657	17.422	3.006	0.003	18.218	86.513
separated	-62.0508	27.190	-2.282	0.022	-115.344	-8.757
divorced	-57.4855	10.688	-5.378	0.000	-78.435	-36.536

Omnibus:	6175.848	Durbin-Watson:	2.008
Prob(Omnibus):	0.000	Jarque-Bera (JB):	291507.734
Skew:	-0.285	Prob(JB):	0.00
Kurtosis:	19.340	Cond. No.	1.00e+16

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1e+16. This might indicate that there are
strong multicollinearity or other numerical problems.

In [88]:
```
X=X[['rent_stdev','rent_sample_weight','rent_gt_10','rent_gt_20','rent_gt_35','hi_median','hi_sample_we
ight','hi_samples',
     'family_mean','family_median','family_sample_weight','hc_mortgage_median','hc_mortgage_stdev',
     'hc_mortgage_sample_weight','hc_mortgage_samples','hc_mean','hc_stdev','hc_samples','hc_sample_wei
ght',
     'home_equity_second_mortgage','second_mortgage','home_equity','debt','home_equity_cdf','debt_cdf',
'male_age_mean',
     'female_age_mean','female_age_median','female_age_stdev','female_age_sample_weight','pct_own','mar
ried_snp',
     'separated','divorced']]
X.shape
# we take only those variables which p-value is less than 0.05 means these variables are statistically
 significant.
```
Out[88]: (26173, 34)

In [89]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1234)
```

In [90]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [91]:
```
X_train_scale = scaler.fit_transform(X_train)
```

In [92]:
```
X_test_scale = scaler.transform(X_test)
```

In [93]:
```
lr.fit( X_train_scale, y_train )
y_pred = lr.predict( X_test_scale )
```

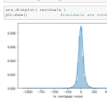
In [94]:
```
lr.coef_
```
Out[94]:
```
array([ 2.08754153e+00, -7.22943284e+00,  1.29809439e+00,  1.23199013e+00,
        2.65528494e-01,  1.51284734e+01,  2.15094940e+01, -1.81115800e+01,
        3.32068742e+00, -2.47474283e+01,  2.41387104e+00,  4.39603762e+02,
        9.71442033e+01, -4.40606731e+01,  3.39884869e+01,  2.36823229e+01,
       -2.26731536e+00, -1.78576452e+01,  2.14703135e+01, -6.27670967e+00,
        7.68640905e+00,  7.11955010e+01,  1.38748370e+01,  7.17807399e+00,
        3.72373514e+00,  3.63559337e+00, -2.38539392e+00,  3.00655114e+00,
       -2.73625454e+00,  5.02750764e+00, -1.08734513e+01,  2.41722495e+00,
       -1.50231113e+00, -3.33768147e+01])
```

In [95]:
```
r2_score(y_pred=y_pred, y_true=y_test) # 98% variation explained by this model
```
Out[95]: 0.9883562760386579

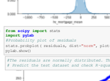
In [96]:
```
rmse=mean_squared_error(y_pred=y_pred, y_true=y_test, squared=False) #  root mean square error is 62
rmse
```
Out[96]: 62.83581674112224

In [97]:
```
residuals = y_test - y_pred
```

In [98]:
```
sns.distplot( residuals )
plt.show()             #residuals are normally distributed.
```



In [99]:
```
from scipy import stats
import pylab
#Probability plot of residuals
stats.probplot( residuals, dist="norm", plot=pylab )
pylab.show()
#The residuals are normally distributed. This model is assumed to be correct.
# Predict the test dataset and check R-squared, RMSE value.
```



In [100]:

In [101]:
```
# split output and input variables
X1=df_te[['rent_stdev','rent_sample_weight','rent_gt_10','rent_gt_20','rent_gt_35','hi_median','hi_samp
le_weight','hi_samples',
     'family_mean','family_median','family_sample_weight','hc_mortgage_median','hc_mortgage_stdev',
     'hc_mortgage_sample_weight','hc_mortgage_samples','hc_mean','hc_stdev','hc_samples','hc_sample_wei
ght',
     'home_equity_second_mortgage','second_mortgage','home_equity','debt','home_equity_cdf','debt_cdf',
'male_age_mean',
     'female_age_mean','female_age_median','female_age_stdev','female_age_sample_weight','pct_own','mar
ried_snp',
     'separated','divorced']]
```

In [102]:
```
y1=df_te['hc_mortgage_mean']
```

In [103]:
```
# scale input variables
X1_scale = scaler.transform(X1)
```

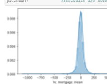
In [104]:
```
# predict y1
y1_pred = lr.predict( X1_scale )
```

In [105]:
```
r2_score(y_pred=y1_pred, y_true=y1) # R-square i.e coefficient of determinant explained 98% variation.
```
Out[105]: 0.9881864281333671

In [106]:
```
rmse=mean_squared_error(y_pred=y1_pred, y_true=y1, squared=False)
rmse
# rmse is 68 which means error is low so we can use this model.
```
Out[106]: 68.61073260062234

In [107]:
```
residuals1 = y1 - y1_pred
```

In [108]:
```
sns.distplot( residuals )
plt.show()             #residuals are normally distributed.
```



End