REAL ESTATE

Project1

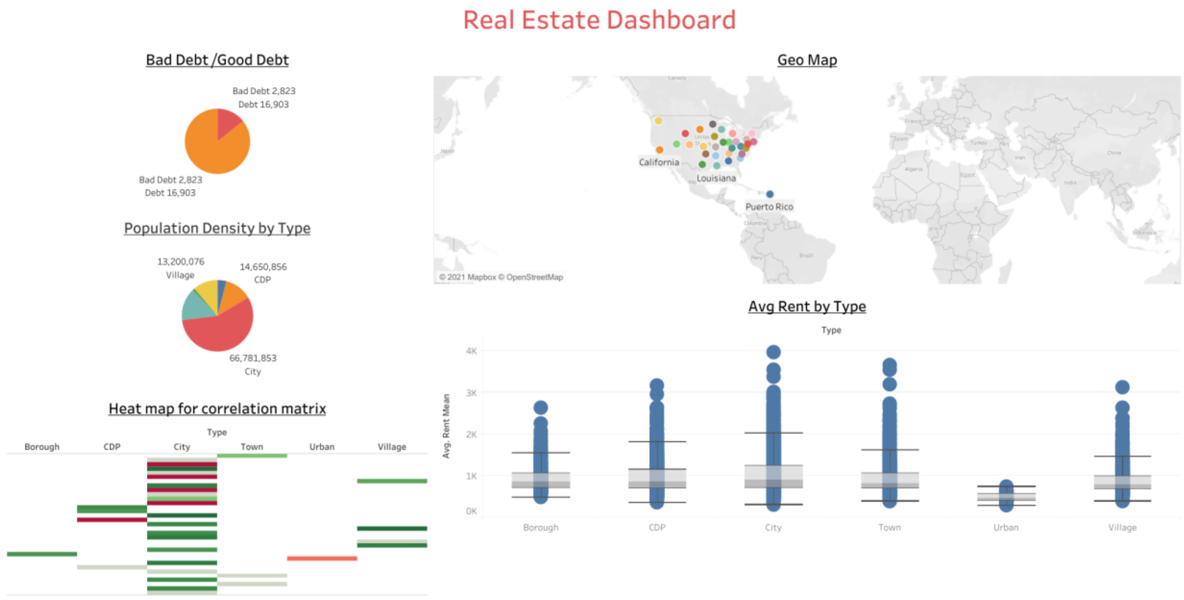
JULY 6, 2021

rayasalehalshammri@gmail.com

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.

Dashboard image:



Dashboard link:

https://public.tableau.com/views/RealEstateProject-1_16255218497970/RealEstateDashboard?:language=en-US&:display_count=n&:origin=viz_share_link

project in python link:

https://github.com/rayas711/Real-Estate-Project1/blob/master/real-estate-project1-raya.ipynb

Output image:

In [1]:
```python
import time
import random
from math import *
import operator
import pandas as pd
import numpy as np
```

In [2]:
```python
#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)
```

Week1, 1- Import data:

In [3]:
```python
df_train=pd.read_csv("../input/realestateproject1/train.csv")
df_test=pd.read_csv("../input/realestateproject1/test.csv")
```

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

Out[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')
```

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

Out[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',
```

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

Out[6]: 27321

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

Out[7]: 11709

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

Out[8]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_median	female_a
0	267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	...	44.48629	45.33333	
1	246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	...	36.48391	37.58333	
2	245683	NaN	140	63	18	Indiana	IN	Danville	Danville	City	...	42.15810	42.83333	
3	279653	NaN	140	127	72	Puerto Rico	PR	San Juan	Guaynabo	Urban	...	47.77526	50.58333	
4	247218	NaN	140	161	20	Kansas	KS	Manhattan	Manhattan City	City	...	24.17693	21.58333	

5 rows × 80 columns

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

Out[9]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	...	female_age_mean	female_age_median	fe
0	255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	...	34.78682	33.75000	
1	252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	...	44.23451	46.66667	
2	276314	NaN	140	15	42	Pennsylvania	PA	Pine City	Millerton	Borough	...	41.62426	44.50000	
3	248614	NaN	140	231	21	Kentucky	KY	Monticello	Monticello City	City	...	44.81200	48.00000	
4	286865	NaN	140	355	48	Texas	TX	Corpus Christi	Edroy	Town	...	40.66618	42.66667	

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

Out[10]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	...	femal
count	27321.000000	0.0	27321.0	27321.000000	27321.000000	27321.000000	27321.000000	27321.000000	27321.000000	2.732100e+04	...	2
mean	257331.996303	NaN	140.0	85.646426	28.271806	50081.999524	596.507668	37.508813	-91.288394	1.295106e+08	...	
std	21343.859725	NaN	0.0	98.333097	16.392846	29558.115660	232.497482	5.588268	16.343816	1.275531e+09	...	
min	220342.000000	NaN	140.0	1.000000	1.000000	602.000000	201.000000	17.929085	-165.453872	4.113400e+04	...	
25%	238816.000000	NaN	140.0	29.000000	13.000000	26554.000000	405.000000	33.899064	-97.816067	1.799408e+06	...	
50%	257220.000000	NaN	140.0	63.000000	28.000000	47715.000000	614.000000	38.755183	-86.554374	4.866940e+06	...	
75%	275818.000000	NaN	140.0	109.000000	42.000000	77093.000000	801.000000	41.380606	-79.782503	3.359820e+07	...	
max	294334.000000	NaN	140.0	840.000000	72.000000	99925.000000	989.000000	67.074018	-65.379332	1.039510e+11	...	

8 rows × 74 columns

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

Out[11]:

	UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	zip_code	area_code	lat	lng	ALand	...	female
count	11709.000000	0.0	11709.0	11709.000000	11709.000000	11709.000000	11709.000000	11709.000000	11709.000000	1.170900e+04	...	1
mean	257525.004783	NaN	140.0	85.710650	28.489196	50123.418396	593.598514	37.405491	-91.340229	1.095500e+08	...	
std	21466.372658	NaN	0.0	99.304334	16.607262	29775.134038	232.074263	5.625904	16.407818	7.624940e+08	...	
min	220336.000000	NaN	140.0	1.000000	1.000000	601.000000	201.000000	17.965835	-166.770979	8.299000e+03	...	
25%	238819.000000	NaN	140.0	29.000000	13.000000	25570.000000	404.000000	33.919813	-97.816561	1.718660e+06	...	
50%	257651.000000	NaN	140.0	61.000000	28.000000	47362.000000	612.000000	38.618092	-86.643344	4.835000e+06	...	
75%	276300.000000	NaN	140.0	109.000000	42.000000	77406.000000	787.000000	41.232973	-79.697311	3.204540e+07	...	
max	294333.000000	NaN	140.0	810.000000	72.000000	99929.000000	989.000000	64.804269	-65.695344	5.520166e+10	...	

8 rows × 74 columns

In [12]: `df_train.info()`

In [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
```

In [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
```

Week 1, 2- Figure out the primary key and look for the requirement of indexing

In [14]:
```
#UID is a 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)
```

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

Out[15]:

UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	...	female_age_mean	female_age_median	female
267822	NaN	140	53	36	New York	NY	Hamilton	Hamilton	City	tract	...	44.48629	45.33333	
246444	NaN	140	141	18	Indiana	IN	South Bend	Roseland	City	tract	...	36.48391	37.58333	

2 rows × 79 columns

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

Out[16]:

UID	BLOCKID	SUMLEVEL	COUNTYID	STATEID	state	state_ab	city	place	type	primary	...	female_age_mean	female_age_median	female
255504	NaN	140	163	26	Michigan	MI	Detroit	Dearborn Heights City	CDP	tract	...	34.78682	33.75000	
252676	NaN	140	1	23	Maine	ME	Auburn	Auburn City	City	tract	...	44.23451	46.66667	

2 rows × 79 columns

Week 1, 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

In [17]:
```
#percantage 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
```

Out[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

In [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
```

Out[18]:

	Percantage of missing values
BLOCKID	42.857143
hc_samples	1.061455
hc_mean	1.061455
hc_median	1.061455
hc_stdev	1.061455

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

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

In [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', 'r
```

In [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', 'r
```

In [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)
```

In [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)
```

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

Out[25]: 0

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

Out[26]: 0

Week 1, Exploratory Data Analysis (EDA):

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

In [27]:
```
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_mortga
pysqldf = lambda q: sqldf(q, globals())
df_train_location_mort_pct=pysqldf(q1)
```

In [28]:
```
df_train_location_mort_pct.head()
```

Out[28]:

	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

In [29]:
```
import plotly.express as px
import plotly.graph_objects as go
```

In [30]:
```
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
```

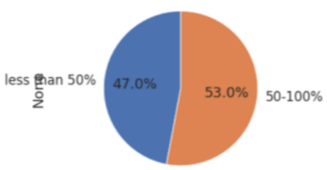
```
        lakecolor = "rgb(255, 255, 255)",
        showsubunits = True,
        showcountries = True,
        resolution = 50,
        projection = dict(
            type = 'conic conformal',
            rotation_lon = -100
        ),
        lonaxis = dict(
            showgrid = True,
            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

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

[31]:
```
df_train['bad_debt']=df_train['second_mortgage']+df_train['home_equity']-df_train['home_equity_second_mortgage']
```

[32]:
```
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()
```



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

[33]:

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

[33]:

```
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')
```

+ Code + Markdown

[34]:

```
#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)
```

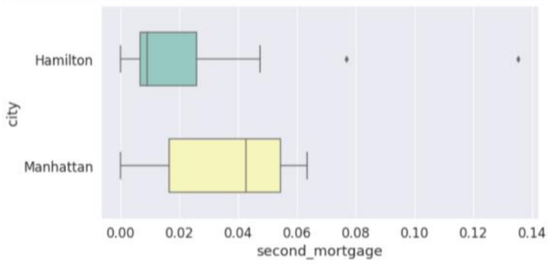
[34]:

	COUNTYID	STATEID	state	state_ab	city	place	type	primary	zip_code	area_code	...	female_age_stdev	female_age_sample_weight	female_age_samples	pct_own	m
UID																
267822	53	36	New York	NY	Hamilton	Hamilton	City	tract	13346	315	...	22.51276	685.33845	2618.0	0.79046	(
263797	21	34	New Jersey	NJ	Hamilton	Yardville	City	tract	8610	609	...	24.05831	732.58443	3124.0	0.64400	(
270979	17	39	Ohio	OH	Hamilton	Hamilton City	Village	tract	45015	513	...	22.66500	565.32725	2528.0	0.61278	(
259028	95	28	Mississippi	MS	Hamilton	Hamilton	CDP	tract	39746	662	...	22.79602	483.01311	1954.0	0.83241	(

4 rows × 79 columns

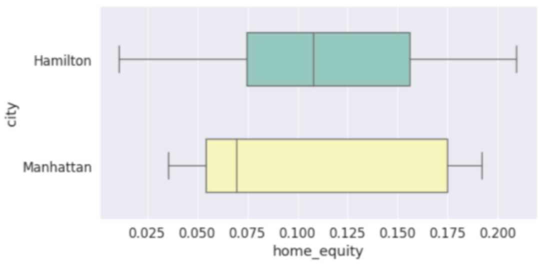
[35]:

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

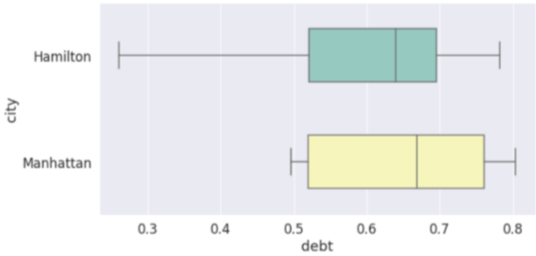


[36]:

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



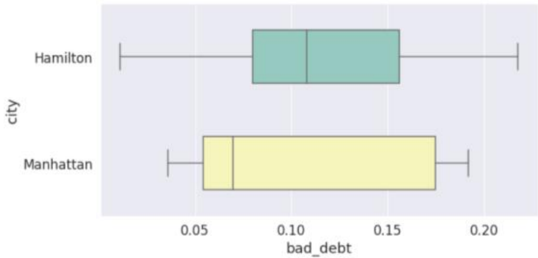
[37]:
```
plt.figure(figsize=(10,5))
sns.boxplot(data=df_box_city,x='debt', y='city',width=0.5,palette="Set3")
plt.show()
```



+ Code + Markdown

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

#Manhattan has higher metrics compared to Hamilton
```



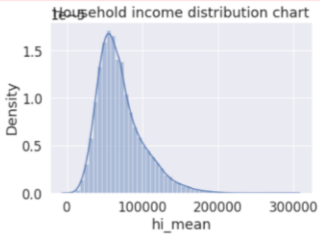
+ Code + Markdown

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

<u>Please notice that</u> the `distplot` is a deprecated function and will be removed in a future version. So, either use `displot` or `histplot`.

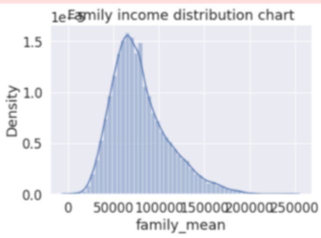
[39]:
```
sns.distplot(df_train['hi_mean'])
plt.title('Household income distribution chart')
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: 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).
```



[40]:
```
sns.distplot(df_train['family_mean'])
plt.title('Family income distribution chart')
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: 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).
```

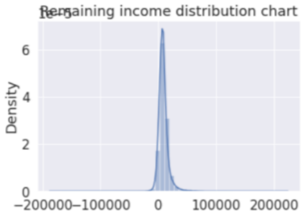


[41]:

```
sns.distplot(df_train['family_mean']-df_train['hi_mean'])
plt.title('Remaining income distribution chart')
plt.show()
#Income distribution almost has normality in its distrbution
```

/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: 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).

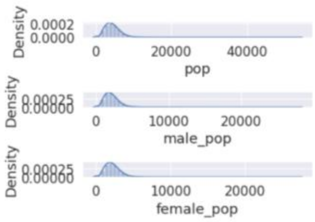


Week 2, Exploratory Data Analysis (EDA):

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):

[42]:

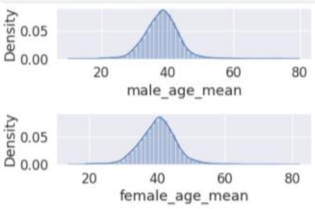
```
fig,(ax1,ax2,ax3)=plt.subplots(3,1)
sns.distplot(df_train['pop'],ax=ax1)
sns.distplot(df_train['male_pop'],ax=ax2)
sns.distplot(df_train['female_pop'],ax=ax3)
plt.subplots_adjust(wspace=0.8,hspace=0.8)
plt.tight_layout()
plt.show()
```



+ Code + Markdown

[43]:

```
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()
```



+ Code + Markdown

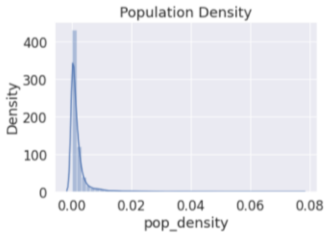
a) Use pop and ALand variables to create a new field called population density

[44]:

```
df_train['pop_density']=df_train['pop']/df_train['ALand']
df_test['pop_density']=df_test['pop']/df_test['ALand']
```

[45]:

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



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

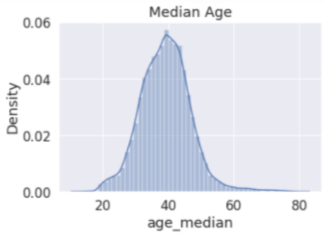
[46]:
```
df_train['age_median']=(df_train['male_age_median']+df_train['female_age_median'])/2
df_test['age_median']=(df_test['male_age_median']+df_test['female_age_median'])/2
```

[47]:
```
df_train[['male_age_median','female_age_median','male_pop','female_pop','age_median']].head()
```

[47]:

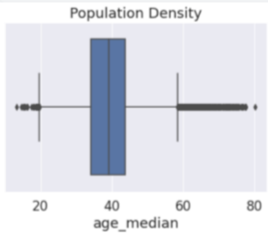
	male_age_median	female_age_median	male_pop	female_pop	age_median
UID					
267822	44.00000	45.33333	2612	2618	44.666665
246444	32.00000	37.58333	1349	1284	34.791665
245683	40.83333	42.83333	3643	3238	41.833330
279653	48.91667	50.58333	1141	1559	49.750000
247218	22.41667	21.58333	2586	3051	22.000000

[48]:
```
sns.distplot(df_train['age_median'])
plt.title('Median Age')
plt.show()
# Age of population is mostly between 20-60
# Majority are of age around 40
# Median age distribution has a gaussian distribution
# Some right skewness is noticed
```



+ Code + Markdown

[49]:
```
sns.boxplot(df_train["age_median"])
plt.title("Population Density")
plt.show()
```



+ Code + Markdown

Week 2, 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.

[50]:
```
df_train['pop'].describe()
```

[50]:
```
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
```

[51]:
```
df_train['pop_bins']=pd.cut(df_train['pop'],bins=5,labels=['very low','low','medium','high','very high'])
```

[52]:
```python
df_train[['pop','pop_bins']]
```

[52]:

UID	pop	pop_bins
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 × 2 columns

[53]:
```python
df_train['pop_bins'].value_counts()
```

[53]:
```
very low     27058
low            246
medium           9
high             7
very high        1
Name: pop_bins, dtype: int64
```

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

[54]:
```python
df_train.groupby(by='pop_bins')[['married','separated','divorced']].count()
```

[54]:

pop_bins	married	separated	divorced
very low	27058	27058	27058
low	246	246	246
medium	9	9	9
high	7	7	7
very high	1	1	1

[55]:
```python
df_train.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean", "median"])
#Very high population group has more married people and less percantage of separated and divorced couples
#In very low population groups, there are more divorced people
```

[55]:

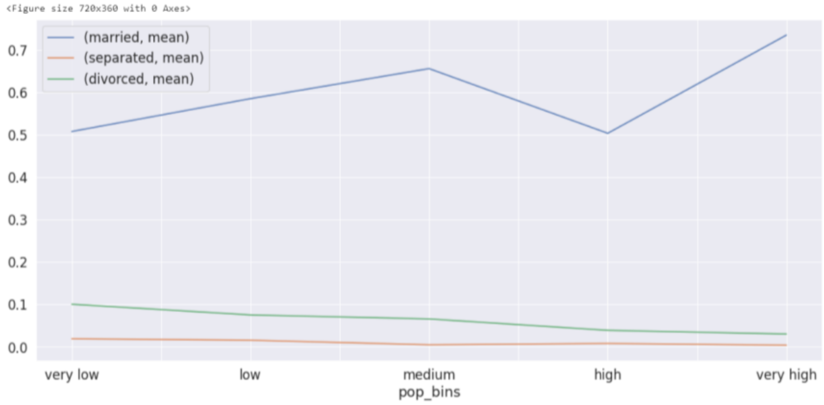
	married		separated		divorced	
	mean	median	mean	median	mean	median
pop_bins						
very low	0.507548	0.524680	0.019126	0.013650	0.100504	0.096020
low	0.584894	0.593135	0.015833	0.011195	0.075348	0.070045
medium	0.655737	0.618710	0.005003	0.004120	0.065927	0.064890
high	0.503359	0.335660	0.008141	0.002500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

+ Code + Markdown

b) Visualize using appropriate chart type

[56]:
```python
plt.figure(figsize=(10,5))
pop_bin_married=df_train.groupby(by='pop_bins')[['married','separated','divorced']].agg(["mean"])
pop_bin_married.plot(figsize=(18,8))
plt.legend(loc='best')
plt.show()
```

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



Week 2, 3. Please detail your observations for rent as a percentage of income at an overall level, and for different states.

[57]:
```
rent_state_mean=df_train.groupby(by='state')['rent_mean'].agg(["mean"])
rent_state_mean.head()
```

[57]:

	mean
state	
Alabama	774.004927
Alaska	1185.763570
Arizona	1097.753511
Arkansas	720.918575
California	1471.133857

[58]:
```
income_state_mean=df_train.groupby(by='state')['family_mean'].agg(["mean"])
income_state_mean.head()
```

[58]:

	mean
state	
Alabama	67030.064213
Alaska	92136.545109
Arizona	73328.238798
Arkansas	64765.377850
California	87655.470820

[59]:
```
rent_perc_of_income=rent_state_mean['mean']/income_state_mean['mean']
rent_perc_of_income.head(10)
```

[59]:
```
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
```

[60]:
```
#overall level rent as a percentage of income
sum(df_train['rent_mean'])/sum(df_train['family_mean'])
```

[60]: 0.013358170721473864

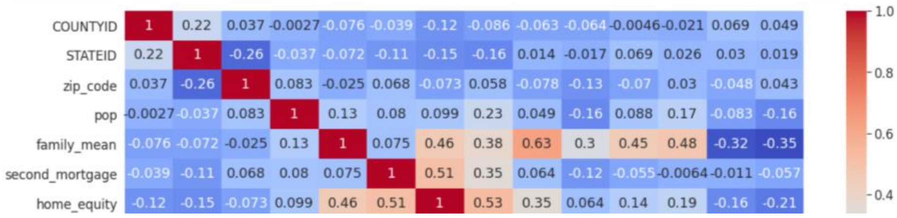
Week 2, 4. Perform correlation analysis for all the relevant variables by creating a heatmap. Describe your findings.

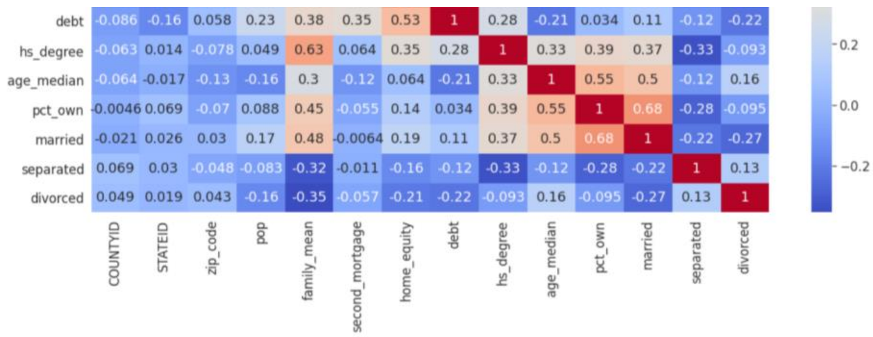
[61]:
```
df_train.columns
```

[61]:
```
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',
```

[62]:
```
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()
```

[63]:
```
plt.figure(figsize=(20,10))
sns.heatmap(cor,annot=True,cmap='coolwarm')
plt.show()
#High positive correaltion is noticed between pop, male_pop and female_pop
#High positive correaltion is noticed between rent_mean,hi_mean, family_mean,hc_mean
```





Week 3, 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

[65]:

```
from sklearn.decomposition import FactorAnalysis
from factor_analyzer import FactorAnalyzer
```

[66]:

```
fa=FactorAnalyzer(n_factors=5)
fa.fit_transform(df_train.select_dtypes(exclude= ('object','category')))
fa.loadings_
```

[66]:

```
array([[-1.12589171e-01,  1.95646470e-02, -2.39331085e-02,
        -6.27632651e-02,  4.23474734e-02],
       [-1.10186764e-01,  1.33506211e-02,  2.79651256e-02,
        -1.49825865e-01,  1.10838806e-01],
       [-8.28678653e-02,  5.16372378e-02, -1.36451869e-01,
        -4.98918613e-02, -1.04024842e-01],
       [ 1.80961133e-02,  1.92013752e-02,  5.81329950e-03,
         2.64842763e-02, -6.12442809e-03],
       [ 9.02324766e-02, -9.72544292e-02, -6.54601353e-02,
        -1.33145903e-01, -1.48594597e-01],
       [-1.07335685e-02, -4.12376822e-02,  1.45853484e-01,
         8.80433198e-03,  1.08227568e-01],
       [-4.28796986e-02, -2.09780219e-02,  3.66726862e-02,
        -9.45597420e-02,  5.91380528e-02],
       [-2.44243063e-03, -1.53245410e-02, -2.68300860e-03,
        -4.52473049e-02,  2.37240653e-02],
       [ 7.92164302e-02,  9.57453324e-01, -8.71151615e-02,
        -6.59923641e-03, -3.97273239e-02],
       [ 7.39808169e-02,  9.18750520e-01, -1.08834837e-01,
        -2.79371562e-02, -3.93153705e-02],
       [ 8.06598875e-02,  9.47839224e-01, -6.08006491e-02,
         1.53627116e-02, -3.86977314e-02],
```

<< "if an error occur in FactorAnalysis that maybe the package doesn't downloaded. So, to download it: !pip install Factor_Analyzer or pip install Factor_Analyzer

Week 4, Data Modeling :

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.

[67]:

```
df_train.columns
```

[67]:

```
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')
```

[68]:
```
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)
```

[69]:
```
df_train['type'].unique()
```

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

[70]:
```
df_test.replace(type_dict,inplace=True)
```

[71]:
```
df_test['type'].unique()
```

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

+ Code + Markdown

[72]:
```
feature_cols=['COUNTYID','STATEID','zip_code','type','pop', 'family_mean',
         'second_mortgage', 'home_equity', 'debt','hs_degree',
          'age_median','pct_own', 'married','separated', 'divorced']
```

[73]:
```
x_train=df_train[feature_cols]
y_train=df_train['hc_mortgage_mean']
```

[74]:
```
x_test=df_test[feature_cols]
y_test=df_test['hc_mortgage_mean']
```

[75]:
```
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
```

[76]:
```
x_train.head()
```

[76]:

UID	COUNTYID	STATEID	zip_code	type	pop	family_mean	second_mortgage	home_equity	debt	hs_degree	age_median	pct_own	married	separated	divorced
267822	53	36	13346	1	5230	67994.14790	0.02077	0.08919	0.52963	0.89288	44.666665	0.79046	0.57851	0.01240	0.08770
246444	141	18	46616	1	2633	50670.10337	0.02222	0.04274	0.60855	0.90487	34.791665	0.52483	0.34886	0.01426	0.09030
245683	63	18	46122	1	6881	95262.51431	0.00000	0.09512	0.73484	0.94288	41.833330	0.85331	0.64745	0.01607	0.10657
279653	127	72	927	2	2700	56401.68133	0.01086	0.01086	0.52714	0.91500	49.750000	0.65037	0.47257	0.02021	0.10106
247218	161	20	66502	1	5637	54053.42396	0.05426	0.05426	0.51938	1.00000	22.000000	0.13046	0.12356	0.00000	0.03109

[77]:
```
sc=StandardScaler()
x_train_scaled=sc.fit_transform(x_train)
x_test_scaled=sc.fit_transform(x_test)
```

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

[78]:
```
linereg=LinearRegression()
linereg.fit(x_train_scaled,y_train)
```

[78]: LinearRegression()

[79]:
```
y_pred=linereg.predict(x_test_scaled)
```

[80]:
```
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)))
#The Accuracy and R2 score are good, but still will investigate the model performance at state level
```

```
Overall R2 score of linear regression model 0.7348210754610929
Overall RMSE of linear regression model 323.1018894984635
```

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

[81]:
```python
state=df_train['STATEID'].unique()
state[0:5]
#Picking a few iDs 20,1,45,6
```

[81]:
```
array([36, 18, 72, 20,  1])
```

[82]:
```python
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 :- 0.6046603766461811
Overall RMSE of linear regression model for state, 20 :- 307.9718899931471


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


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

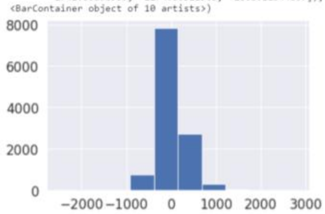
+ Code + Markdown

[83]:
```python
#Check the residuals
residuals=y_test-y_pred
residuals
```

[83]:
```
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
```

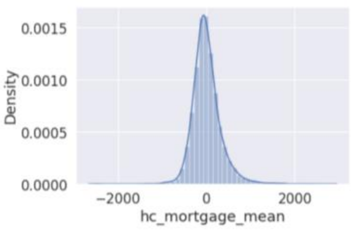
[84]:
```python
plt.hist(residuals)
#Normal distribution of residuals
```

[84]:
```
(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>)
```



[85]:
```python
sns.distplot(residuals)
```

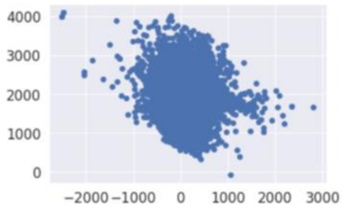
[85]:
```
<AxesSubplot:xlabel='hc_mortgage_mean', ylabel='Density'>
```



[86]:

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

[86]: <matplotlib.collections.PathCollection at 0x7f97f45517d0>



Please click on the project link above to see the output in more suitable way!