Basic Importing

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math as m
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

Importing Data

In [2]:

```
california_data = pd.read_csv('california_housing_data.csv')#main data
california_cities = pd.read_csv('cal_cities_lat_long.csv')#city locations
```

Cleaning the Data

In [209]:

```
california_data.describe()
```

Out[209]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	p
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	2064
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	142
std	2.003532	2.135952	12.585558	2181.615252	421.385070	113
min	-124.350000	32.540000	1.000000	2.000000	1.000000	
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	78
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	116
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	172
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	3568

note 20640 total rows, only 5% can be taken out giving 19608 rows still have to be included at the end of cleaning

In [210]:

```
california_data.isnull().sum()
```

Out[210]:

```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

The cell below drops rows with missing values

In [4]:

```
housing_raw=california_data.dropna()
housing_raw.isnull().sum()
```

Out[4]:

```
longitude             0
latitude              0
housing_median_age    0
total_rooms           0
total_bedrooms        0
population            0
households            0
median_income         0
median_house_value    0
ocean_proximity       0
dtype: int64
```

In [212]:

```
print(housing_raw.nunique())
```

```
longitude               844
latitude                861
housing_median_age       52
total_rooms            5911
total_bedrooms         1923
population             3879
households             1809
median_income         12825
median_house_value     3833
ocean_proximity           5
dtype: int64
```

since there is no one feild that has unique values for the whole dataframe we will create a primary key to enabling merging of datasets.

In [5]:

```
ey = 1
for o, row in housing_raw.iterrows():
    housing_raw.loc[o,'id']=ey
    ey += 1
```

```
C:\Users\benjo\AppData\Local\Temp\ipykernel_4548\1766678878.py:3: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  housing_raw.loc[o,'id']=ey
```

The cell below creates 3 new columns to help us search for outliers

In [6]:

```
housing_raw['population_density'] = housing_raw['population']/housing_raw['households']
housing_raw['bedless'] = housing_raw['population']/housing_raw['total_bedrooms']
housing_raw.describe()
```

```
C:\Users\benjo\AppData\Local\Temp\ipykernel_4548\4209021603.py:1: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  housing_raw['population_density'] = housing_raw['population']/housing_ra
w['households']
C:\Users\benjo\AppData\Local\Temp\ipykernel_4548\4209021603.py:2: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  housing_raw['bedless'] = housing_raw['population']/housing_raw['total_be
drooms']
```

Out[6]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	p
count	20433.000000	20433.000000	20433.000000	20433.000000	20433.000000	2043
mean	-119.570689	35.633221	28.633094	2636.504233	537.870553	142
std	2.003578	2.136348	12.591805	2185.269567	421.385070	113
min	-124.350000	32.540000	1.000000	2.000000	1.000000	
25%	-121.800000	33.930000	18.000000	1450.000000	296.000000	78
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	116
75%	-118.010000	37.720000	37.000000	3143.000000	647.000000	172
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	3568

keeping valid data

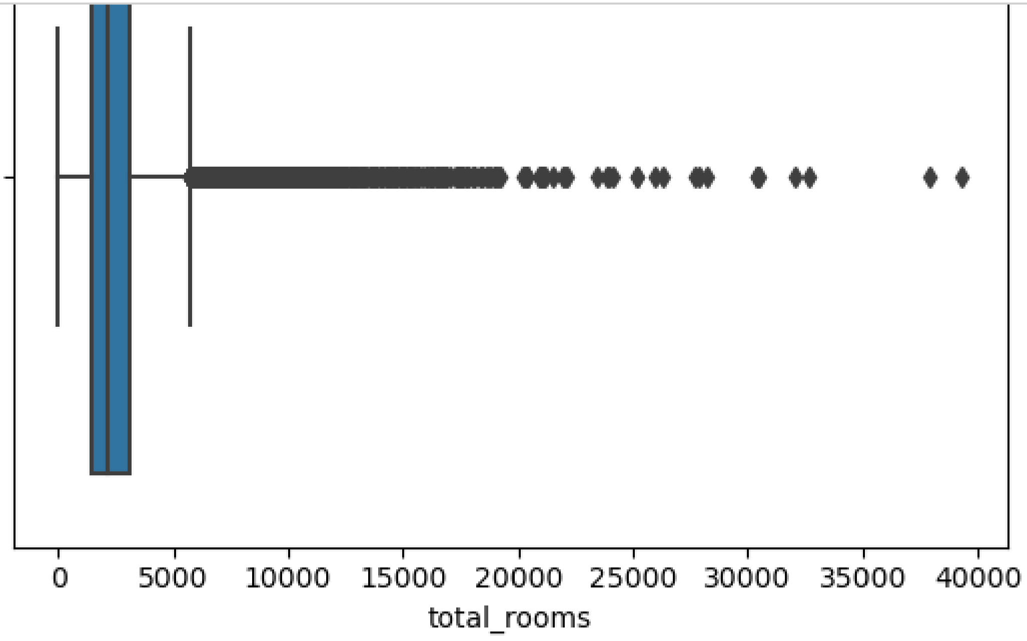
In [ ]:

In [ ]:

Plotting box plots to show outliers

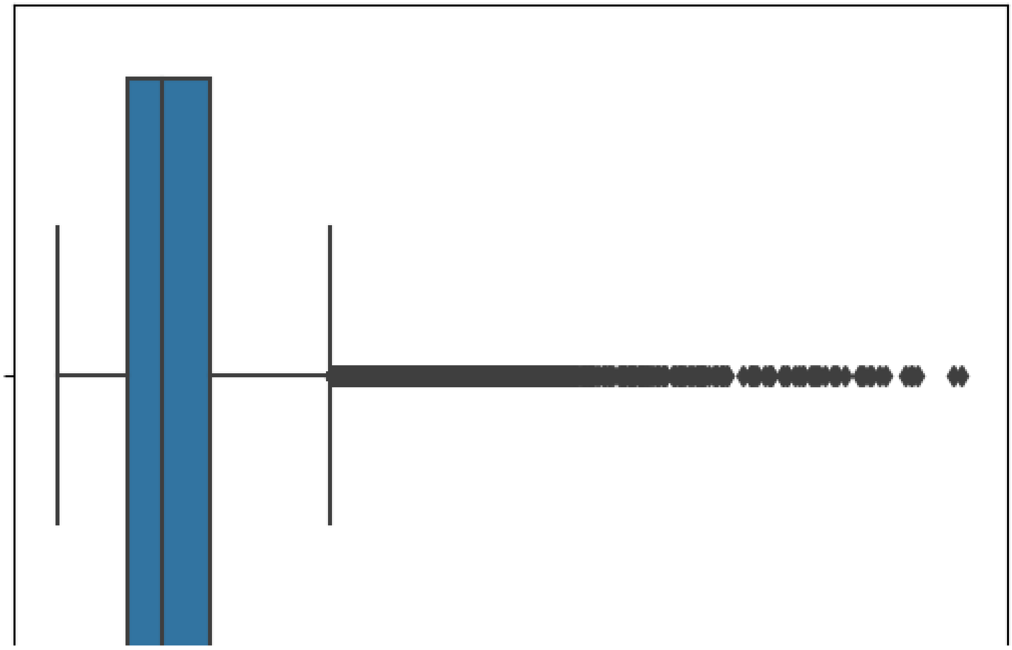
In [215]:

```
sns.boxplot(x=housing_raw['housing_median_age'])
plt.show()
sns.boxplot(x=housing_raw['total_rooms'])
plt.show()
sns.boxplot(x=housing_raw['total_bedrooms'])
plt.show()
sns.boxplot(x=housing_raw['population'])
plt.show()
sns.boxplot(x=housing_raw['households'])
plt.show()
sns.boxplot(x=housing_raw['median_income'])
plt.show()
sns.boxplot(x=housing_raw['median_house_value'])
plt.show()
sns.boxplot(x=housing_raw['population_density'])
plt.show()
sns.boxplot(x=housing_raw['bedless'])
plt.show()
```



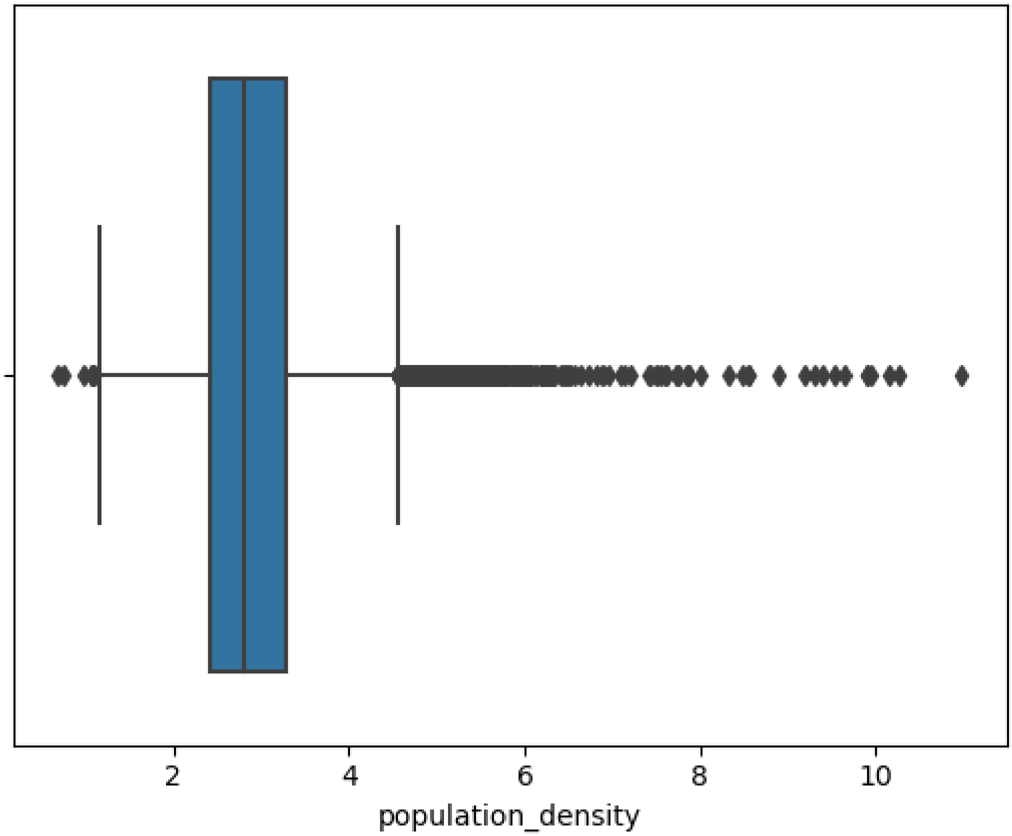
In [7]:

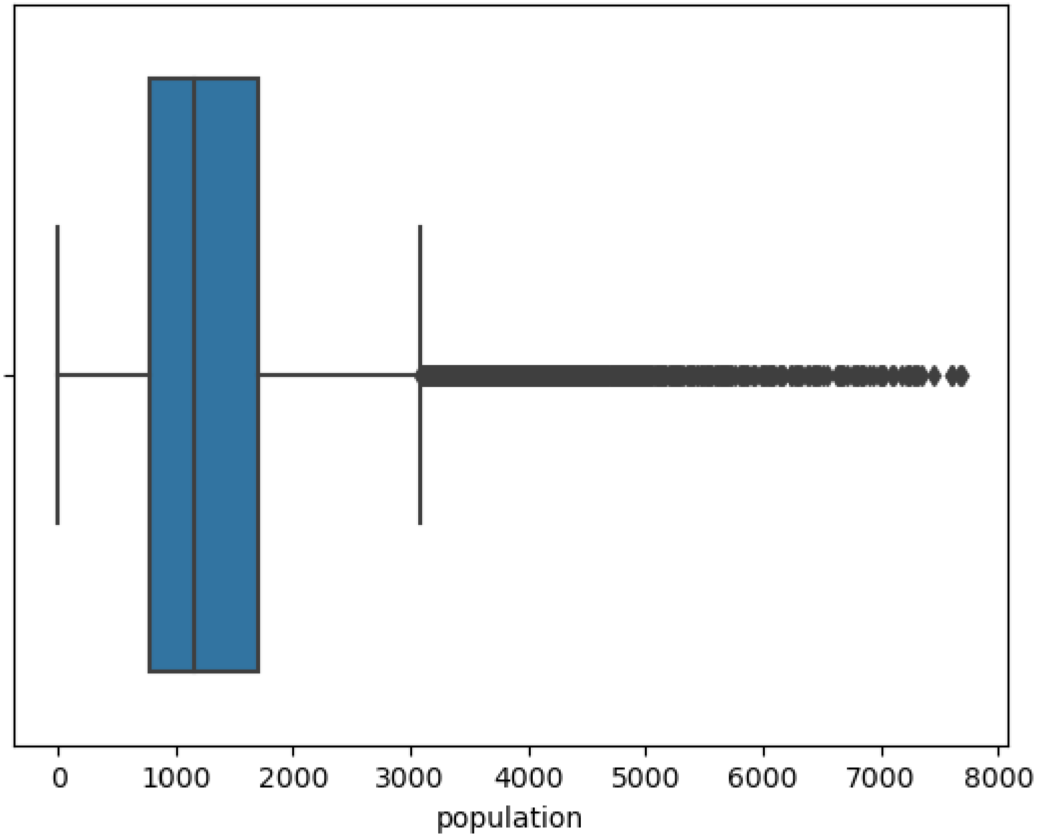
```
#calculate upper fences

wisker_fence_population = np.percentile(housing_raw['population'], 75) + 1.5 * (np.perce
upper_fence_population = wisker_fence_population + 1.5 * wisker_fence_population


wisker_fence_total_bedrooms = np.percentile(housing_raw['total_bedrooms'], 75) + 1.5 * (
upper_fence_total_bedrooms = wisker_fence_total_bedrooms + 1.5 * wisker_fence_total_bedr

wisker_fence_households = np.percentile(housing_raw['households'], 75) + 1.5 * (np.perce
upper_fence_households = wisker_fence_households + 1.5 * wisker_fence_households

wisker_fence_median_income = np.percentile(housing_raw['median_income'], 75) + 1.5 * (np
upper_fence_median_income = wisker_fence_median_income + 1.5 * wisker_fence_median_incom

wisker_fence_median_income = np.percentile(housing_raw['bedless'], 75) + 1.5 * (np.perce
upper_fence_median_income = wisker_fence_median_income + 1.5 * wisker_fence_median_incom

wisker_fence_median_income = np.percentile(housing_raw['population_density'], 75) + 1.5
upper_fence_median_income = wisker_fence_median_income + 1.5 * wisker_fence_median_incom

# Filter outliers
housing = housing_raw[(housing_raw['population'] <= upper_fence_population)]
housing = housing[(housing['total_bedrooms'] <= upper_fence_total_bedrooms)]
housing = housing[(housing['households'] <= upper_fence_households)]
housing = housing[(housing['median_income'] <= upper_fence_median_income)]
housing = housing[(housing['bedless'] <= upper_fence_median_income)]
housing = housing[(housing['population_density'] <= upper_fence_median_income)]



#plot boxs again
sns.boxplot(x=housing['total_rooms'])
plt.show()
sns.boxplot(x=housing['total_bedrooms'])
plt.show()
sns.boxplot(x=housing['population'])
plt.show()
sns.boxplot(x=housing['households'])
plt.show()
sns.boxplot(x=housing['median_income'])
plt.show()
sns.boxplot(x=housing['population_density'])
plt.show()
sns.boxplot(x=housing['bedless'])
plt.show()
```



In [188]:

```
sns.boxplot(x=housing['population_density'])
plt.show()
sns.boxplot(x=housing['population'])
plt.show()
```





In [10]:

```
housing.describe()
```

Out[10]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	p
count	20167.000000	20167.000000	20167.000000	20167.000000	20167.000000	2016
mean	-119.572935	35.637605	28.694055	2563.436406	525.327862	138
std	2.003969	2.138954	12.527329	1846.640076	362.369160	95
min	-124.350000	32.540000	1.000000	2.000000	2.000000	
25%	-121.800000	33.930000	18.000000	1449.500000	297.000000	78
50%	-118.500000	34.260000	29.000000	2120.000000	435.000000	116
75%	-118.010000	37.720000	37.000000	3120.000000	644.000000	17
max	-114.310000	41.950000	52.000000	18634.000000	2885.000000	777

As we can see, significant outliers have been taken out, yet we still remain below our 5% removal limit so we will now focus on cleaning the capped 500000

we found that houses that are on islands are always very expensive, therefore any rows that have 500000 as average house value will be kept we found that houses close to the city are also tend to be very expensive so any rows with houses close to the city will be kept

first we need to know the distance from the city

In [8]:

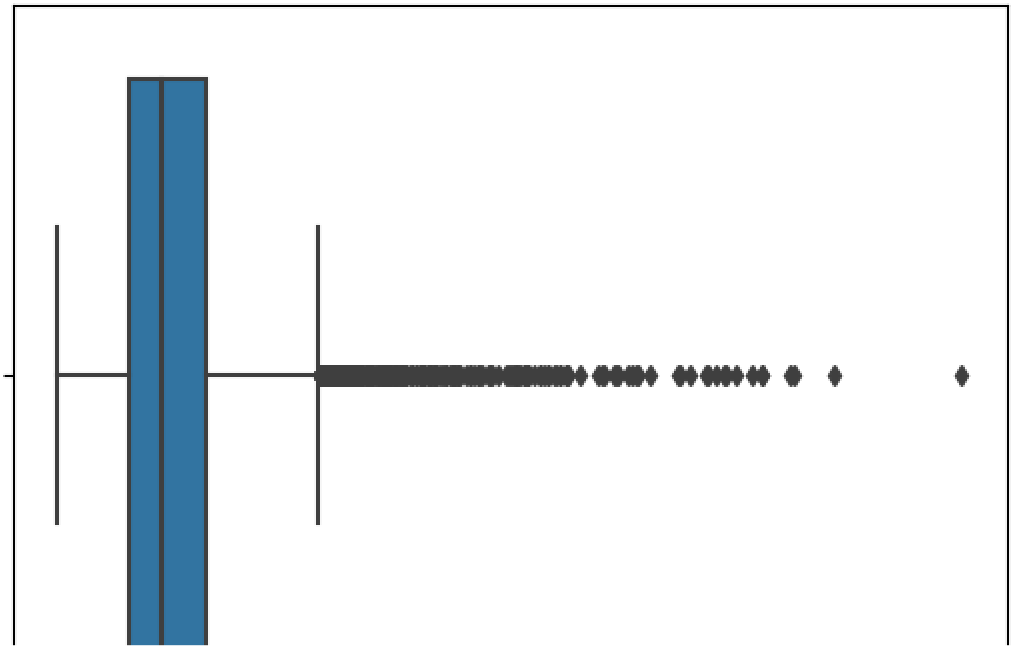
```
import math

housing.loc[:, 'distance_to_closest_city'] = 0

for r, block_row in housing.iterrows(): #for each row
    lati = block_row['latitude'] #create variable to hold latitude
    longi = block_row['longitude'] #create variable to hold latitude
    latcitydif = [] #empty a list hold the differences between the block & citys
    longcitydif = []
    distance = [] #empty a list to distances between block and city's

    for p, city_row in california_cities.iterrows(): #for each row in cities df
        # Subtract the latitude value from each row in the second dataframe
        latdiff = lati - city_row['latitude']
        longdiff = longi - city_row['longitude']
        # Append the resulting series to the list
        latcitydif.append(latdiff)
        longcitydif.append(longdiff)

    for d in range(len(latcitydif)): #for every value in list
        #find the distance using the euclidean distance
        sq = (latcitydif[d])**2 + (longcitydif[d])**2
        dist = math.sqrt(sq)
        #append the distance to distance list
        distance.append(dist)
    #identify the minimum distance in the list
    min_distance = min(distance)
    #append minimum in list to column
    housing.loc[r, 'distance_to_closest_city'] = min_distance

housing.describe()
```

Out[8]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	p
count	20167.000000	20167.000000	20167.000000	20167.000000	20167.000000	2016
mean	-119.572935	35.637605	28.694055	2563.436406	525.327862	138
std	2.003969	2.138954	12.527329	1846.640076	362.369160	95
min	-124.350000	32.540000	1.000000	2.000000	2.000000	
25%	-121.800000	33.930000	18.000000	1449.500000	297.000000	78
50%	-118.500000	34.260000	29.000000	2120.000000	435.000000	116
75%	-118.010000	37.720000	37.000000	3120.000000	644.000000	17
max	-114.310000	41.950000	52.000000	18634.000000	2885.000000	777

In [11]:

```
#create dataframes consisting of island houses and
df_city_houses = housing[(housing['distance_to_closest_city'] <= 0.15)]
df_island_houses = housing[(housing['ocean_proximity'] == 'island')]

safe_rows = pd.concat([df_island_houses, df_city_houses], ignore_index=True, sort=False)
df_island_city = safe_rows.describe()

housing1 = housing[(housing['median_house_value'] <= 500000)]
housing2 = pd.concat([housing1, safe_rows], ignore_index=True, sort=False)
housing = housing2.drop_duplicates()
housing2.describe()

sns.boxplot(x=df_city_houses['total_rooms'])
plt.show()
sns.boxplot(x=df_city_houses['total_bedrooms'])
plt.show()
sns.boxplot(x=df_city_houses['population'])
plt.show()
sns.boxplot(x=df_city_houses['households'])
plt.show()
sns.boxplot(x=df_city_houses['median_income'])
plt.show()
sns.boxplot(x=df_city_houses['population_density'])
plt.show()
sns.boxplot(x=df_city_houses['bedless'])
plt.show()
```



In [12]:

```
housing.describe()
#note, original count was 20433, now it's 19506, 972 rows taken out, approximately 4.5%
```

Out[12]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	p
count	19543.000000	19543.000000	19543.000000	19543.000000	19543.000000	1954
mean	-119.575704	35.657503	28.623343	2548.926828	526.160927	139
std	2.008108	2.148433	12.544141	1829.788661	361.904789	95
min	-124.350000	32.540000	1.000000	2.000000	2.000000	
25%	-121.780000	33.930000	18.000000	1444.000000	298.000000	79
50%	-118.510000	34.270000	29.000000	2111.000000	436.000000	117
75%	-118.000000	37.730000	37.000000	3109.000000	645.000000	172
max	-114.310000	41.950000	52.000000	18634.000000	2885.000000	777

The code below creates a new column with the distance a block is from a major city

In [241]:

Out[241]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	p
count	20258.000000	20258.000000	20258.000000	20258.000000	20258.000000	2025
mean	-119.575046	35.636977	28.754418	2537.309409	520.139599	137
std	2.003614	2.137204	12.523427	1750.309365	351.709278	92
min	-124.350000	32.540000	1.000000	2.000000	2.000000	
25%	-121.800000	33.930000	18.000000	1448.000000	295.000000	78
50%	-118.500000	34.260000	29.000000	2119.000000	433.000000	116
75%	-118.010000	37.720000	37.000000	3114.000000	641.000000	170
max	-114.310000	41.950000	52.000000	14125.000000	2823.000000	769

The cell below creates some new dataframes containing the data in the area of los angeles and san francisco

In [28]:

```
longlostangel = housing[ (housing['longitude'] <= -117) & (housing['longitude'] >= -119)
lostangel = longlostangel[ (longlostangel['latitude'] >= 33) & (longlostangel['latitude'

santa = housing[ (housing['longitude'] <= -121) & (housing['longitude'] >= -123)]
san = santa[ (santa['latitude'] >= 37) & (santa['latitude'] <= 39)]
san.head(10)
```

Out[28]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41	880	129.0	322	
1	-122.22	37.86	21	7099	1106.0	2401	
2	-122.24	37.85	52	1467	190.0	496	
3	-122.25	37.85	52	1274	235.0	558	
4	-122.25	37.85	52	1627	280.0	565	
5	-122.25	37.85	52	919	213.0	413	
6	-122.25	37.84	52	2535	489.0	1094	
7	-122.25	37.84	52	3104	687.0	1157	
8	-122.26	37.84	42	2555	665.0	1206	
9	-122.25	37.84	52	3549	707.0	1551	

Bedlessness

Import shapefile

In [13]:

```
from shapely.geometry import Point, Polygon
import os
import seaborn as sns
import shapely.ops
import netCDF4
from pyproj import Transformer
import matplotlib.patches as mpatches
import matplotlib.lines as mlines
import geopandas as gpd
```

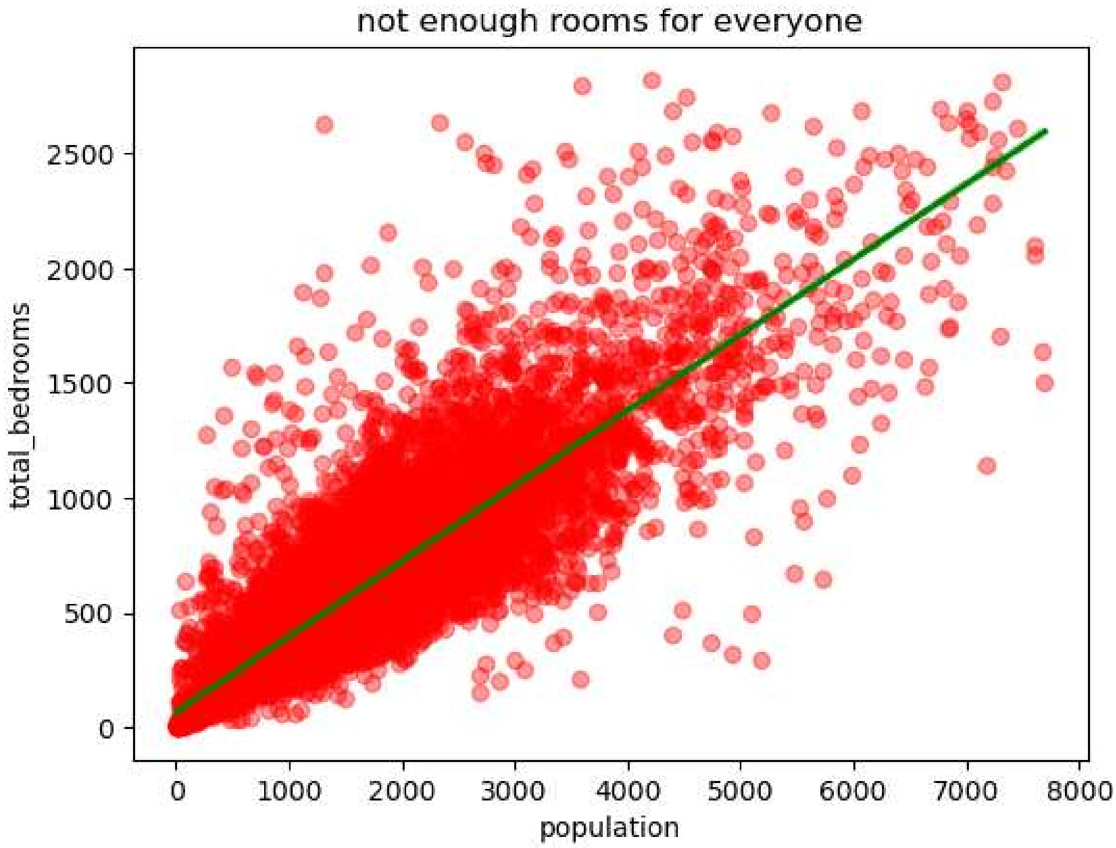
In [20]:

```
os.environ['SHAPE_RESTORE_SHX'] = 'YES' #for some reason I got an error that told me to
#importing our stuff
county_shapefile = gpd.read_file('CA_Counties_TIGER2016.shp') #shape file for Cali map
county_shapefile['geometry'] = county_shapefile['geometry'].apply(lambda geom: shapely.o
```

Regression Plot of population to bedrooms

In [144]:

```
sns.regplot(data=housing, x=housing['population'], y=housing['total_bedrooms'],line_kws=
plt.title('not enough rooms for everyone')
plt.show()
p1 = np.poly1d(np.polyfit(housing['population'], housing['total_bedrooms'], 1))
pv = p1(housing['population'])
r2 = r2_score(housing['total_bedrooms'],pv)
print(p1)
print(r2)



#finding the % of people that have to share a room
```



```
0.3288 x + 65.78
0.7481271644738751
```

In [74]:

```
housing.loc[:, 'alone'] = housing['bedless'] < 2
sharing = housing['alone'].value_counts()[False]
not_sharing = housing['alone'].value_counts()[True]
sharing / (not_sharing + sharing)
```

```
C:\Users\benjo\AppData\Local\Temp\ipykernel_23440\1235303491.py:1: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  housing.loc[:, 'alone'] = housing['bedless'] < 2
```

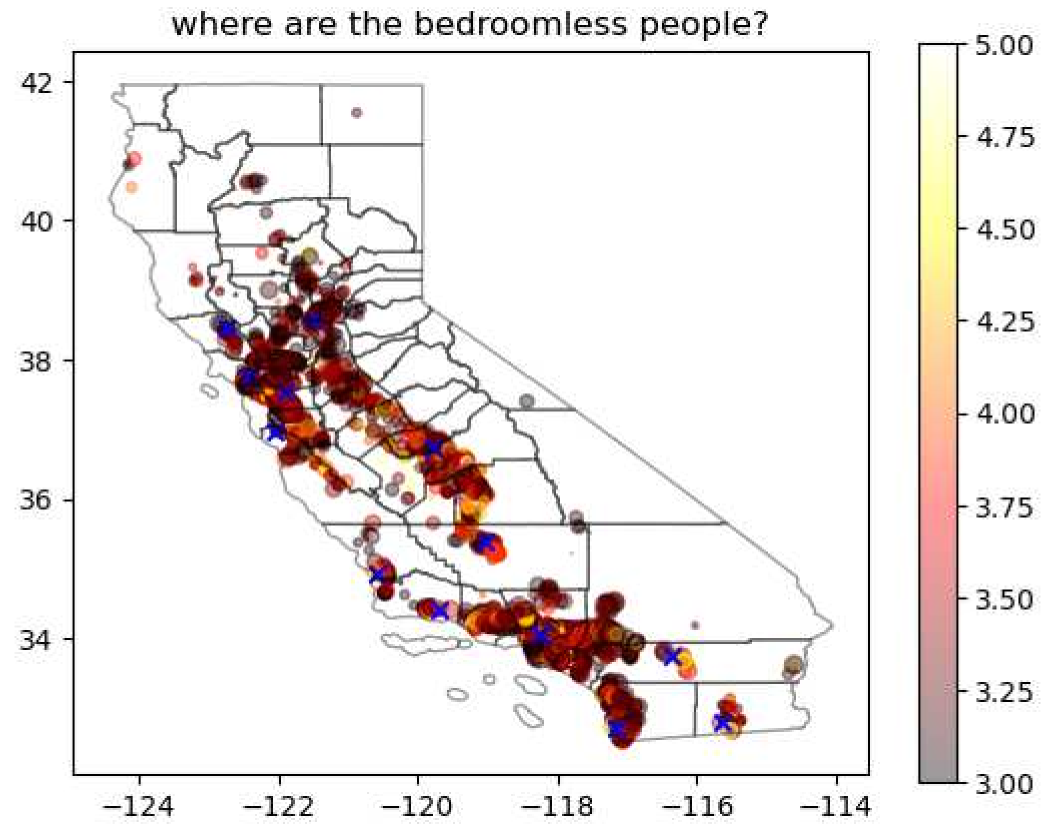
Out[74]:

```
0.8569749015999591
```

On average 70% of Californians either share a bedroom or don't have a bedroom to sleep in. A significant amount of these bedless people reside in the Los Angeles area and the San Francisco area as seen in the map below showing the locations where the population and total number of bedrooms exceeds 4:1. We can also see from the regression plot showing the relationship between distance from major city and homelessness, that bedlessness is more frequent the closer you get to a city

In [220]:

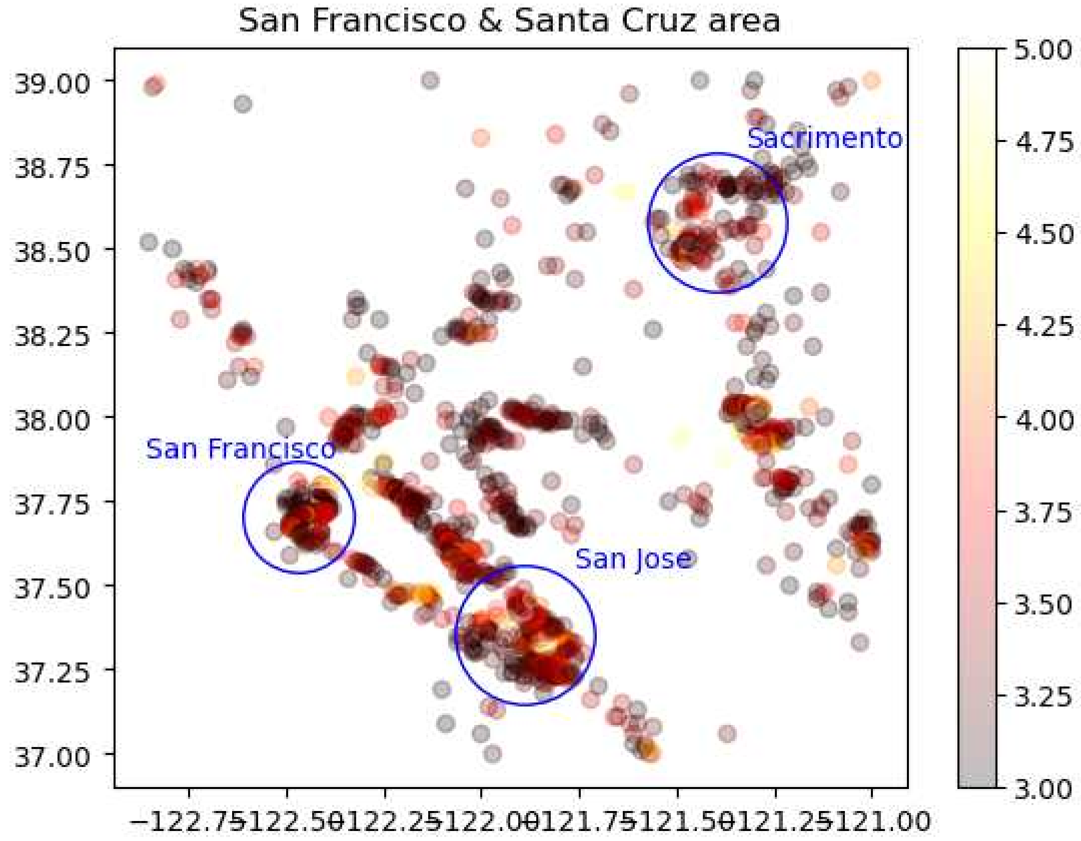
```
calirooms = housing[(housing['population'] >= 3*housing['total_bedrooms'])]
#make sure there are no outliers so that we can visualise better:
upper_fence_rooms_total_bedrooms = np.percentile(calirooms['total_bedrooms'], 75) + 1.5
upper_fence_rooms_population = np.percentile(calirooms['population'], 75) + 1.5 * (np.pe
lower_fence_rooms_total_bedrooms = np.percentile(calirooms['total_bedrooms'], 25) - 1.5
lower_fence_rooms_population = np.percentile(calirooms['population'], 25) - 1.5 * (np.pe
calirooms = calirooms[(calirooms['population'] >= lower_fence_rooms_population) &(caliro

#lets set the coloring
calirooms['coloring'] = 0
for m, row in calirooms.iterrows():
    if row['population'] / row['total_bedrooms'] <= 5:
        calirooms.at[m, 'coloring'] = row['population'] / row['total_bedrooms']
    else:
        calirooms.at[m, 'coloring'] = 5

#now lets plot it with respect to a map
fig, ax = plt.subplots()
plt.scatter(x=calirooms['longitude'],y=calirooms['latitude'],alpha=0.4,s=50*calirooms['h
plt.colorbar()
plt.scatter(x=california_cities['longitude'],y=california_cities['latitude'],alpha = 1,m
#for i in range(california_cities.shape[0]):
#    plt.text(x=california_cities.longitude[i]+0.1,y=california_cities.latitude[i]+0.1,s
county_shapefile.plot(ax=ax, color='None', edgecolor='black',alpha=0.4)
plt.title('where are the bedroomless people?')
plt.show();
```



In [140]:

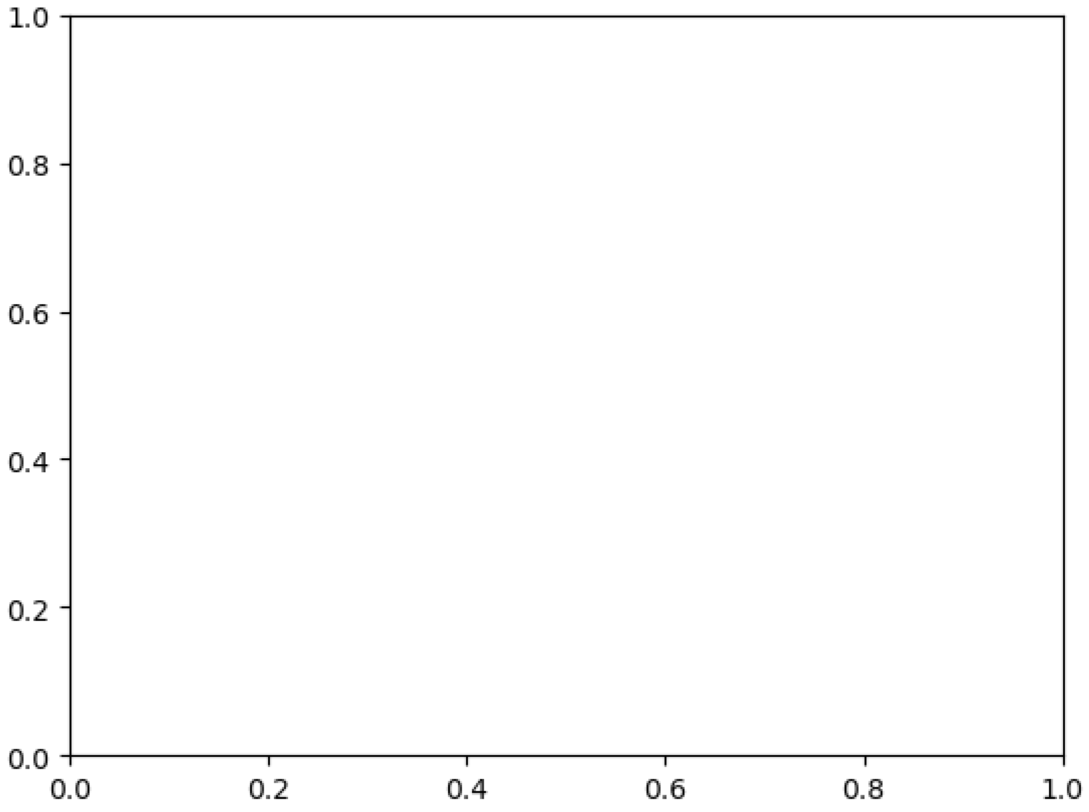
```
calirooms = san[(san['population'] >= 3*san['total_bedrooms'])]

#make sure there are no outliers so that we can visualise better:
#upper_fence_rooms_total_bedrooms = np.percentile(calirooms['total_bedrooms'], 75) + 1.5
#upper_fence_rooms_population = np.percentile(calirooms['population'], 75) + 1.5 * (np.p
#lower_fence_rooms_total_bedrooms = np.percentile(calirooms['total_bedrooms'], 25) - 1.5
#lower_fence_rooms_population = np.percentile(calirooms['population'], 25) - 1.5 * (np.p
#calirooms = calirooms[(calirooms['population'] >= lower_fence_rooms_population) &(calir

#lets set color
calirooms['coloring'] = 0
for m, row in calirooms.iterrows():
    if row['population'] / row['total_bedrooms'] <= 5:
        calirooms.at[m, 'coloring'] = row['population'] / row['total_bedrooms']
    else:
        calirooms.at[m, 'coloring'] = 5

#now lets plot it with respect to a map
fig, ax = plt.subplots()




plt.scatter(x=calirooms['longitude'],y=calirooms['latitude'],alpha=0.25,c=calirooms['col
#plt.scatter(x=calirooms_large['longitude'],y=calirooms_large['latitude'],alpha=0.5,c=ca
plt.colorbar()
plt.title('San Francisco & Santa Cruz area')
#plt.plot(-122.030797,37.090017, marker="o", markersize=25, markeredgecolor="blue", mark
#plt.text(-122.230797,37.154117, 'Santa Cruz',c='blue')
plt.plot(-121.889096, 37.354960, marker="o", markersize=50, markeredgecolor="blue", mark
plt.text(-121.759096, 37.554960, 'San Jose',c='blue')
plt.plot(-122.469417,37.704931, marker="o", markersize=40, markeredgecolor="blue", marke
plt.text(-122.856034,37.883974, 'San Francisco',c='blue')
plt.plot(-121.3944,38.581572, marker="o", markersize=50, markeredgecolor="blue", markerf
plt.text(-121.3244,38.801572, 'Sacrimento',c='blue')


plt.show();
```

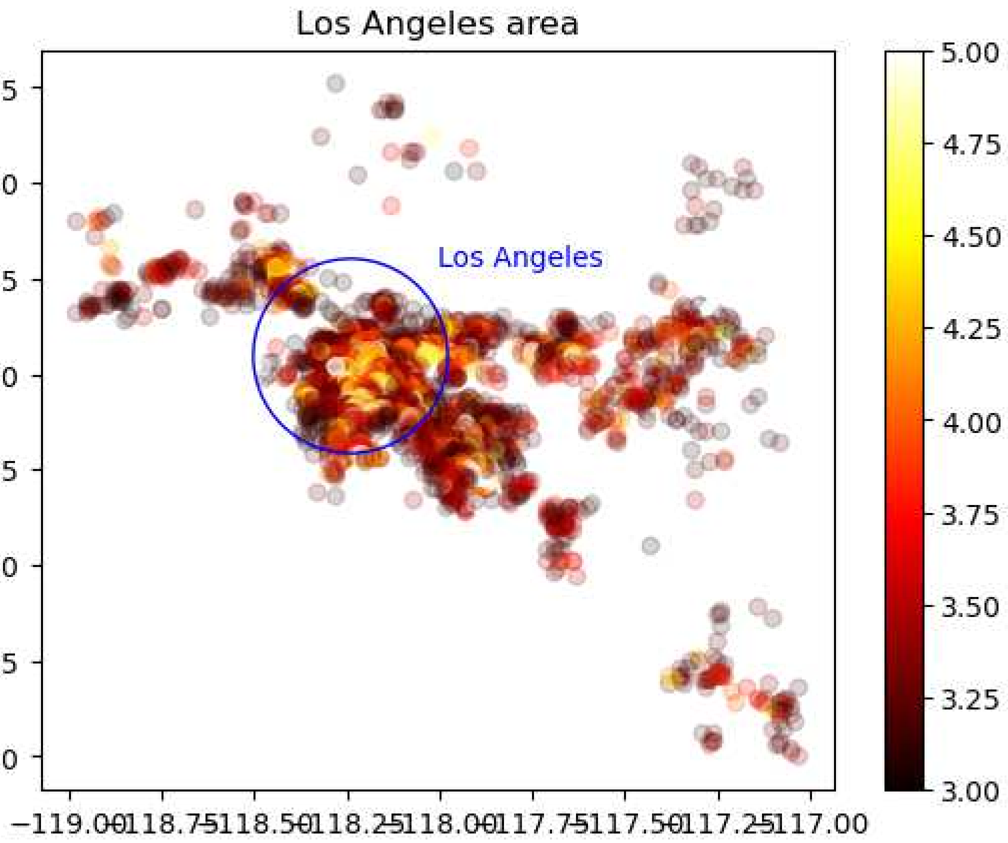
```
C:\Users\benjo\AppData\Local\Temp\ipykernel_23440\273978217.py:11: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  calirooms['coloring'] = 0
C:\Users\benjo\AppData\Local\Temp\ipykernel_23440\273978217.py:14: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  calirooms.at[m, 'coloring'] = row['population'] / row['total_bedrooms']
```



In [222]:

```
calirooms = lostangel[(lostangel['population'] >= 3*lostangel['total_bedrooms'])]
#make sure there are no outliers so that we can visualise better:
upper_fence_rooms_total_bedrooms = np.percentile(calirooms['total_bedrooms'], 75) + 1.5
upper_fence_rooms_population = np.percentile(calirooms['population'], 75) + 1.5 * (np.pe
lower_fence_rooms_total_bedrooms = np.percentile(calirooms['total_bedrooms'], 25) - 1.5
lower_fence_rooms_population = np.percentile(calirooms['population'], 25) - 1.5 * (np.pe
calirooms = calirooms[(calirooms['population'] >= lower_fence_rooms_population) &(caliro
#lets set color
calirooms['coloring'] = 0
for m, row in calirooms.iterrows():
    if row['population'] / row['total_bedrooms'] <= 5:
        calirooms.at[m, 'coloring'] = row['population'] / row['total_bedrooms']
    else:
        calirooms.at[m, 'coloring'] = 5

#now lets plot it with respect to a map
fig, ax = plt.subplots()

#now lets plot it with respect to a map
fig, ax = plt.subplots()
plt.scatter(x=calirooms['longitude'],y=calirooms['latitude'],alpha=calirooms['bedless']/
#plt.scatter(x=calirooms['longitude'],y=calirooms['latitude'],alpha=0.5,c=calirooms['pop
plt.colorbar()
#plt.legend()
plt.title('Los Angeles area')
plt.plot(-118.243686,34.052233, marker="o", markersize=70, markeredgecolor="blue", marke
plt.text(-118.005,34.282233, 'Los Angeles',c='blue')
plt.show();
lostcalirooms = calirooms
```

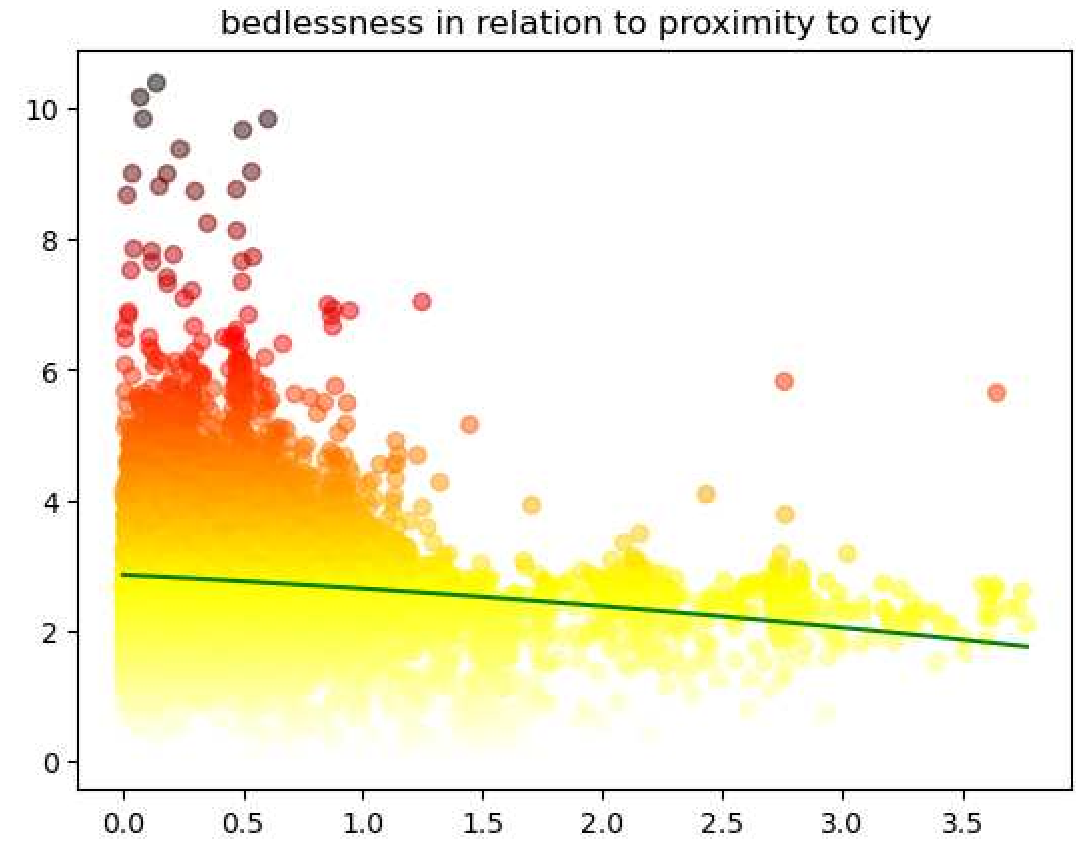



Los Angeles area
34.75
34.50
34.25
34.00
33.75
33.50
33.25
33.00
−119.00
−118.75
−118.50
−118.25
−118.00
−117.75
−117.50
−117.25
−117.00
Los Angeles
5.00
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00

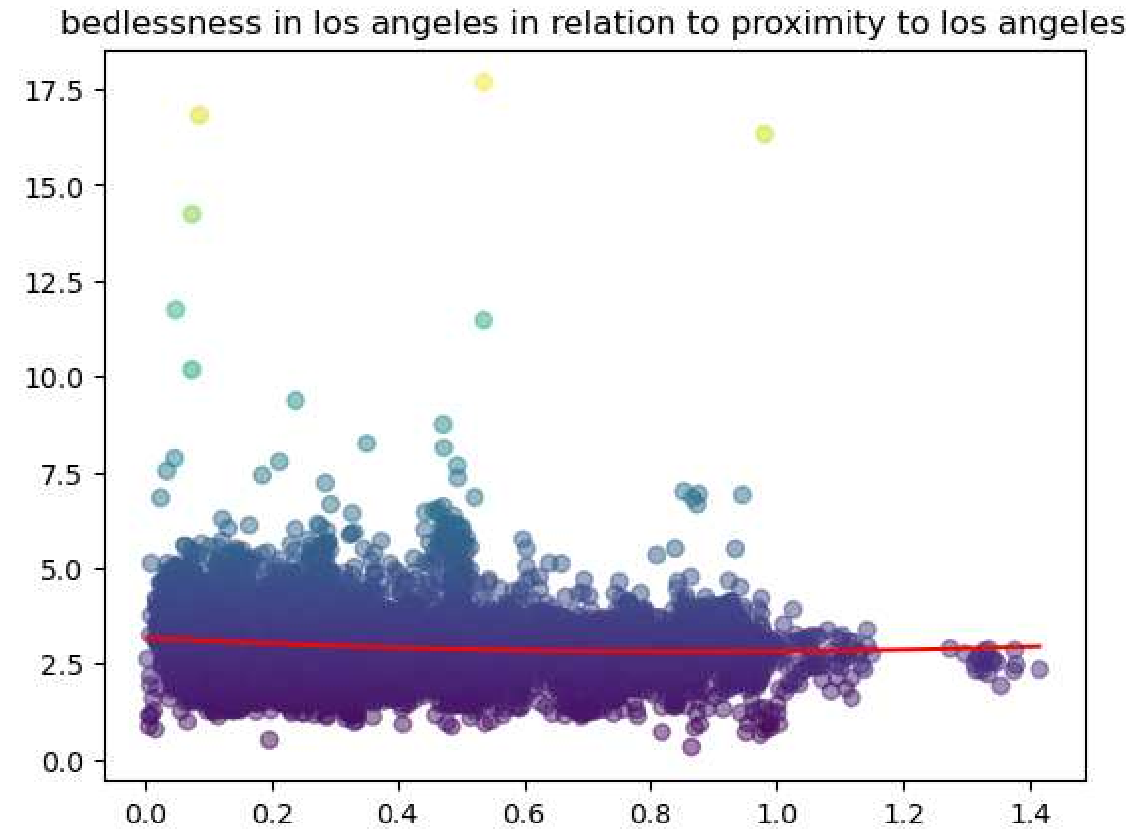
In [202]:

```
plt.title('bedlessness in relation to proximity to city')
p2 = np.poly1d(np.polyfit(housing['distance_to_closest_city'], housing['bedless'], 2))
housefit = np.linspace(housing['distance_to_closest_city'].min(), housing['distance_to_c
plt.scatter(x=housing['distance_to_closest_city'], y=housing['bedless'],alpha=0.5,c=hous
plt.plot(housefit,p2(housefit),label="Average",c='green')
pv = p2(housing['distance_to_closest_city'])
r2 = r2_score(housing['bedless'],pv)
print(r2)
plt.show()


plt.title('bedlessness in los angeles in relation to proximity to los angeles')
p2 = np.poly1d(np.polyfit(lostangel['distance_to_closest_city'], lostangel['bedless'], 2
housefit = np.linspace(lostangel['distance_to_closest_city'].min(), lostangel['distance_
plt.scatter(x=lostangel['distance_to_closest_city'], y=lostangel['bedless'],alpha=0.5,c=
plt.plot(housefit,p2(housefit),label="Average",c='red')
pv = p2(lostangel['distance_to_closest_city'])
r2 = r2_score(lostangel['bedless'],pv)
print(r2)
plt.show()

plt.title('bedlessness in San Fransisco in relation to proximity to San Francisco, San J
p2 = np.poly1d(np.polyfit(san['distance_to_closest_city'], san['bedless'], 2))
housefit = np.linspace(san['distance_to_closest_city'].min(), san['distance_to_closest_c
plt.scatter(x=san['distance_to_closest_city'], y=san['bedless'],alpha=0.5,c=san['bedless
plt.plot(housefit,p2(housefit),label="Average",c='red')
pv = p2(san['distance_to_closest_city'])
r2 = r2_score(san['bedless'],pv)
print(r2)
plt.show()
```

```
0.016053007143745024
```



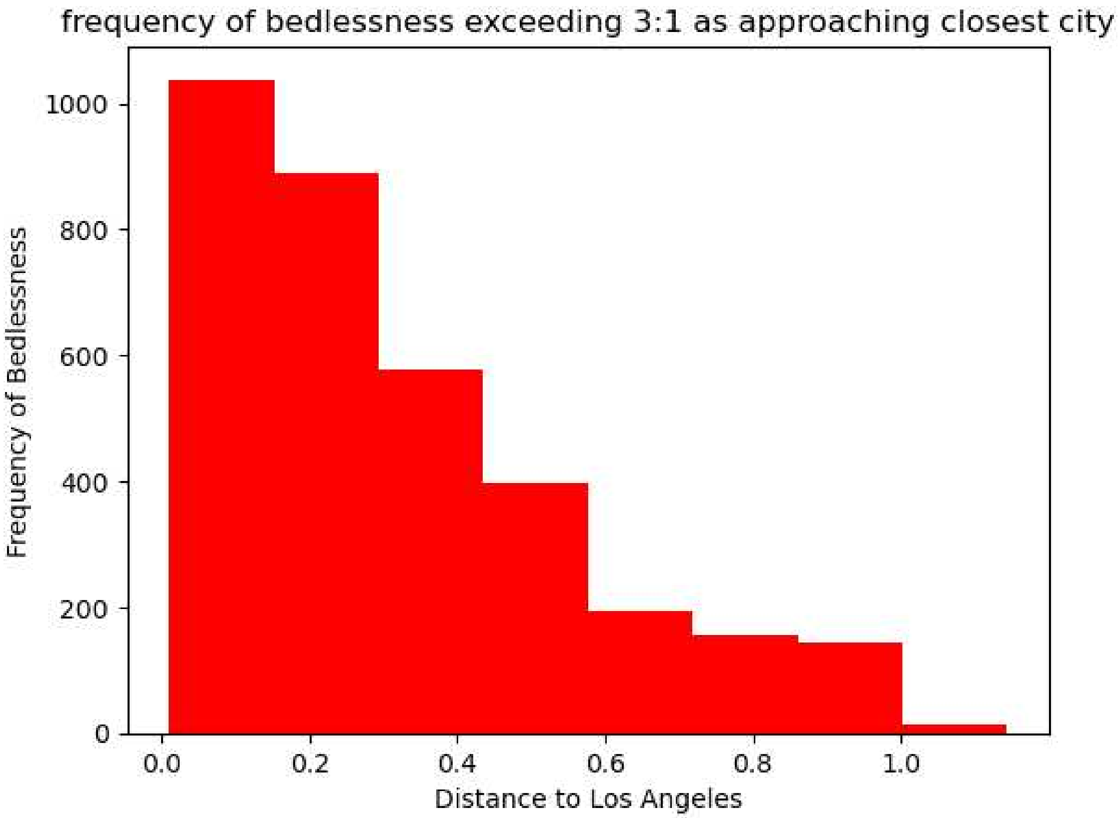
```
0.011681415152787755
```



```
0.012205883191009459
```

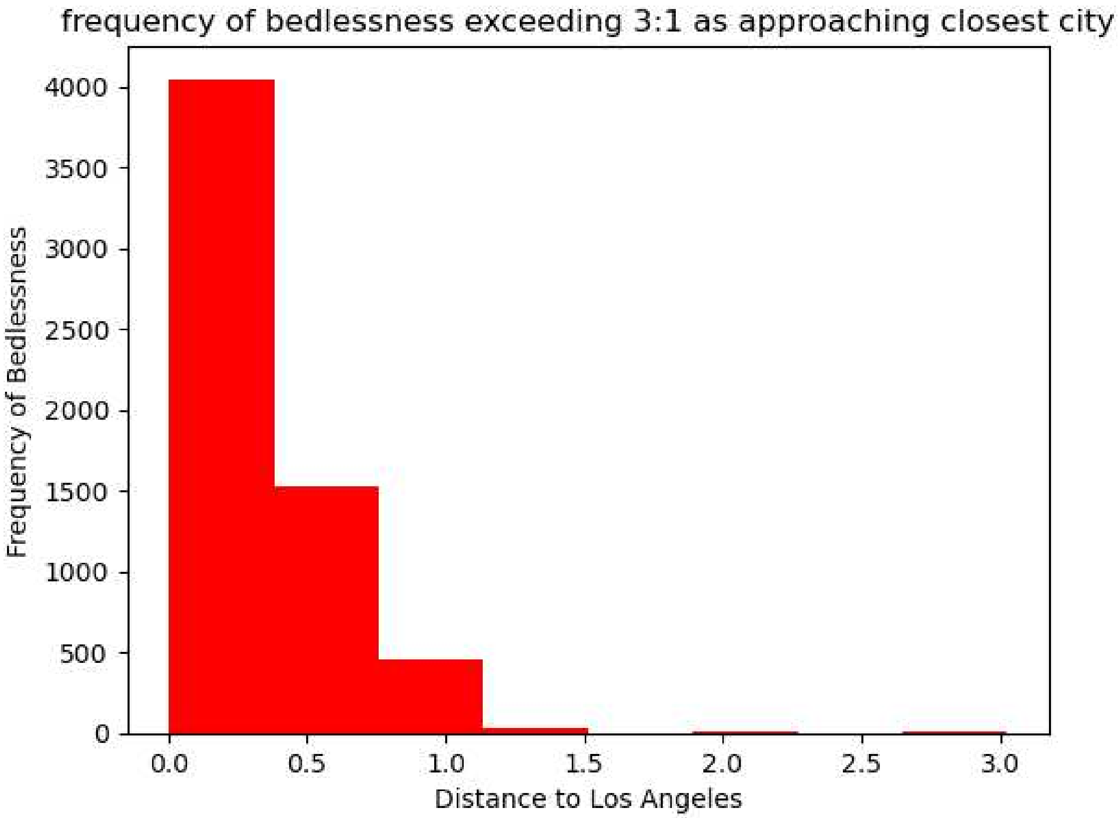
bedlessness in San Fransisco in relation to proximity to San Francisco, San Jose and Santa Cruz

```
lostcalirooms.hist('distance_to_closest_city',bins=8,grid=False,color='red')
plt.title('frequency of bedlessness exceeding 3:1 as approaching closest city')
plt.xlabel('Distance to Los Angeles')
plt.ylabel('Frequency of Bedlessness')
plt.show()
```



In [206]:

```
#plotting frequency

calirooms.hist('distance_to_closest_city',bins=8,grid=False,color='red')
plt.title('frequency of bedlessness exceeding 3:1 as approaching closest city')
plt.xlabel('Distance to Los Angeles')
plt.ylabel('Frequency of Bedlessness')
plt.show()
```



Median income and house value as approaching city

Median Income

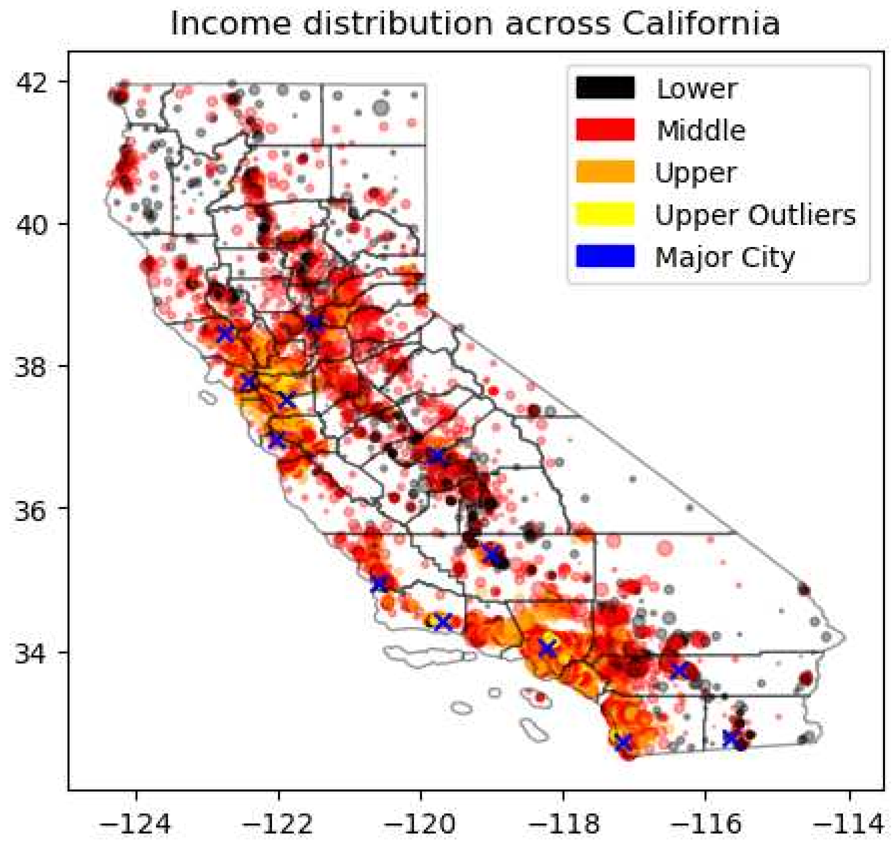
In [21]:

```
caliincomebin = pd.cut(housing['median_income'], 4, precision=2)
calivalubin = pd.cut(housing['median_house_value'], 4, precision=2)

conditions = [
    (housing['median_income'] <= 2.24),
    (housing['median_income'] > 2.24) & (housing['median_income'] <= 4.48),
    (housing['median_income'] > 4.48) & (housing['median_income'] <= 8),
    (housing['median_income'] > 8)
    ]
fence_house_value = np.percentile(housing['median_house_value'], 75) + 1.5 * (np.percent
Vconditions = [
    (housing['median_house_value'] <= np.percentile(housing['median_house_value'], 25)),
    (housing['median_house_value'] > np.percentile(housing['median_house_value'], 25)) &
    (housing['median_house_value'] > np.percentile(housing['median_house_value'], 75)) &
    (housing['median_house_value'] > fence_house_value)
    ]
conditions_age = [
    (housing['housing_median_age'] <= 12),
    (housing['housing_median_age'] > 10) & (housing['housing_median_age'] <= 30),
    (housing['housing_median_age'] > 30) & (housing['housing_median_age'] <= 50),
    (housing['housing_median_age'] > 50)
    ]

status = ['poor', 'middle_class', 'wealthy', 'uber_rich']
housing['status'] = np.select(conditions, status)
house_bins = ['Low_Value', 'Medium_Value', 'High_Value', 'Highest_Value']
housing['house_bin'] = np.select(Vconditions, house_bins)
housing.head()
age_bins = ['<10', '10<30','30<50','50<']
housing['age_bin'] = np.select(conditions_age, age_bins)
```

```
C:\Users\benjo\AppData\Local\Temp\ipykernel_23440\450019214.py:25: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  housing['status'] = np.select(conditions, status)
C:\Users\benjo\AppData\Local\Temp\ipykernel_23440\450019214.py:27: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  housing['house_bin'] = np.select(Vconditions, house_bins)
C:\Users\benjo\AppData\Local\Temp\ipykernel_23440\450019214.py:30: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  housing['age_bin'] = np.select(conditions_age, age_bins)
```

In [173]:

```
#california
fig, ax = plt.subplots()

status_colors = {'poor': 'black', 'middle_class': 'red', 'wealthy': 'orange', 'uber_rich

# Map the categories to colors for each data point
statuscolors = housing['status'].map(status_colors)


#county_shapefile.plot(ax=ax, color='white', edgecolor='black')

plt.scatter(x=housing['longitude'],y=housing['latitude'],alpha = (35+((1.066**(100*(hous

plt.scatter(x=california_cities['longitude'],y=california_cities['latitude'],alpha = 1,m
#for i in range(california_cities.shape[0]):
 #   plt.text(x=california_cities.longitude[i]+0.1,y=california_cities.latitude[i]+0.1,s
county_shapefile.plot(ax=ax, color='None', edgecolor='black',alpha=0.4)

plt.title("Income distribution across California")


outliars = mpatches.Patch(color='yellow', label='Upper Outliers')
poor = mpatches.Patch(color='black', label='Lower')
mid = mpatches.Patch(color='red', label='Middle')
upper = mpatches.Patch(color='orange', label='Upper')
city = mpatches.Patch(color='blue', label='Upper')
plt.legend(handles=[poor,mid,upper,outliars,city])



plt.show();
```

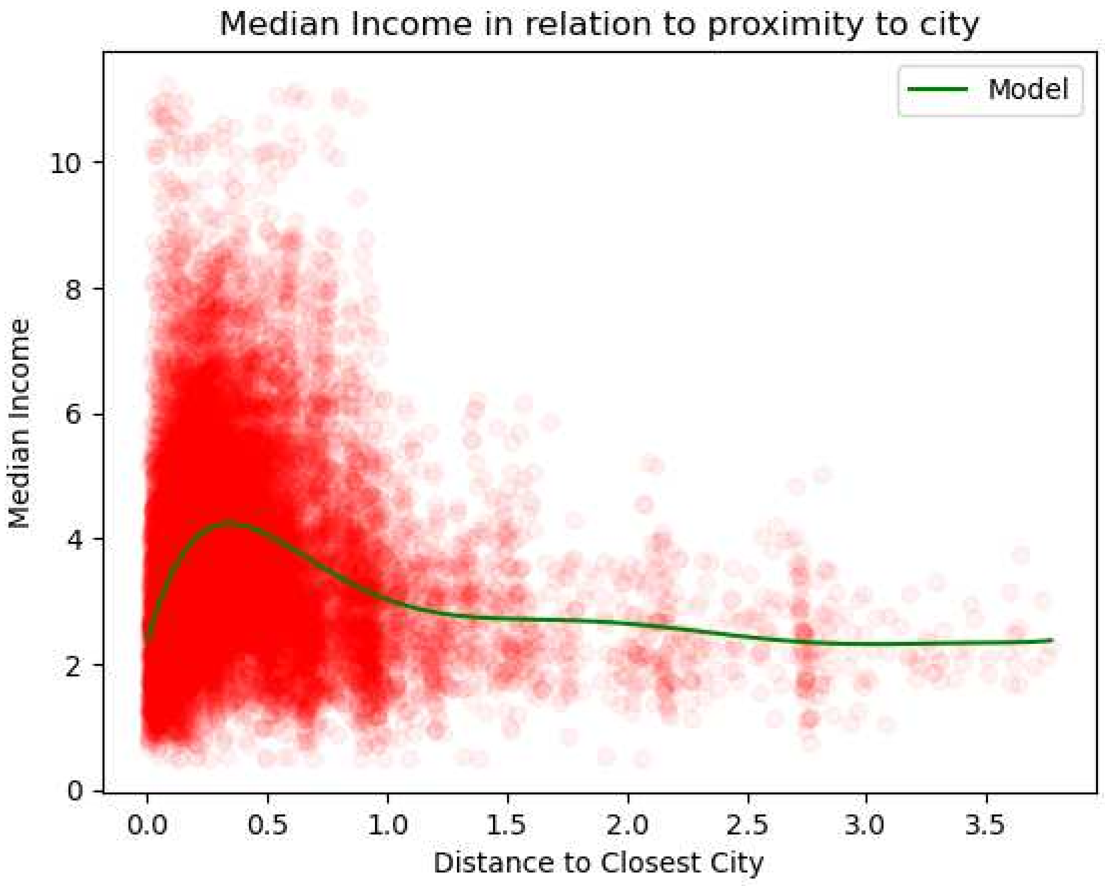

Income distribution across California
Lower
Middle
Upper
Upper Outliers
Major City
42
40
38
36
34
−124
−122
−120
−118
−116
−114

```
In [ ]:
```

In [195]:

```
plt.title('Median Income in relation to proximity to city')
p2 = np.poly1d(np.polyfit(housing['distance_to_closest_city'], housing['median_income'],
housefit = np.linspace(housing['distance_to_closest_city'].min(), housing['distance_to_c
plt.scatter(x=housing['distance_to_closest_city'], y=housing['median_income'], c='red',
plt.plot(housefit, p2(housefit), label="Model", c='green')
print(r2)
plt.legend()
plt.xlabel('Distance to Closest City')
plt.ylabel('Median Income')
plt.show()
```

```
0.012205883191009459
```



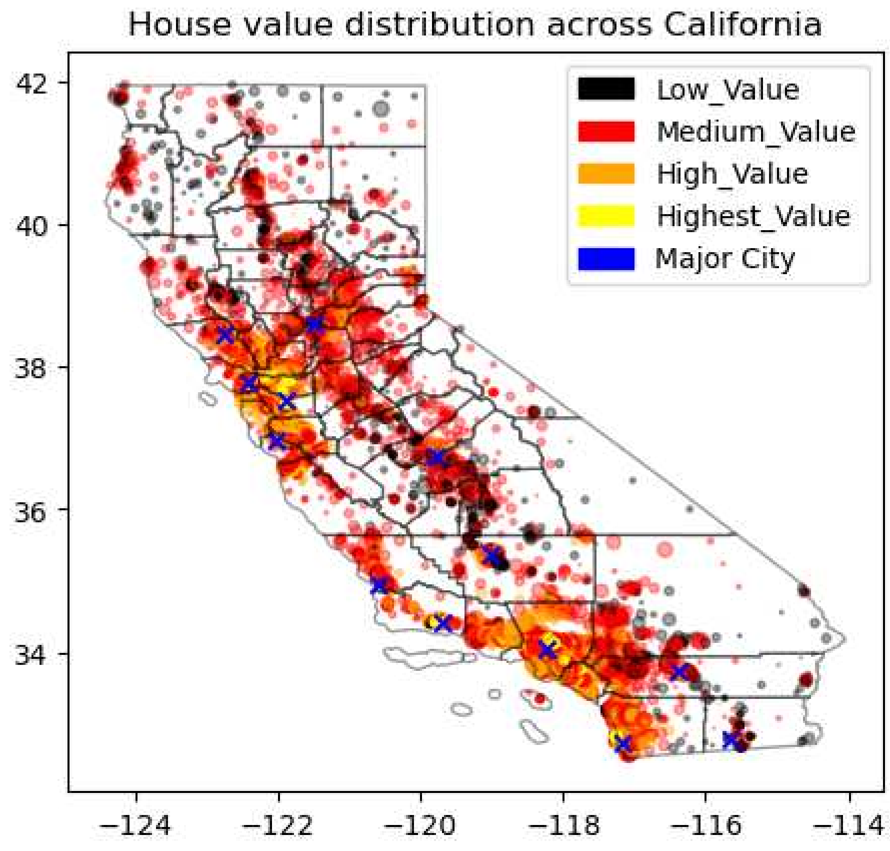
In [ ]:

```
```

Median House price

In [176]:

```
fig, ax = plt.subplots()

house_bin_colors = {'Low_Value': 'orange', 'Medium_Value': 'green', 'High_Value': 'blue'

# Map the categories to colors for each data point
house_bin_colors = housing['house_bin'].map(house_bin_colors)


plt.scatter(x=housing['longitude'],y=housing['latitude'],alpha = (35+((1.066**(100*(hous
county_shapefile.plot(ax=ax, color='None', edgecolor='black',alpha=0.4)

plt.title("House value distribution across California")

plt.scatter(x=california_cities['longitude'],y=california_cities['latitude'],alpha = 1,m


Highest_Value = mpatches.Patch(color='yellow', label='Highest_Value')
Low_Value = mpatches.Patch(color='black', label='Low_Value')
Medium_Value = mpatches.Patch(color='red', label='Medium_Value')
High_Value = mpatches.Patch(color='orange', label='High_Value')
city = mpatches.Patch(color='blue', label='Major City')
plt.legend(handles=[Low_Value,Medium_Value,High_Value,Highest_Value,city])

plt.show();
```



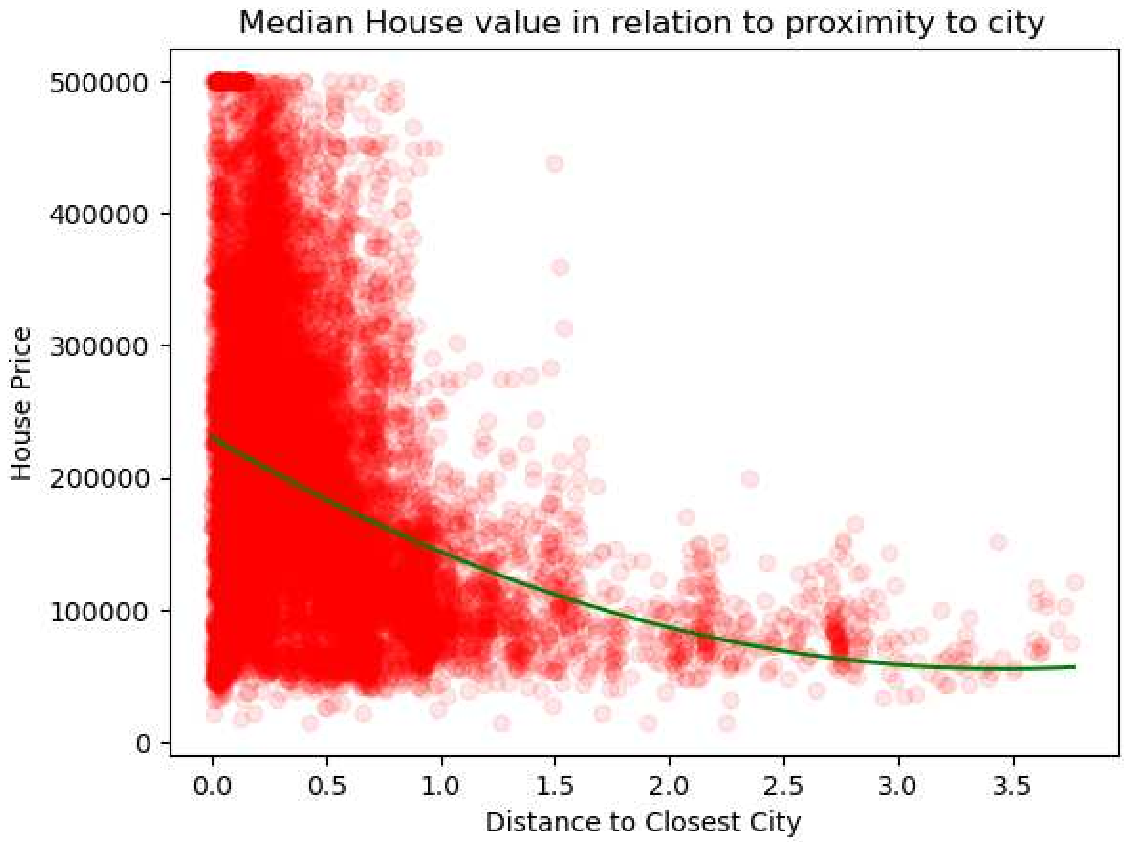
In [ ]:

```

```

In [196]:

```
plt.title('Median House value in relation to proximity to city')
p2 = np.poly1d(np.polyfit(housing['distance_to_closest_city'], housing['median_house_val
housefit = np.linspace(housing['distance_to_closest_city'].min(), housing['distance_to_c
plt.scatter(x=housing['distance_to_closest_city'], y=housing['median_house_value'],c='re
plt.plot(housefit,p2(housefit),label="Average",c='green')
pv = p2(housing['distance_to_closest_city'])
r2 = r2_score(housing['median_house_value'],pv)
print(r2)
plt.xlabel('Distance to Closest City')
plt.ylabel('House Price')
plt.show()
```

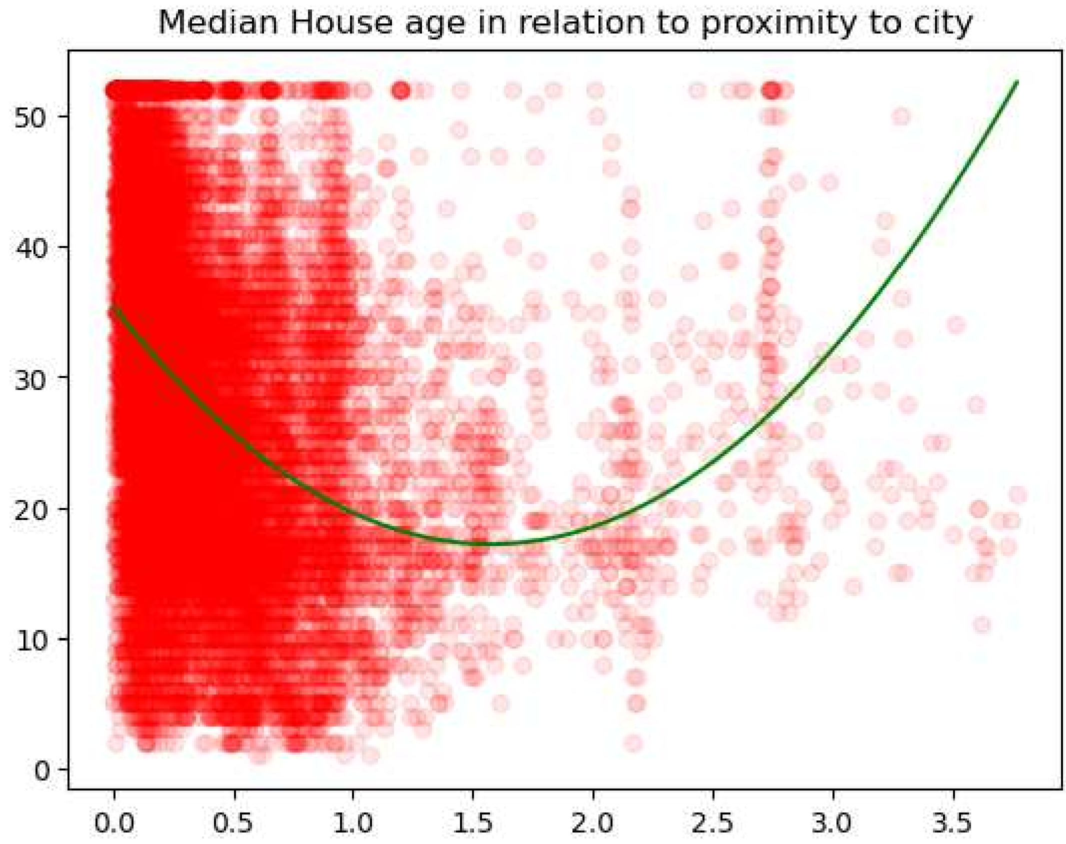
```
0.09070381120347082
```



In [157]:

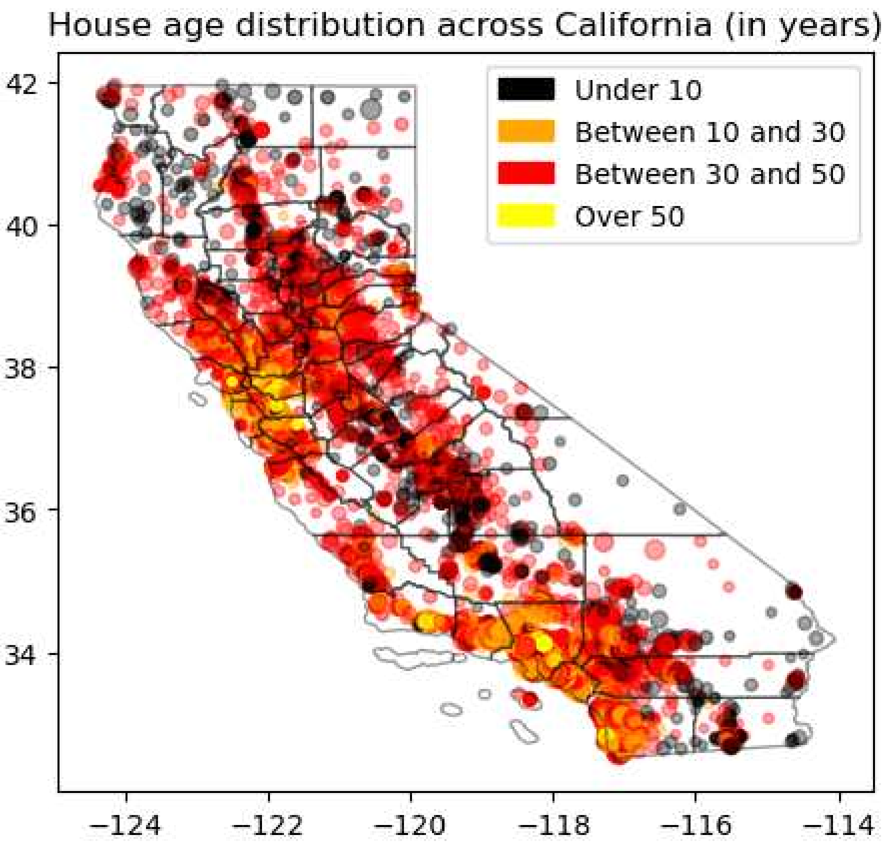
```
plt.title('Median House age in relation to proximity to city')
p2 = np.poly1d(np.polyfit(housing['distance_to_closest_city'], housing['housing_median_a
housefit = np.linspace(housing['distance_to_closest_city'].min(), housing['distance_to_c
plt.scatter(x=housing['distance_to_closest_city'], y=housing['housing_median_age'],c='re
plt.plot(housefit,p2(housefit),label="Average",c='green')
pv = p2(housing['distance_to_closest_city'])
r2 = r2_score(housing['housing_median_age'],pv)
print(r2)
plt.show()
```

```
0.13657061055191722
```



In [152]:

```
fig, ax = plt.subplots()

age_bin_colors = {'<10': 'black', '10<30': 'red', '30<50': 'orange', '50<': 'yellow'}

# Map the categories to colors for each data point
age_bin_colors = housing['age_bin'].map(age_bin_colors)


plt.scatter(x=housing['longitude'],y=housing['latitude'],alpha = (35+((1.066**(100*(hous
county_shapefile.plot(ax=ax, color='None', edgecolor='black',alpha=0.4)

plt.title("House age distribution across California (in years)")


outliars = mpatches.Patch(color='yellow', label='Over 50')
poor = mpatches.Patch(color='black', label='Under 10')
mid = mpatches.Patch(color='orange', label='Between 10 and 30')
upper = mpatches.Patch(color='red', label='Between 30 and 50')
plt.legend(handles=[poor,mid,upper,outliars])



plt.show();
```



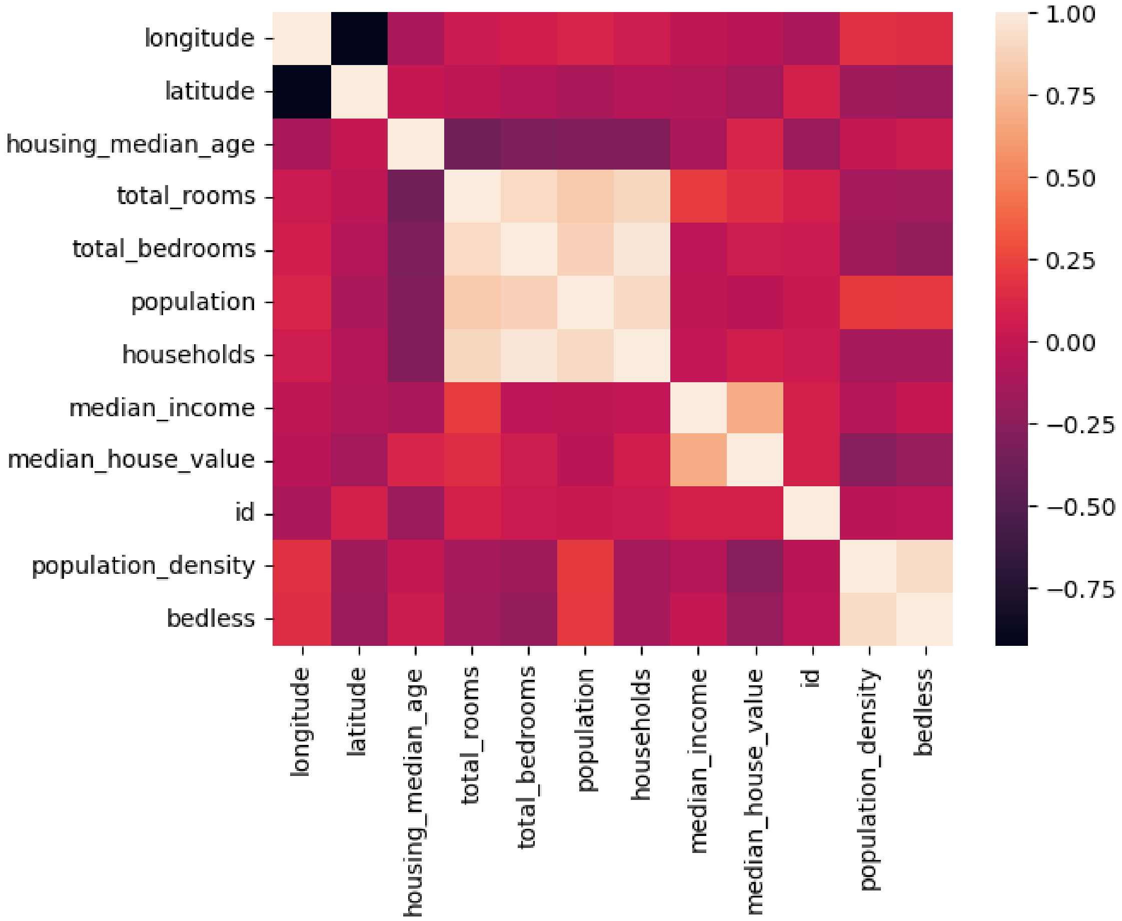
In [ ]:

In [15]:

```
sns.heatmap(housing.corr())
```

Out[15]:

```
<AxesSubplot:>
```

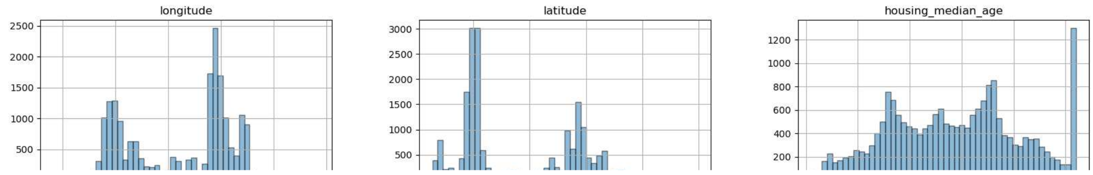


In [16]:

```
housing.hist(alpha=0.5, bins=50,edgecolor='black',figsize=(20,15))
```

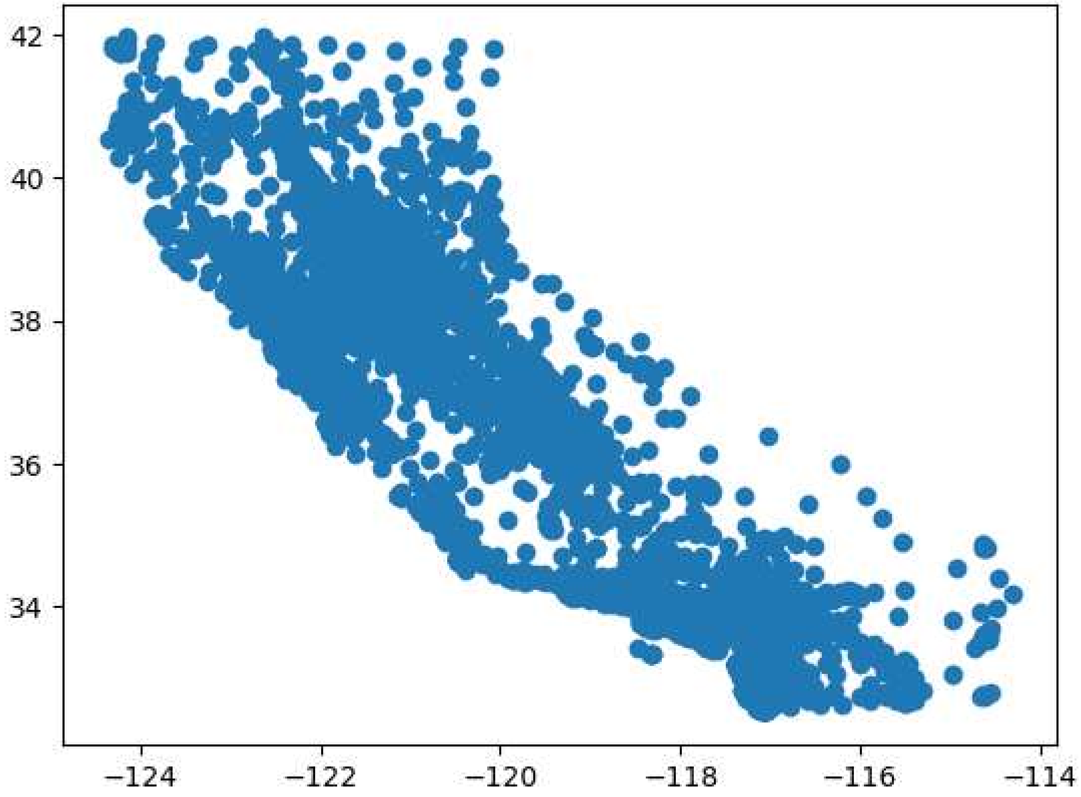
Out[16]:

```
array([[<AxesSubplot:title={'center':'longitude'}>,
        <AxesSubplot:title={'center':'latitude'}>,
        <AxesSubplot:title={'center':'housing_median_age'}>],
       [<AxesSubplot:title={'center':'total_rooms'}>,
        <AxesSubplot:title={'center':'total_bedrooms'}>,
        <AxesSubplot:title={'center':'population'}>],
       [<AxesSubplot:title={'center':'households'}>,
        <AxesSubplot:title={'center':'median_income'}>,
        <AxesSubplot:title={'center':'median_house_value'}>],
       [<AxesSubplot:title={'center':'id'}>,
        <AxesSubplot:title={'center':'population_density'}>,
        <AxesSubplot:title={'center':'bedless'}>]], dtype=object)
```



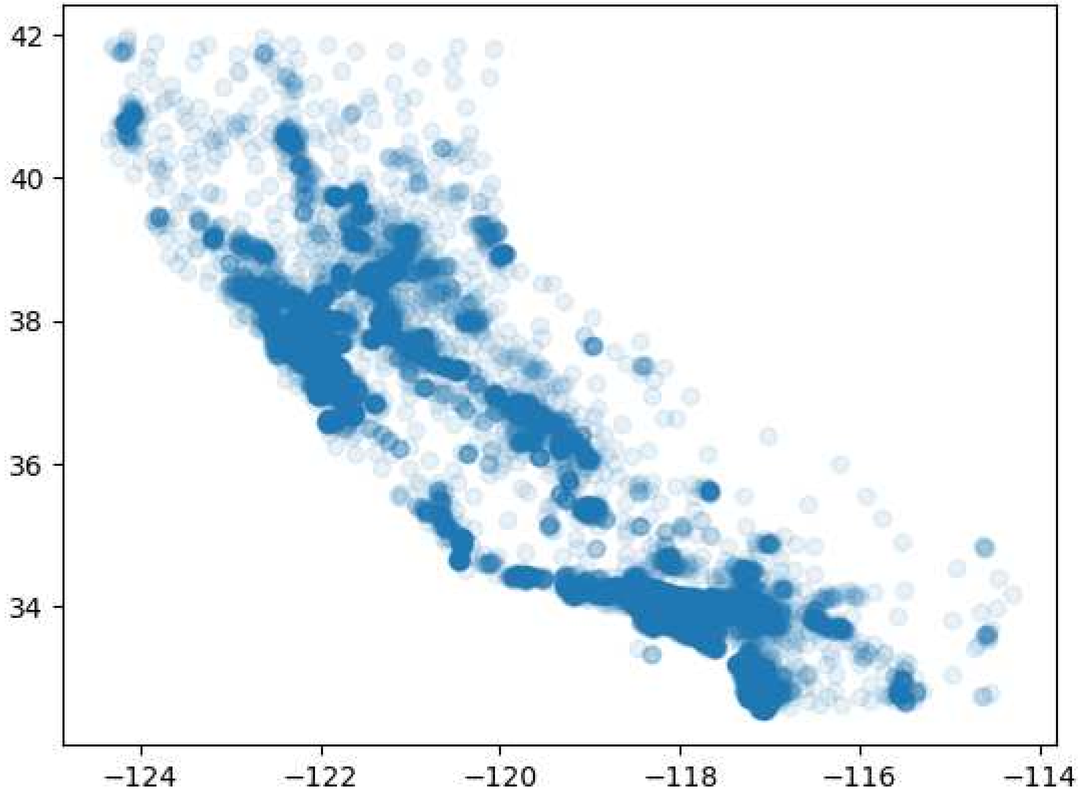
In [19]:

```
plt.scatter(x=housing['longitude'],y=housing['latitude'])
plt.show();
#we can't see claerly
```



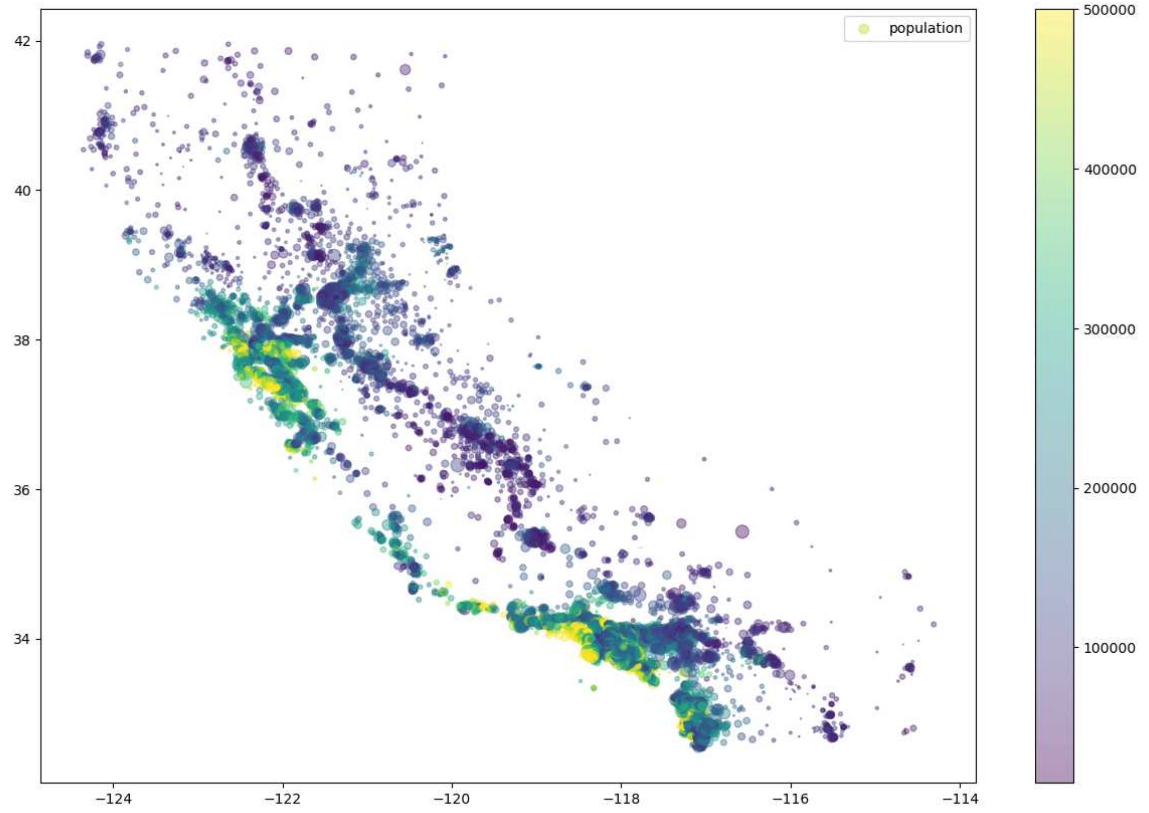
In [20]:

```
plt.scatter(x=housing['longitude'],y=housing['latitude'],alpha=0.1)
plt.show();
#we now can see the density of the blocks
```



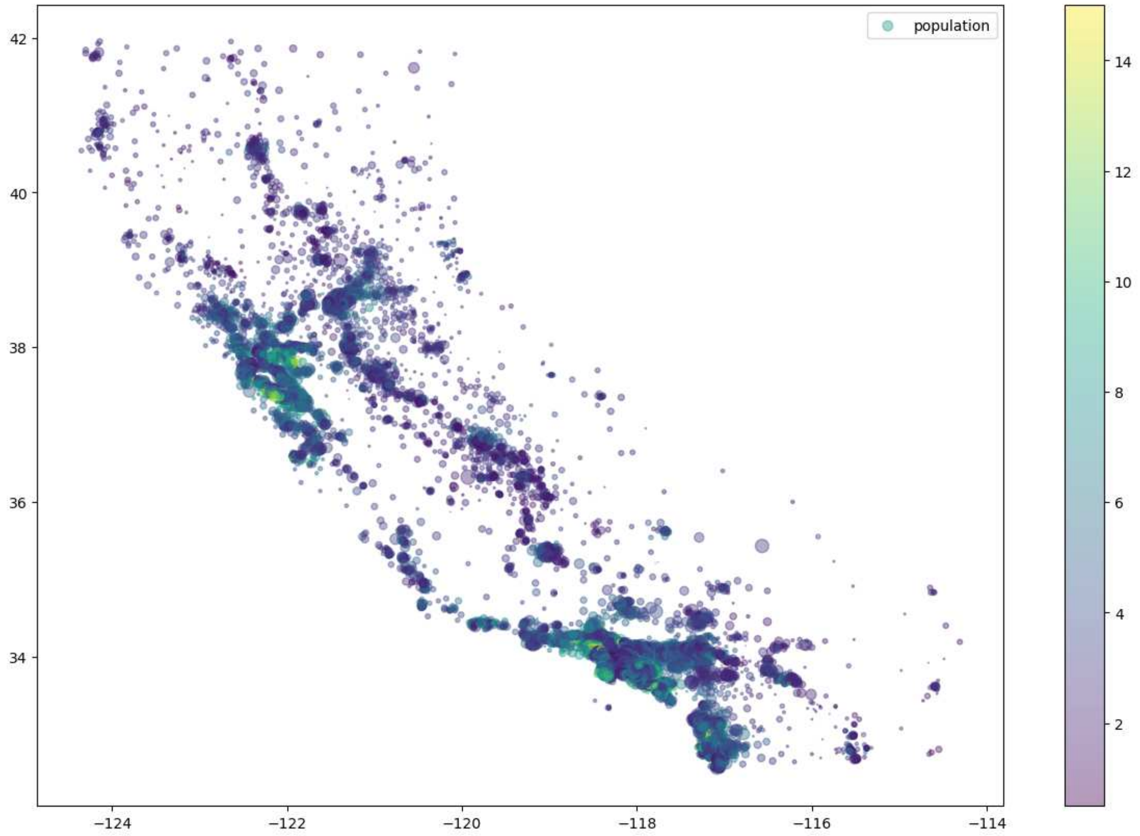
In [21]:

```
plt.figure(figsize=(15,10))

plt.scatter(x=housing['longitude'],y=housing['latitude'],s=housing['population']/80,labe
plt.colorbar()
plt.legend()
plt.show()

#the size of the circle represents the population and the colour of the circle represent
```



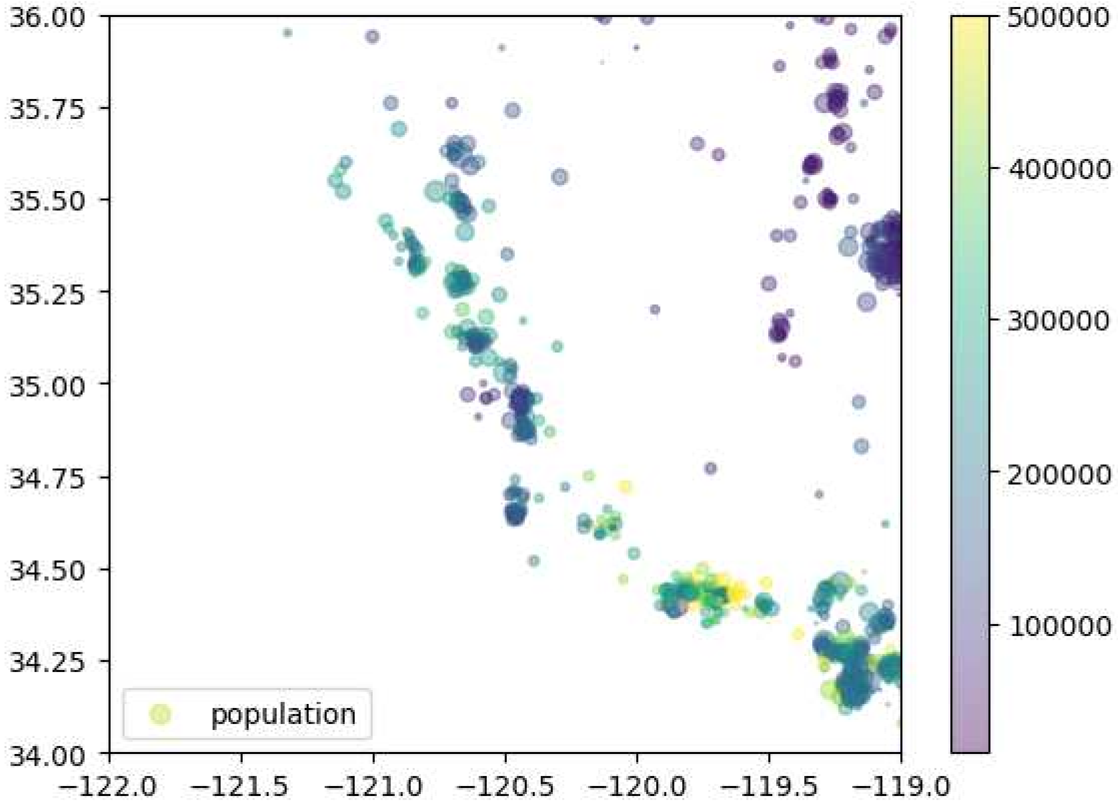
In [22]:

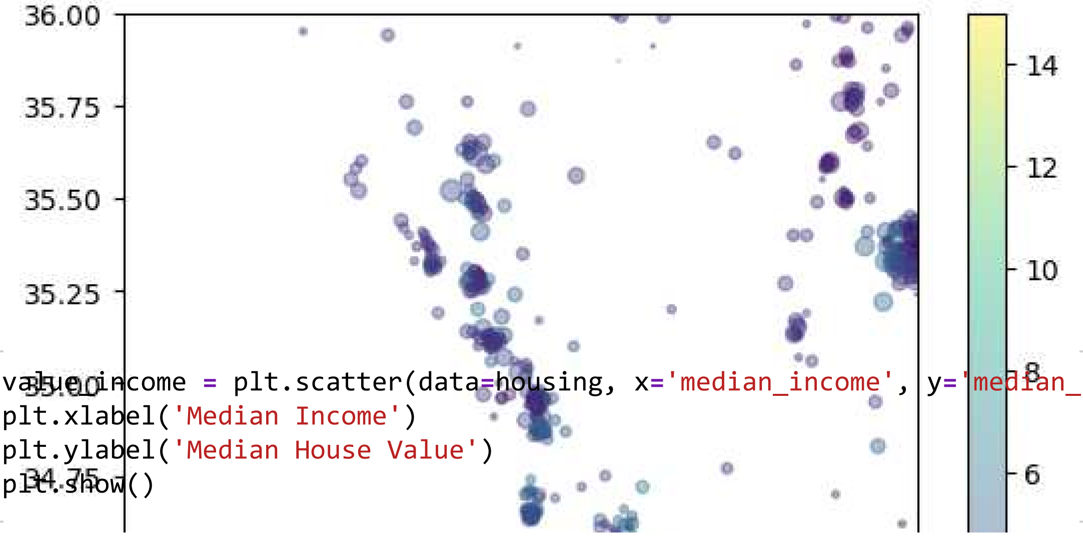
```
plt.figure(figsize=(15,10))

plt.scatter(x=housing['longitude'],y=housing['latitude'],s=housing['population']/80,labe
plt.colorbar()
plt.legend()
plt.show()
```



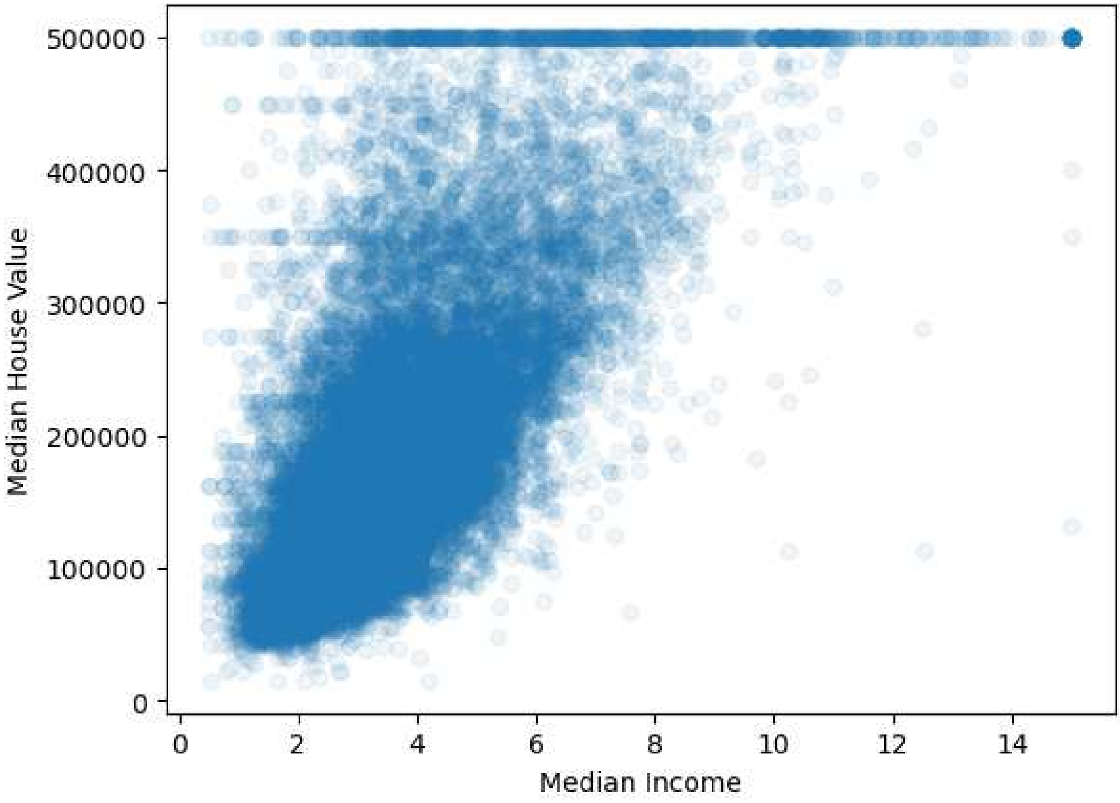
In [24]:

```
plt.scatter(x=housing['longitude'],y=housing['latitude'],s=housing['population']/80,labe
plt.xlim([-122, -119])
plt.ylim([34, 36])
plt.colorbar()
plt.legend()
plt.show()

plt.scatter(x=housing['longitude'],y=housing['latitude'],s=housing['population']/80,labe
plt.xlim([-122, -119])
plt.ylim([34, 36])
plt.colorbar()
plt.legend()
plt.show()
```



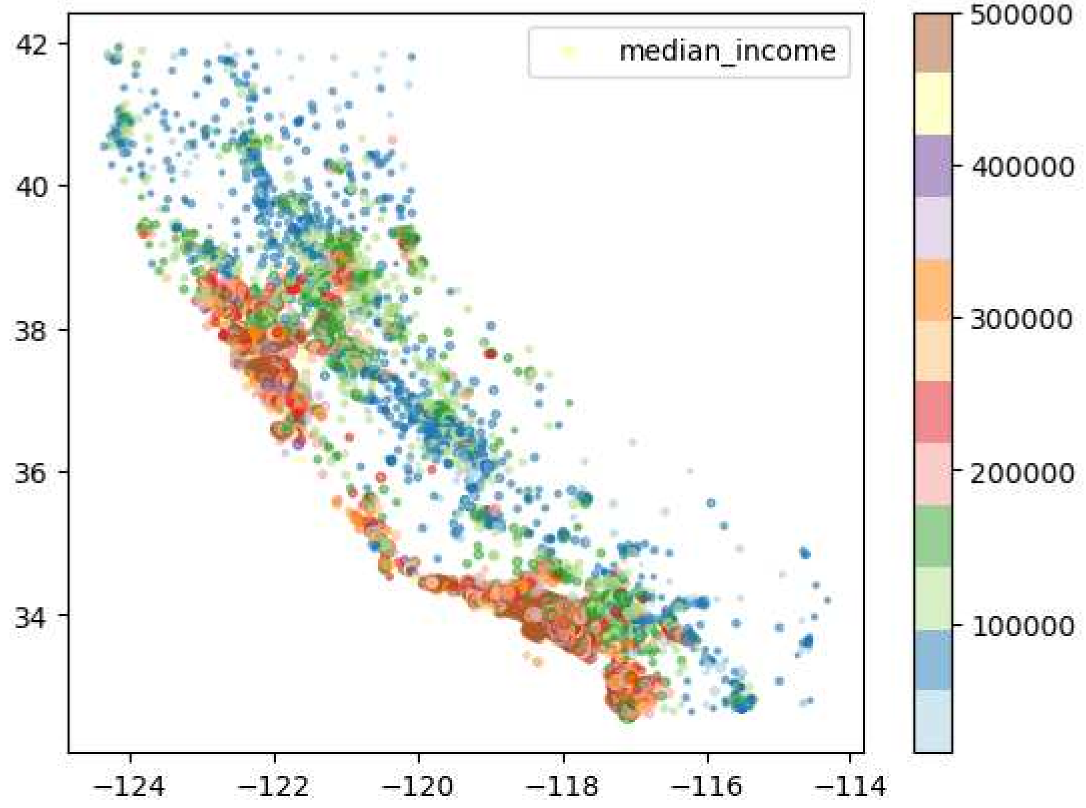


```
value_income = plt.scatter(data=housing, x='median_income', y='median_house_value',alpha
plt.xlabel('Median Income')
plt.ylabel('Median House Value')
plt.show()
```



In [17]:

```
plt.scatter(x=housing['longitude'],y=housing['latitude'],s=housing['median_income']/0.5,
            label='median_income',alpha=0.5,c=housing['median_house_value'],cmap=plt.get
plt.colorbar()
plt.legend()
plt.show()
```



In [18]:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

X = housing[['median_income']]
Y = housing['median_house_value']

linear = LinearRegression(fit_intercept=True)
linear.fit(X, Y)
training_score = linear.score(X, Y)

print("Training score is",np.round(training_score, 3))
print("Correlation score is",np.round(np.sqrt(training_score), 3))
print("Coefficients are",np.round(linear.coef_, 3))
print("Intercept is",np.round(linear.intercept_,3))
```

```
Training score is 0.474
Correlation score is 0.688
Coefficients are [41928.972]
Intercept is 44672.703
```

Population and Households Analysis

In [25]:

```
housing['population'].describe()
```

Out[25]:

```
count    20253.000000
mean      1374.551671
std        920.221690
min          3.000000
25%        785.000000
50%       1160.000000
75%       1703.000000
max       7694.000000
Name: population, dtype: float64
```

In [26]:

```
housing['households'].describe()
```

Out[26]:

```
count    20253.000000
mean       483.661137
std        319.975304
min          2.000000
25%        280.000000
50%        408.000000
75%        598.000000
max       2660.000000
Name: households, dtype: float64
```

In [28]:

```
X=pd.DataFrame(housing['households'])
Y=pd.DataFrame(housing['population'])
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.8,random_state=33)
reg = LinearRegression(fit_intercept=True).fit(X_train,Y_train)
training_score=reg.score(X_train,Y_train)
print('Regression coefficient estimate is',reg.coef_)
print('Regression y-int estimate is',reg.intercept_)
```

```
Regression coefficient estimate is [[2.67036084]]
Regression y-int estimate is [90.68257648]
```

In [30]:

```
training_score = reg.score(X_train,Y_train)
predictions = reg.predict(X_test)
testing_score = r2_score(Y_test,predictions)
print("Training score is",np.round(training_score, 3))
print("Testing score is",np.round(testing_score, 3))
```

```
Training score is 0.824
Testing score is 0.813
```

In [244]:

```
import math

housing.loc[:, 'distance_to_closest_city'] = 0

for r, block_row in housing.iterrows():
    lati = block_row['latitude']
    longi = block_row['longitude']
    latcitydif = []
    longcitydif = []
    distance = []

    for p, city_row in california_cities.iterrows():
        # Subtract the latitude value from each row in the second dataframe
        latdiff = lati - city_row['latitude']
        longdiff = longi - city_row['longitude']
        # Append the resulting series to the list
        latcitydif.append(latdiff)
        longcitydif.append(longdiff)

    for d in range(len(latcitydif)):
        sq = (latcitydif[d])**2 + (longcitydif[d])**2
        dist = math.sqrt(sq)
        distance.append(dist)

    min_distance = min(distance)
    housing.loc[r, 'distance_to_closest_city'] = min_distance

housing.describe()
```

Out[244]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	p
count	20253.000000	20253.000000	20253.000000	20253.000000	20253.000000	2025
mean	-119.575193	35.636999	28.755098	2537.761072	520.231966	137
std	2.003462	2.137220	12.524251	1750.246360	351.694421	92
min	-124.350000	32.540000	1.000000	2.000000	2.000000	
25%	-121.800000	33.930000	18.000000	1448.000000	296.000000	78
50%	-118.500000	34.260000	29.000000	2119.000000	433.000000	116
75%	-118.010000	37.720000	37.000000	3114.000000	641.000000	170
max	-114.310000	41.950000	52.000000	14125.000000	2823.000000	769

In [229]:

```
#create dataframe conataining highest income values
#create dataframe containing highest house value values
#inner merge both dataframes to a new dataframe, at end of outlier removal, merge datafr


df_city_houses = housing[(housing['distance_to_closest_city'] <= 0.15)]
df_island_houses = housing[(housing['ocean_proximity'] == 'island')]


safe_rows = pd.concat([df_island_houses, df_city_houses], ignore_index=True, sort=False)
df_island_city = safe_rows.describe()

housing1 = housing[(housing['median_house_value'] <= 500000)]
housing2 = pd.concat([housing1, safe_rows], ignore_index=True, sort=False)
housing = housing2.drop_duplicates()
housing2.describe()
```

Out[229]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	p
count	24244.000000	24244.000000	24244.000000	24244.000000	24244.000000	2424
mean	-119.545886	35.589427	29.720714	2483.257672	523.736883	138
std	2.002870	2.132375	12.739749	1698.486030	349.722415	90
min	-124.350000	32.540000	1.000000	2.000000	2.000000	
25%	-121.740000	33.950000	19.000000	1436.000000	301.000000	80
50%	-118.460000	34.240000	30.000000	2079.500000	437.000000	116
75%	-118.060000	37.720000	39.000000	3042.000000	643.000000	171
max	-114.310000	41.950000	52.000000	14125.000000	2823.000000	767

In [ ]:

The code below creates a new column with the distance a block is from a major city

In [241]:

```

```

Out[241]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	p
count	20258.000000	20258.000000	20258.000000	20258.000000	20258.000000	2025
mean	-119.575046	35.636977	28.754418	2537.309409	520.139599	137
std	2.003614	2.137204	12.523427	1750.309365	351.709278	92
min	-124.350000	32.540000	1.000000	2.000000	2.000000	
25%	-121.800000	33.930000	18.000000	1448.000000	295.000000	78
50%	-118.500000	34.260000	29.000000	2119.000000	433.000000	116
75%	-118.010000	37.720000	37.000000	3114.000000	641.000000	170
max	-114.310000	41.950000	52.000000	14125.000000	2823.000000	769

The cell below creates some new dataframes containing the data in the area of los angeles and san francisco

In [221]:

```
longlostangel = housing[ (housing['longitude'] <= -117) & (housing['longitude'] >= -119)
lostangel = longlostangel[ (longlostangel['latitude'] >= 33) & (longlostangel['latitude'

santa = housing[ (housing['longitude'] <= -121) & (housing['longitude'] >= -123)]
san = santa[ (santa['latitude'] >= 37) & (santa['latitude'] <= 39)]
san.head(10)
```

Out[221]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41	880	129.0	322	
1	-122.22	37.86	21	7099	1106.0	2401	
2	-122.24	37.85	52	1467	190.0	496	
3	-122.25	37.85	52	1274	235.0	558	
4	-122.25	37.85	52	1627	280.0	565	
5	-122.25	37.85	52	919	213.0	413	
6	-122.25	37.84	52	2535	489.0	1094	
7	-122.25	37.84	52	3104	687.0	1157	
8	-122.26	37.84	42	2555	665.0	1206	
9	-122.25	37.84	52	3549	707.0	1551	

Bedlessness

Import shapefile

In [234]:

```
from shapely.geometry import Point, Polygon
import os
import seaborn as sns
import shapely.ops
import netCDF4
from pyproj import Transformer
import matplotlib.patches as mpatches
import matplotlib.lines as mlines
import geopandas as gpd
```

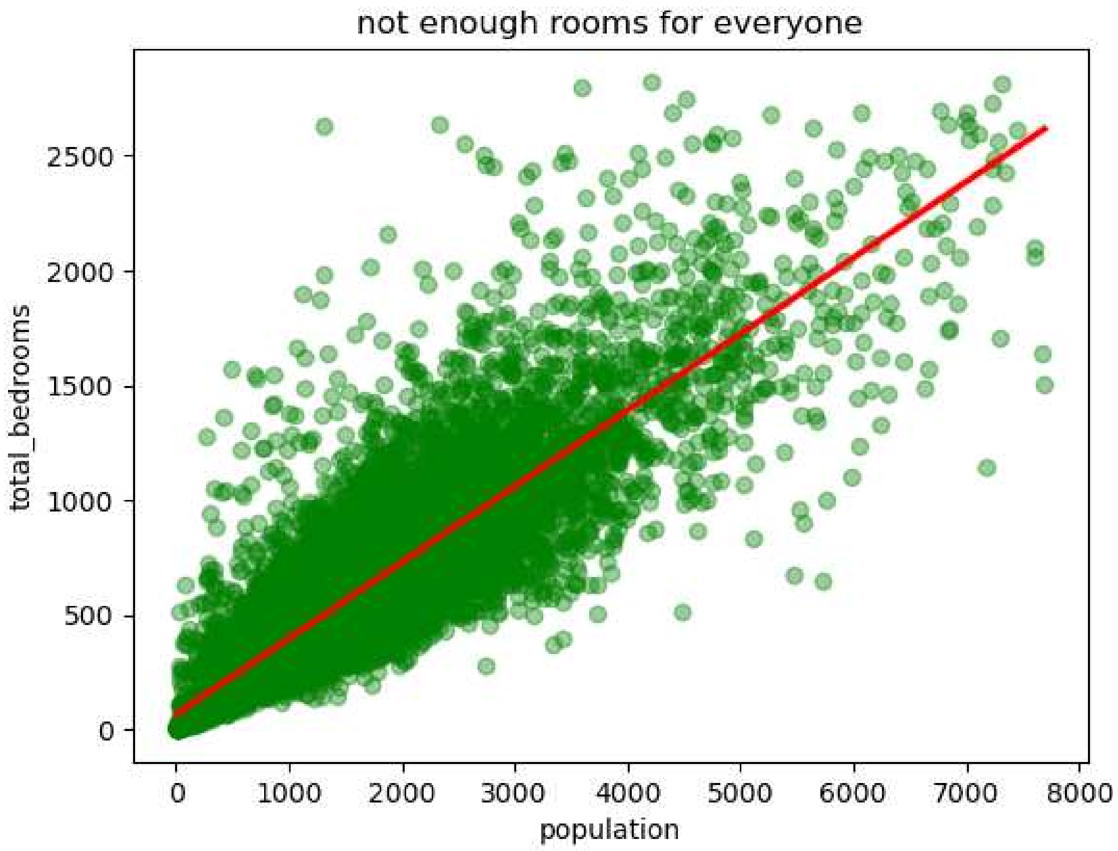
In [235]:

```
os.environ['SHAPE_RESTORE_SHX'] = 'YES' #for some reason I got an error that told me to
#importing our stuff
county_shapefile = gpd.read_file('CA_Counties_TIGER2016.shp') #shape file for Cali map
county_shapefile['geometry'] = county_shapefile['geometry'].apply(lambda geom: shapely.o
```

Regression Plot of population to bedrooms

In [236]:

```
sns.regplot(data=housing, x=housing['population'], y=housing['total_bedrooms'],line_kws=
plt.title('not enough rooms for everyone')
plt.show()
p1 = np.poly1d(np.polyfit(housing['population'], housing['total_bedrooms'], 1))
pv = p1(housing['population'])
r2 = r2_score(housing['total_bedrooms'],pv)
print(p1)
print(r2)
```

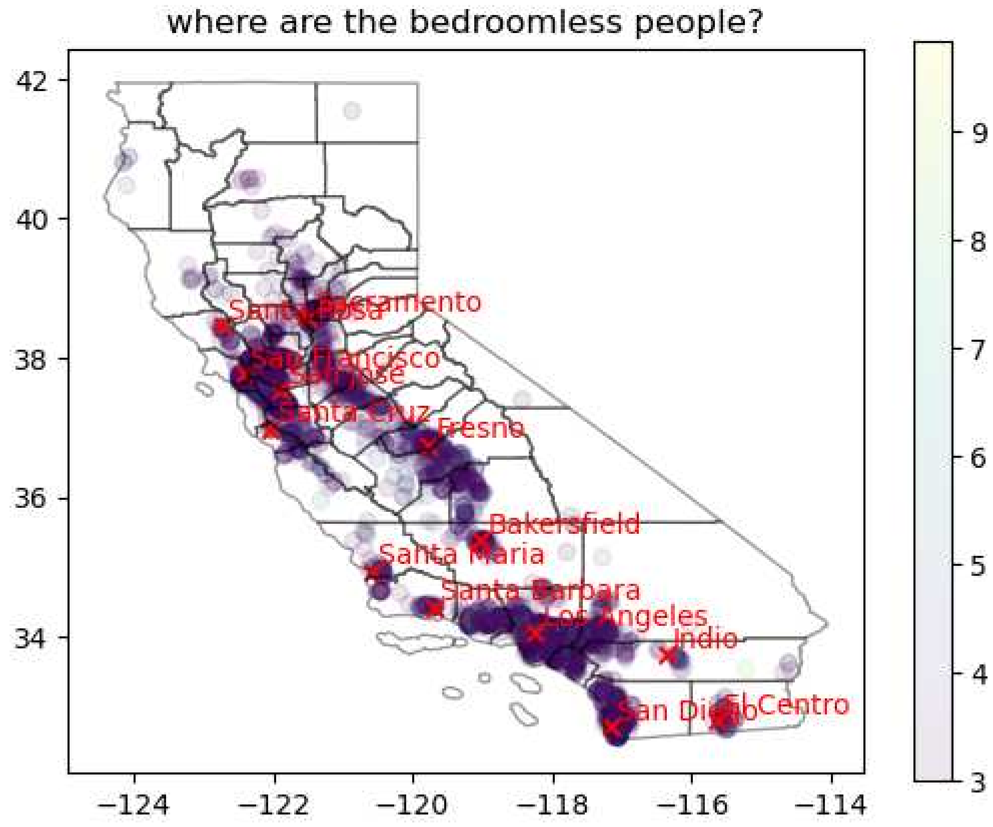


```
0.3317 x + 64.14
0.7536994142357631
```

On average 70% of Californians either share a bedroom or don't have a bedroom to sleep in. A significant amount of these bedless people reside in the Los Angeles area and the San Francisco area as seen in the map below showing the locations where the population and total number of bedrooms exceeds 4:1. We can also see from the regression plot showing the relationship between distance from major city and homelessness, that bedlessness is more frequent the closer you get to a city

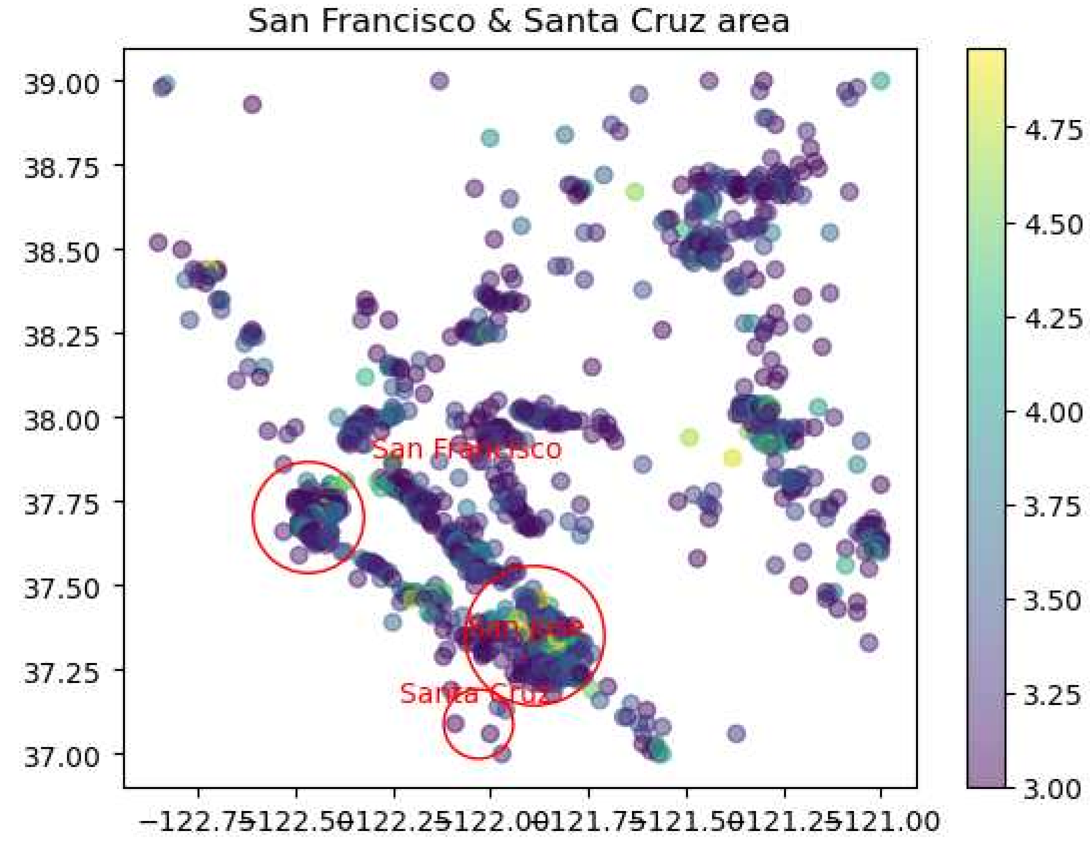
In [237]:

```
calirooms = housing[(housing['population'] >= 3*housing['total_bedrooms'])]
#make sure there are no outliers so that we can visualise better:
upper_fence_rooms_total_bedrooms = np.percentile(calirooms['total_bedrooms'], 75) + 1.5
upper_fence_rooms_population = np.percentile(calirooms['population'], 75) + 1.5 * (np.pe
lower_fence_rooms_total_bedrooms = np.percentile(calirooms['total_bedrooms'], 25) - 1.5
lower_fence_rooms_population = np.percentile(calirooms['population'], 25) - 1.5 * (np.pe
calirooms = calirooms[(calirooms['population'] >= lower_fence_rooms_population) &(caliro
#now lets plot it with respect to a map
fig, ax = plt.subplots()
plt.scatter(x=calirooms['longitude'],y=calirooms['latitude'],alpha=0.1,c=calirooms['popu
plt.colorbar()
plt.scatter(x=california_cities['longitude'],y=california_cities['latitude'],alpha = 1,m
for i in range(california_cities.shape[0]):
    plt.text(x=california_cities.longitude[i]+0.1,y=california_cities.latitude[i]+0.1,s=
county_shapefile.plot(ax=ax, color='None', edgecolor='black',alpha=0.4)
plt.title('where are the bedroomless people?')
plt.show();
```



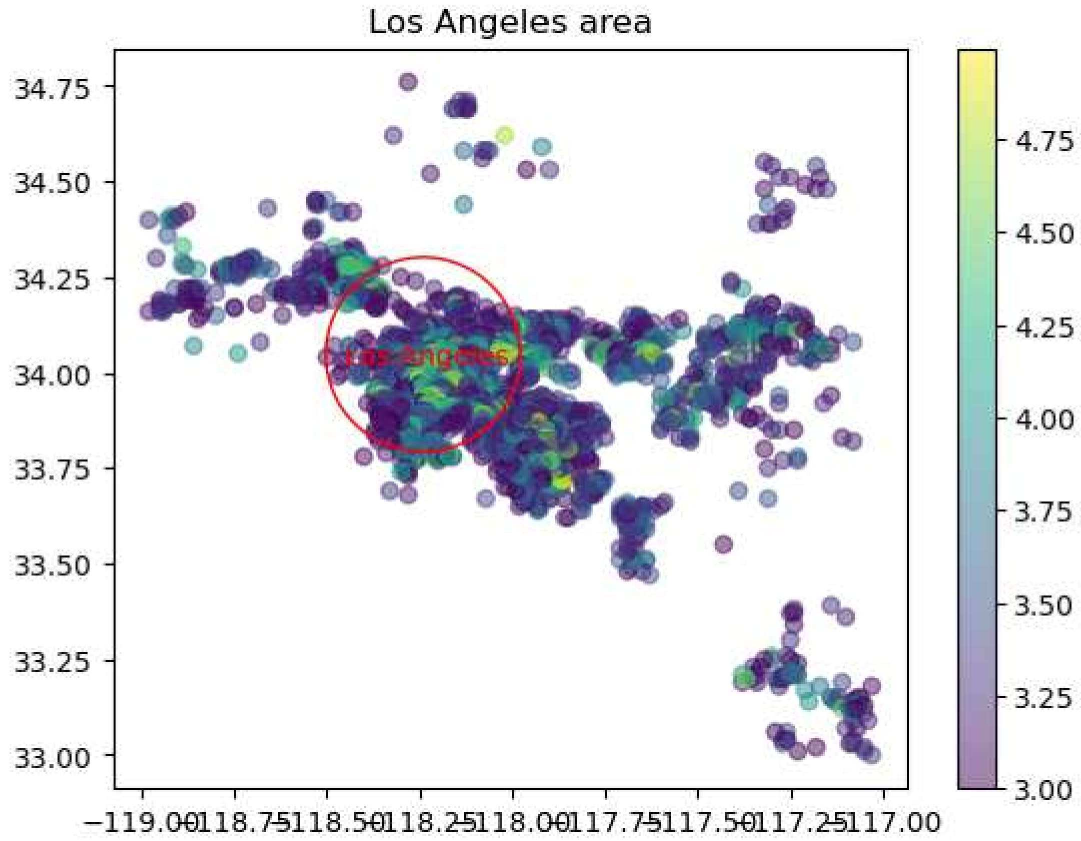
In [238]:

```
calirooms = san[(san['population'] >= 3*san['total_bedrooms'])]
#make sure there are no outliers so that we can visualise better:
#upper_fence_rooms_total_bedrooms = np.percentile(calirooms['total_bedrooms'], 75) + 1.5
#upper_fence_rooms_population = np.percentile(calirooms['population'], 75) + 1.5 * (np.p
#lower_fence_rooms_total_bedrooms = np.percentile(calirooms['total_bedrooms'], 25) - 1.5
#lower_fence_rooms_population = np.percentile(calirooms['population'], 25) - 1.5 * (np.p
#calirooms = calirooms[(calirooms['population'] >= lower_fence_rooms_population) &(calir
#now lets plot it with respect to a map
fig, ax = plt.subplots()
plt.scatter(x=calirooms['longitude'],y=calirooms['latitude'],alpha=0.5,c=calirooms['popu
plt.colorbar()
plt.title('San Francisco & Santa Cruz area')
plt.plot(-122.030797,37.090017, marker="o", markersize=25, markeredgecolor="red", marker
plt.text(-122.230797,37.154117, 'Santa Cruz',c='red')
plt.plot(-121.889096, 37.354960, marker="o", markersize=50, markeredgecolor="red", marke
plt.text(-122.059096, 37.354960, 'San Jose',c='red')
plt.plot(-122.469417,37.704931, marker="o", markersize=40, markeredgecolor="red", marker
plt.text(-122.306034,37.883974, 'San Francisco',c='red')
plt.show();
```



In [239]:

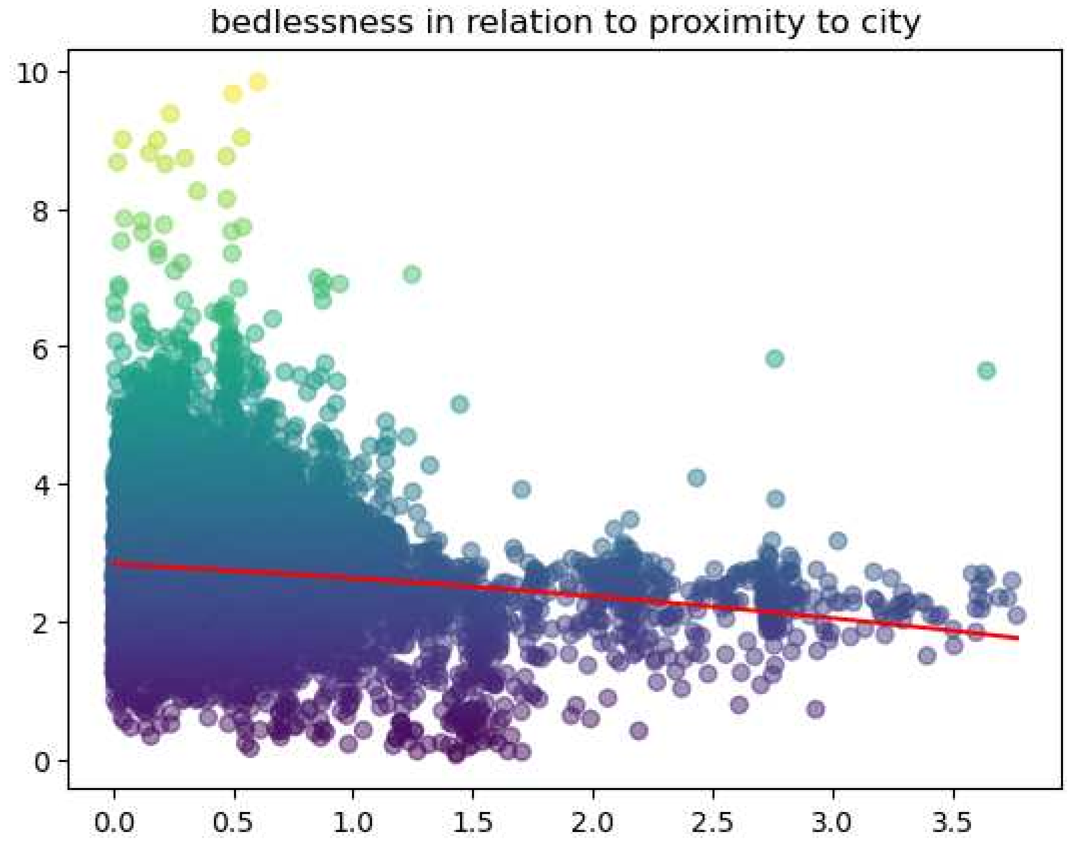
```
calirooms = lostangel[(lostangel['population'] >= 3*lostangel['total_bedrooms'])]
#make sure there are no outliers so that we can visualise better:
upper_fence_rooms_total_bedrooms = np.percentile(calirooms['total_bedrooms'], 75) + 1.5
upper_fence_rooms_population = np.percentile(calirooms['population'], 75) + 1.5 * (np.pe
lower_fence_rooms_total_bedrooms = np.percentile(calirooms['total_bedrooms'], 25) - 1.5
lower_fence_rooms_population = np.percentile(calirooms['population'], 25) - 1.5 * (np.pe
calirooms = calirooms[(calirooms['population'] >= lower_fence_rooms_population) &(caliro
#now lets plot it with respect to a map
fig, ax = plt.subplots()
plt.scatter(x=calirooms['longitude'],y=calirooms['latitude'],alpha=0.5,c=calirooms['popu
plt.colorbar()
#plt.legend()
plt.title('Los Angeles area')
plt.plot(-118.243686,34.052233, marker="o", markersize=70, markeredgecolor="red", marker
plt.text(-118.455,34.022233, 'Los Angeles',c='red')
plt.show();
```



In [249]:

```
plt.title('bedlessness in relation to proximity to city')
p2 = np.poly1d(np.polyfit(housing['distance_to_closest_city'], housing['bedless'], 2))
housefit = np.linspace(housing['distance_to_closest_city'].min(), housing['distance_to_c
plt.scatter(x=housing['distance_to_closest_city'], y=housing['bedless'],alpha=0.5,c=hous
plt.plot(housefit,p2(housefit),label="Average",c='red')
pv = p2(housing['distance_to_closest_city'])
r2 = r2_score(housing['bedless'],pv)
print(r2)
plt.show()
```

```
0.014904420412508168
```



Median income and house value as approaching city

Median Income

In [48]:

```
caliincomebin = pd.cut(housing['median_income'], 4, precision=2)
calivalubin = pd.cut(housing['median_house_value'], 4, precision=2)

conditions = [
    (housing['median_income'] <= 2.24),
    (housing['median_income'] > 2.24) & (housing['median_income'] <= 4.48),
    (housing['median_income'] > 4.48) & (housing['median_income'] <= 8),
    (housing['median_income'] > 8)
    ]
fence_house_value = np.percentile(housing['median_house_value'], 75) + 1.5 * (np.percent
Vconditions = [
    (housing['median_house_value'] <= np.percentile(housing['median_house_value'], 25)),
    (housing['median_house_value'] > np.percentile(housing['median_house_value'], 25)) &
    (housing['median_house_value'] > np.percentile(housing['median_house_value'], 75)) &
    (housing['median_house_value'] > fence_house_value)
    ]

status = ['poor', 'middle_class', 'wealthy', 'uber_rich']
housing['status'] = np.select(conditions, status)
house_bins = ['Low_Value', 'Medium_Value', 'High_Value', 'Highest_Value']
housing['house_bin'] = np.select(Vconditions, house_bins)
housing.head()
```

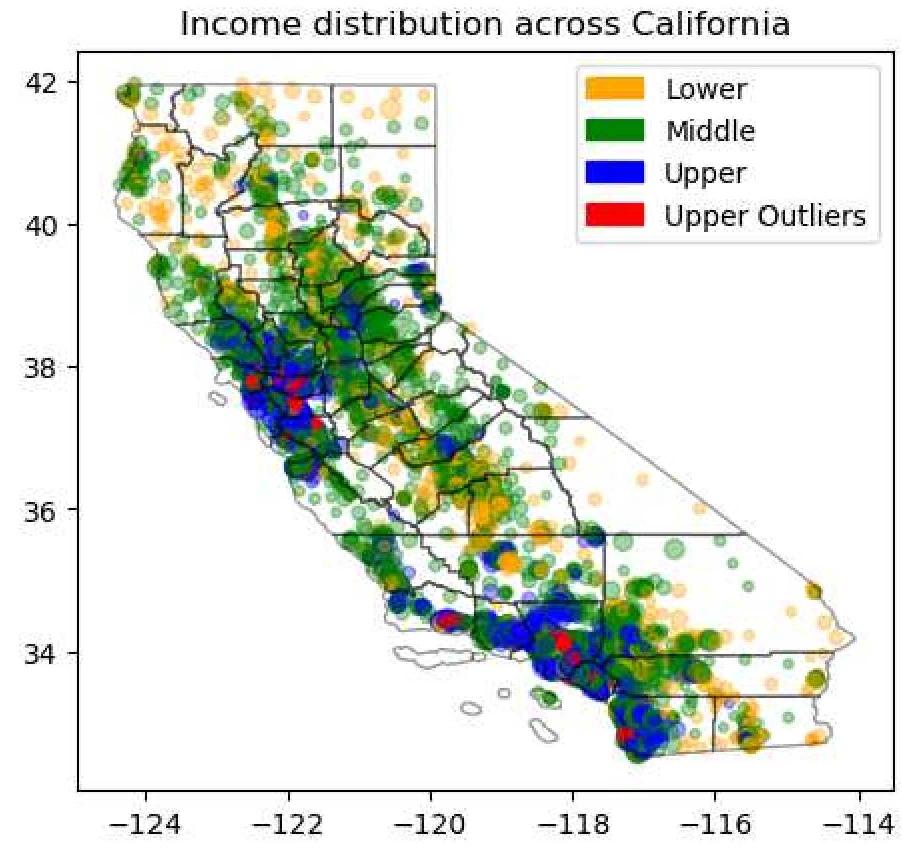
```
C:\Users\benjo\AppData\Local\Temp\ipykernel_4648\19209354.py:19: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  housing['status'] = np.select(conditions, status)
C:\Users\benjo\AppData\Local\Temp\ipykernel_4648\19209354.py:21: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  housing['house_bin'] = np.select(Vconditions, house_bins)
```

Out[48]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41	880	129.0	322	
1	-122.22	37.86	21	7099	1106.0	2401	
2	-122.24	37.85	52	1467	190.0	496	
3	-122.25	37.85	52	1274	235.0	558	
4	-122.25	37.85	52	1627	280.0	565	

In [47]:

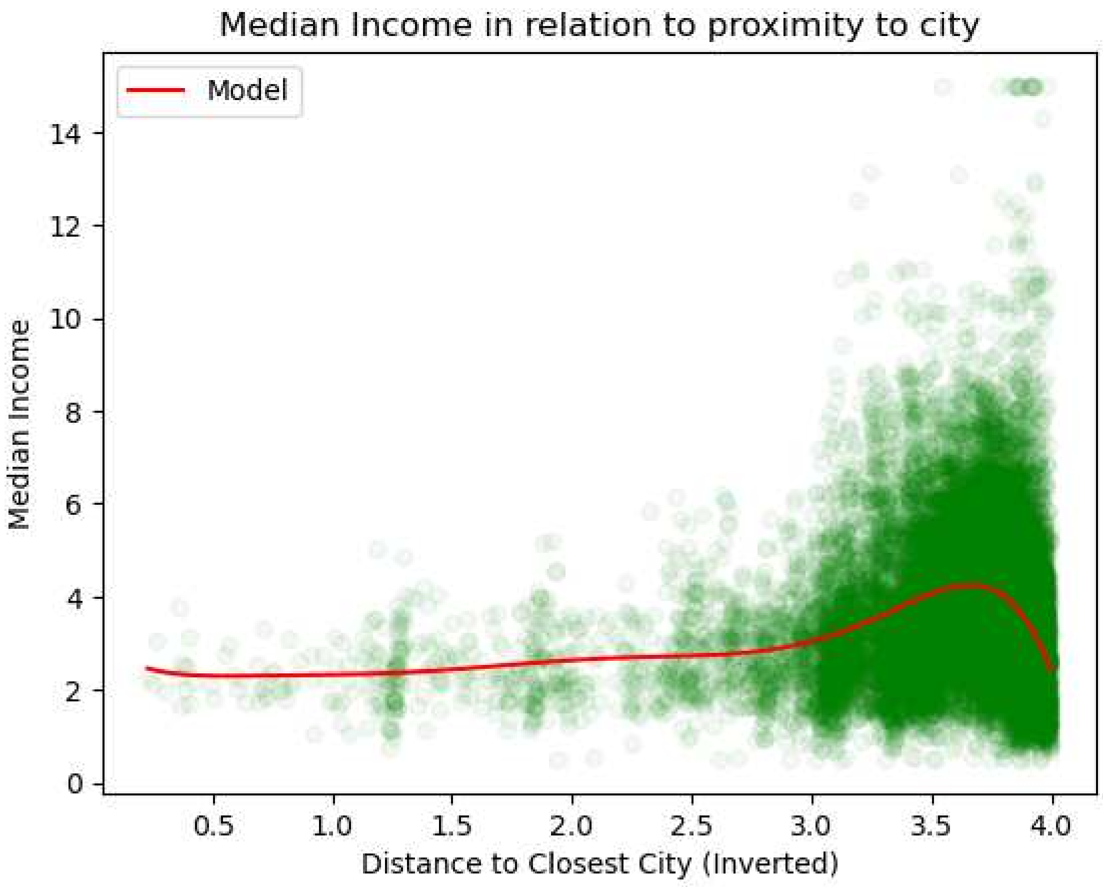
```
fig, ax = plt.subplots()

status_colors = {'poor': 'orange', 'middle_class': 'green', 'wealthy': 'blue', 'uber_ric

# Map the categories to colors for each data point
statuscolors = housing['status'].map(status_colors)


#county_shapefile.plot(ax=ax, color='white', edgecolor='black')

plt.scatter(x=housing['longitude'],y=housing['latitude'],alpha = (35+((1.066**(100*(hous

#plt.scatter(x=california_cities['longitude'],y=california_cities['latitude'],alpha = 1,
#for i in range(california_cities.shape[0]):
#    plt.text(x=california_cities.longitude[i]+0.1,y=california_cities.latitude[i]+0.1,s
county_shapefile.plot(ax=ax, color='None', edgecolor='black',alpha=0.4)

plt.title("Income distribution across California")


outliars = mpatches.Patch(color='red', label='Upper Outliers')
poor = mpatches.Patch(color='orange', label='Lower')
mid = mpatches.Patch(color='green', label='Middle')
upper = mpatches.Patch(color='blue', label='Upper')
plt.legend(handles=[poor,mid,upper,outliars])



plt.show();
```



In [196]:

```
plt.title('Median Income in relation to proximity to city')
p2 = np.poly1d(np.polyfit(housing['distance_to_closest_city'], housing['median_income'],
housefit = np.linspace(housing['distance_to_closest_city'].min(), housing['distance_to_c
plt.scatter(x=4-housing['distance_to_closest_city'], y=housing['median_income'], c='gree
plt.plot(4-housefit, p2(housefit), label="Model", c='red')
print(r2)
plt.legend()
plt.xlabel('Distance to Closest City (Inverted)')
plt.ylabel('Median Income')
plt.show()
```

```
0.095043303104324O7
```

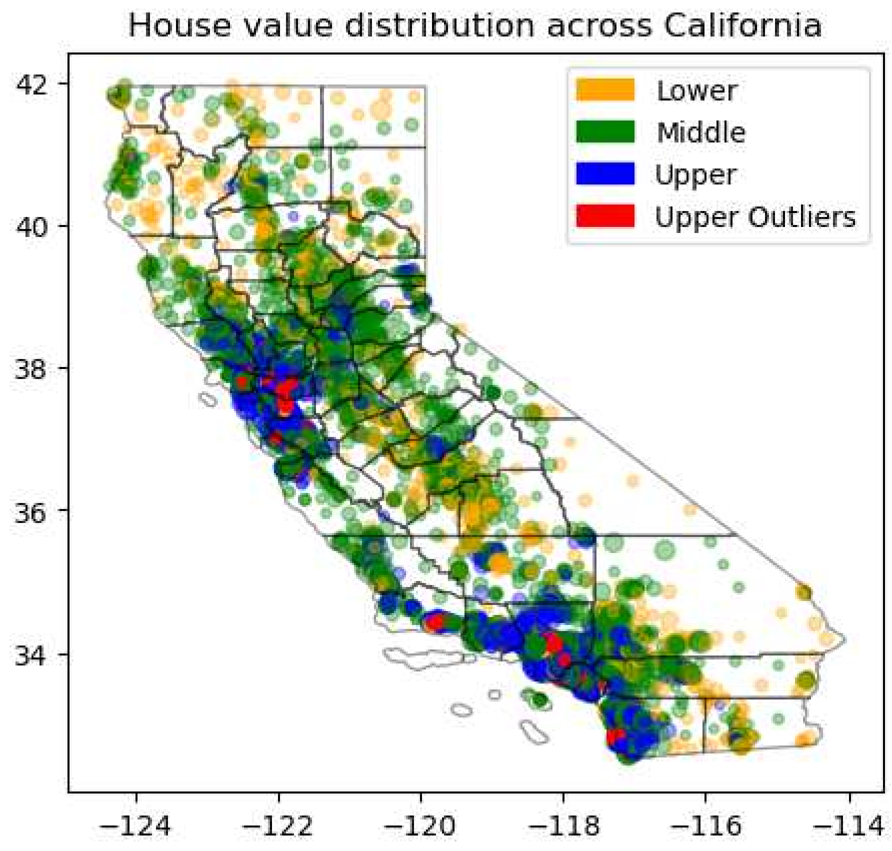


In [ ]:

Median House price

In [49]:

```
fig, ax = plt.subplots()

house_bin_colors = {'Low_Value': 'orange', 'Medium_Value': 'green', 'High_Value': 'blue'

# Map the categories to colors for each data point
house_bin_colors = housing['house_bin'].map(house_bin_colors)


plt.scatter(x=housing['longitude'],y=housing['latitude'],alpha = (35+((1.066**(100*(hous
county_shapefile.plot(ax=ax, color='None', edgecolor='black',alpha=0.4)

plt.title("House value distribution across California")


Highest_Value = mpatches.Patch(color='red', label='Highest_Value')
Low_Value = mpatches.Patch(color='orange', label='Low_Value')
Medium_Value = mpatches.Patch(color='green', label='Medium_Value')
High_Value = mpatches.Patch(color='blue', label='High_Value')
plt.legend(handles=[poor,mid,upper,outliars])



plt.show();
```

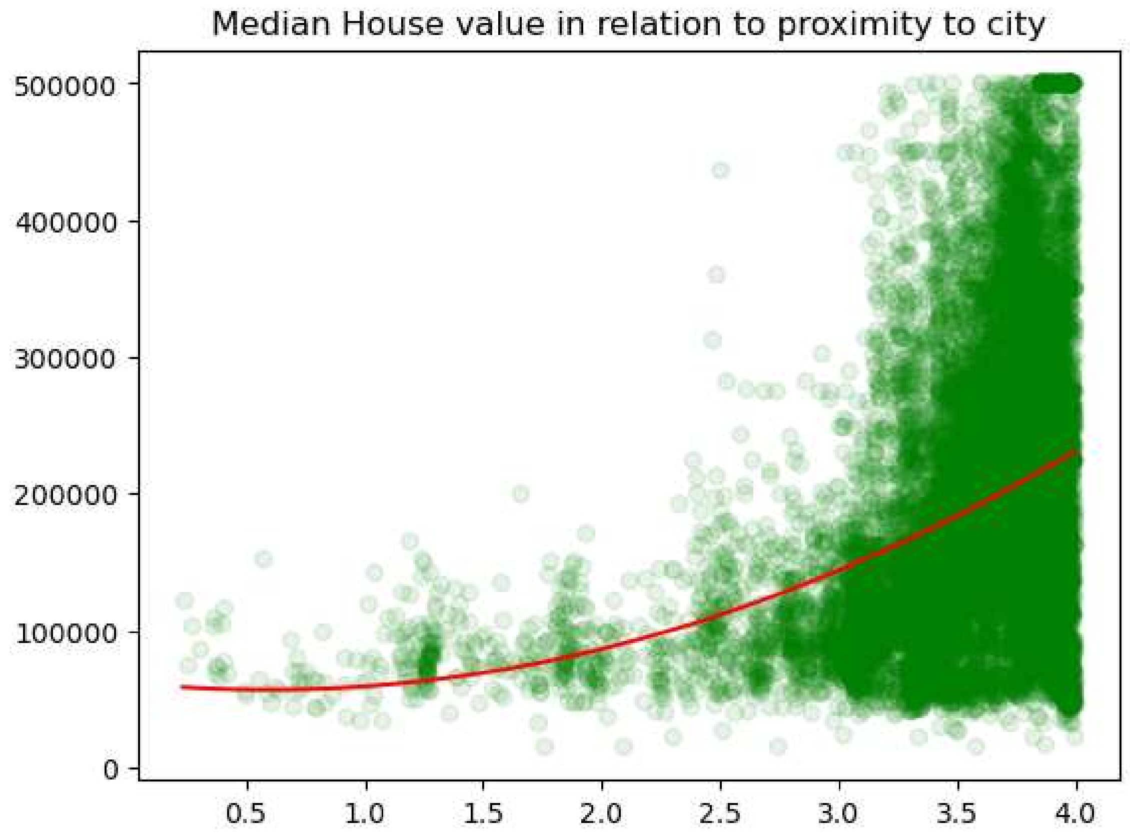


In [ ]:

In [215]:

```
plt.title('Median House value in relation to proximity to city')
p2 = np.poly1d(np.polyfit(housing['distance_to_closest_city'], housing['median_house_val
housefit = np.linspace(housing['distance_to_closest_city'].min(), housing['distance_to_c
plt.scatter(x=4-housing['distance_to_closest_city'], y=housing['median_house_value'],c='
plt.plot(4-housefit,p2(housefit),label="Average",c='red')
pv = p2(housing['distance_to_closest_city'])
r2 = r2_score(housing['median_house_value'],pv)
print(r2)
plt.show()
```

```
0.09123367804765248
```



In [ ]: