Task 2: Data Cleansing and Transformation

Instructions:

1. Acquire a real-world dataset requiring data cleaning and transformation.
2. Address data quality issues (missing values, inconsistent formats, outliers).
3. Develop a cleaning strategy (imputation, outlier detection, normalization).
4. Implement necessary transformation steps (feature engineering, aggregation).
5. Validate the cleaned and transformed dataset for integrity and usability.
6. Document the steps taken and provide clear explanations.
7. Present the cleaned and transformed dataset for further analysis.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

In [2]:
```
data = pd.read_csv("Datasets/california_housing/housing.csv")
```

1. Acquired Real-world dataset

Dataset details

- Description: The dataset is California Housing Prices dataset.
- Columns: [longitude, latitude, housing_median_age, total_rooms, total_bedrooms, population, households, median_income, median_house_value, ocean_proximity]

In [3]:
```
# data.head()
data.info()
```

```
# data['longitude'].isnull().value_counts()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

- Only one categorical data Ocean_Proximity

In [4]:
```
data['ocean_proximity'].value_counts()
```

Out[4]:
```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

In [5]:
```
data.describe()
```

Out[5]:

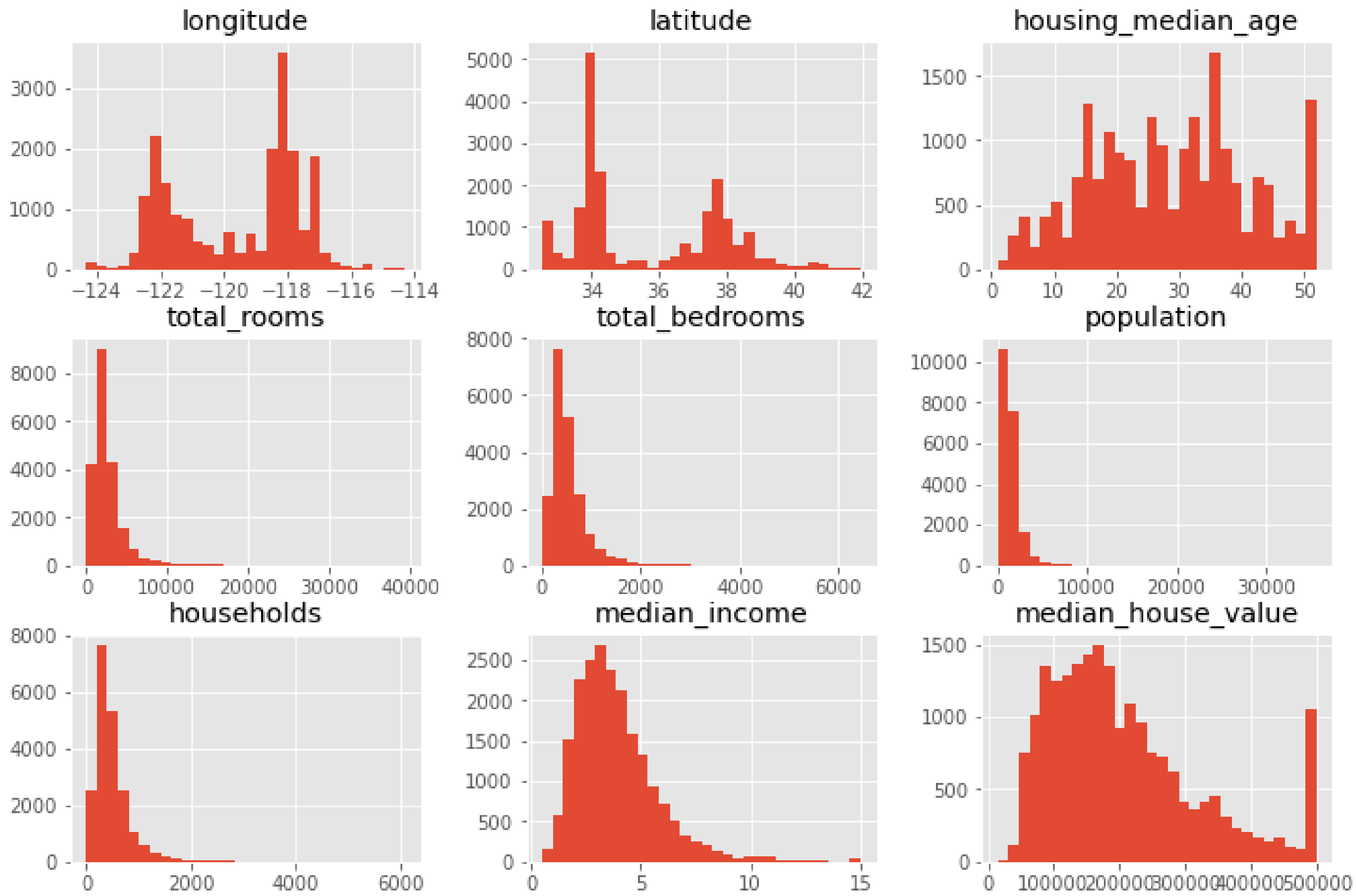
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	r
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	

In [6]:

```
data.hist(bins=30,figsize=(12,8))
```

Out[6]:

```
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'}>]],
      dtype=object)
```



2. Data quality and 3. Data cleaning

Pre-processing

- For this task, the data in not normally distributed. Using Stratified sampling technique to prepare the test dataset.
- Creating a new feature income_label which is income category and used if for sampling.

In [7]:
```
data['income_label']=np.ceil(data['median_income']/1.5)
data['income_label'].where(data['income_label']<5,5.0,inplace=True)
```

In [8]:
```
from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits=1,test_size=0.2,random_state=42)

for train_index,test_index in split.split(data,data['income_label']):
    strat_train_set=data.loc[train_index]
    strat_test_set=data.loc[test_index]
```

In [9]:
```
strat_train_set.drop('income_label',axis=1,inplace=True)
strat_test_set.drop('income_label',axis=1,inplace=True)
strat_train_set.to_csv("Datasets/california_housing/strat_train_set.csv",index=False)
strat_test_set.to_csv("Datasets/california_housing/strat_test_set.csv",index=False)
```

In [24]:
```
data=pd.read_csv('Datasets/california_housing/strat_train_set.csv')
# data.info()
```

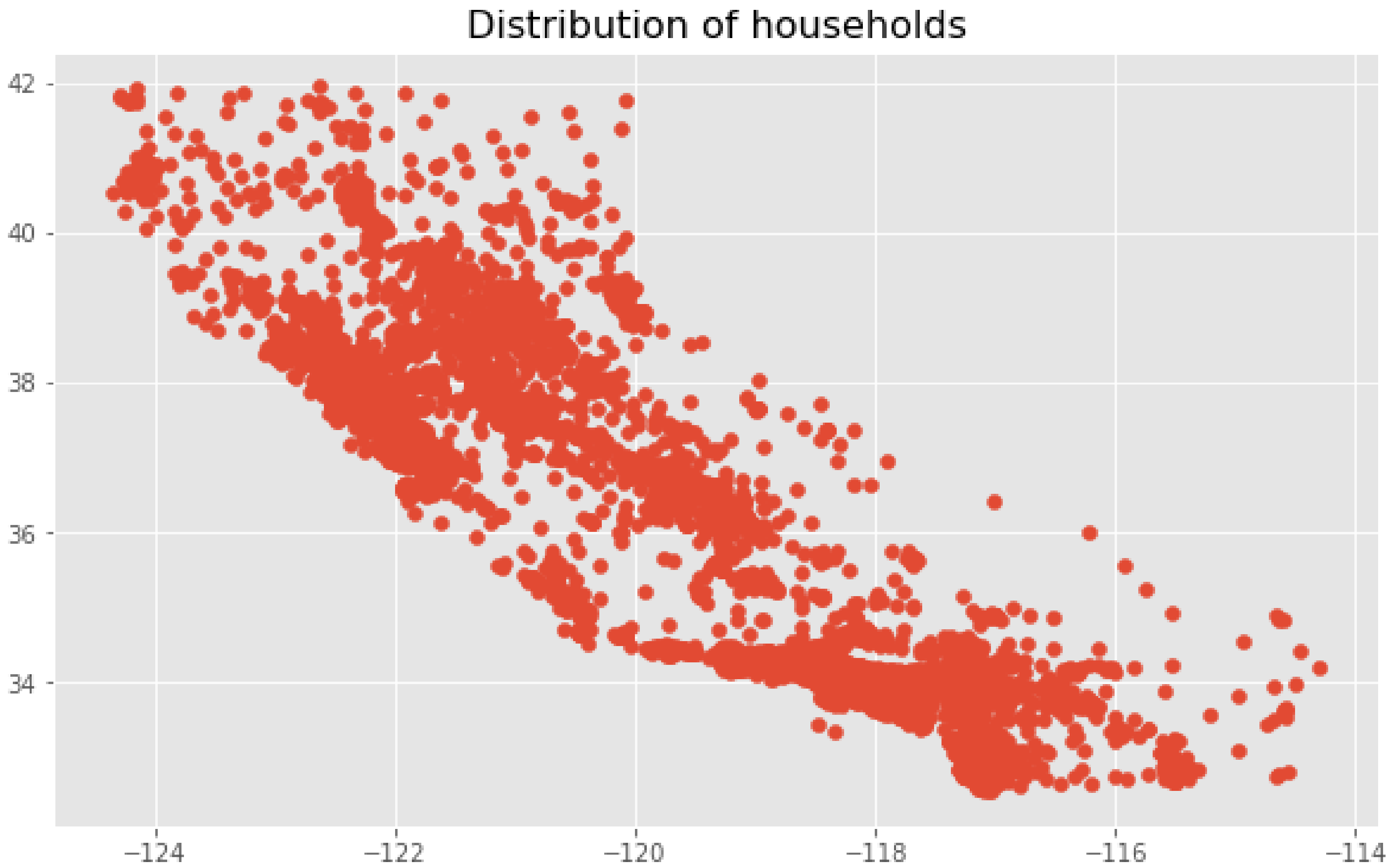
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16512 entries, 0 to 16511
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           16512 non-null  float64
 1   latitude            16512 non-null  float64
 2   housing_median_age  16512 non-null  float64
 3   total_rooms         16512 non-null  float64
 4   total_bedrooms      16354 non-null  float64
 5   population          16512 non-null  float64
 6   households          16512 non-null  float64
 7   median_income       16512 non-null  float64
 8   median_house_value  16512 non-null  float64
 9   ocean_proximity     16512 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.3+ MB
```

EDA

- Carrying out various visualization on train dataset for realising patterns, correlations and getting the sense of the data

In [11]:
```
plt.figure(figsize=(10,6))
plt.scatter(x=data['longitude'],y=data['latitude'])
plt.title("Distribution of households",size=16)
```

Out[11]:
```
Text(0.5, 1.0, 'Distribution of households')
```

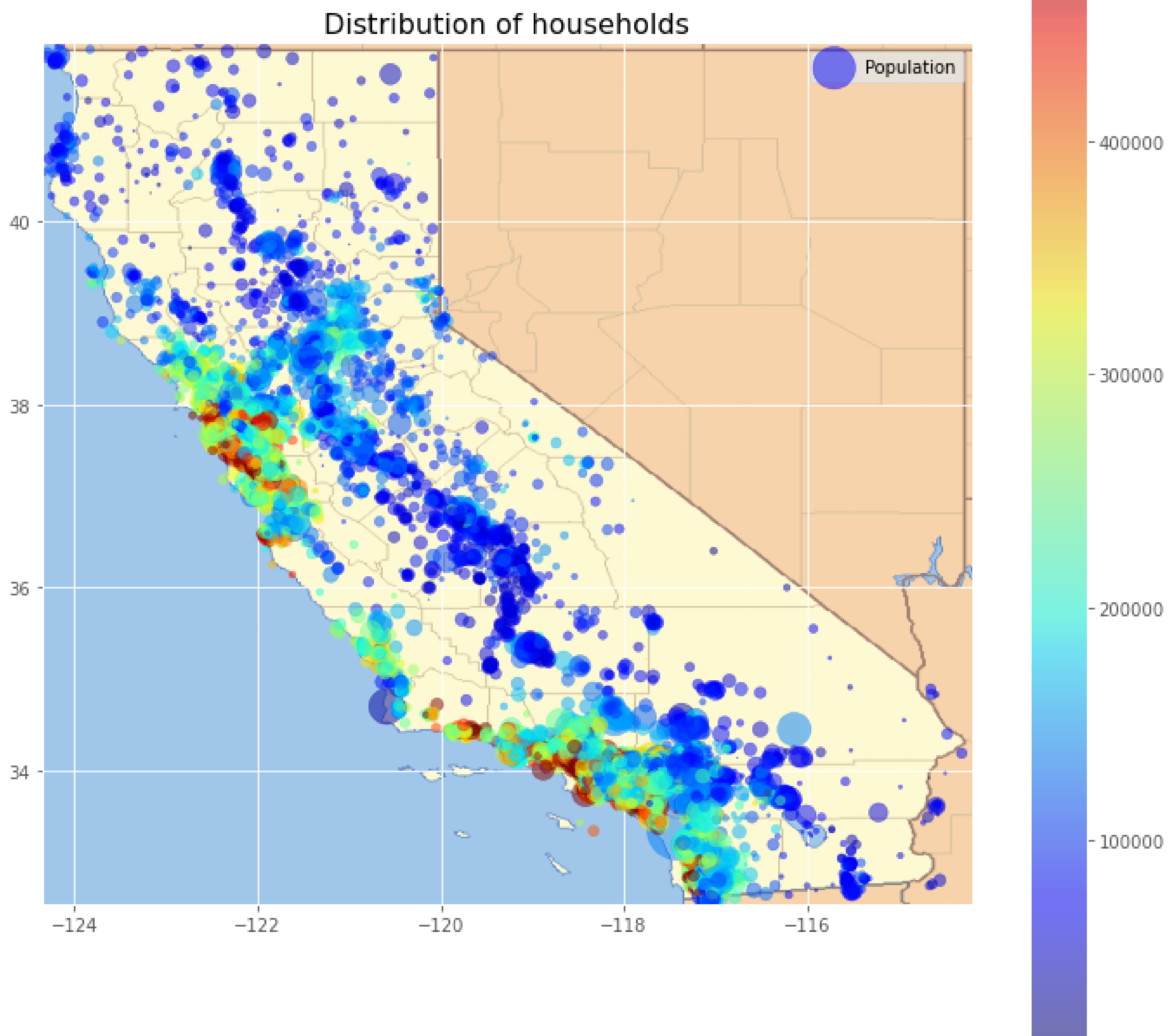


In [12]:
```
plt.figure(figsize=(12,12))
img=plt.imread('Datasets/california_housing/california.png')
plt.imshow(img,zorder=0,extent=[-124.35,-114.2,32.54,41.95])

plt.scatter(x=data['longitude'],y=data['latitude'],alpha=0.5,s=data['population']/30,c=data['median_house_value']
plt.colorbar()
plt.title("Distribution of households",size=16)
plt.legend()
```

```
C:\Users\MAHAVIR\AppData\Local\Temp\ipykernel_5504\1422664156.py:6: MatplotlibDeprecationWarning: Auto-removal of
grids by pcolor() and pcolormesh() is deprecated since 3.5 and will be removed two minor releases later; please c
all grid(False) first.
  plt.colorbar()
```

Out[12]:

```
<matplotlib.legend.Legend at 0x1fad86b2c70>
```



Distribution of households
Population
40
38
36
34
−124
−122
−120
−118
−116
500000
400000
300000
200000
100000

Inights from Visualization

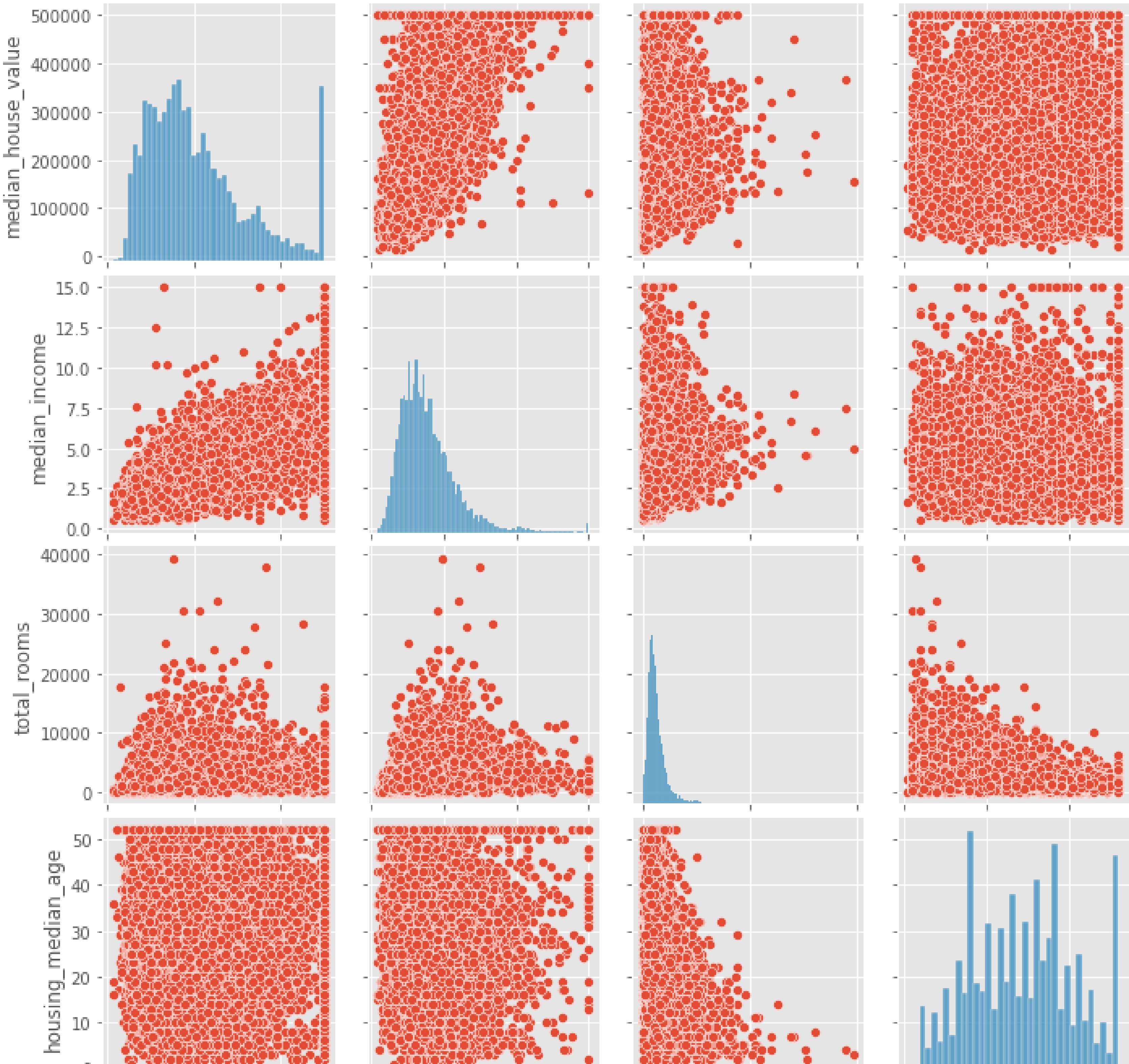
- Housing prices are much related to location and population density.
- Housing prices near ocean are higher except in northern california.
- Now, see the correlation of 'medial house value' with other columns. This is Pearson's correlation coefficient.

In [13]:
```
corr_matrix=data.corr()
corr_matrix['median_house_value'].sort_values(ascending=False)
```

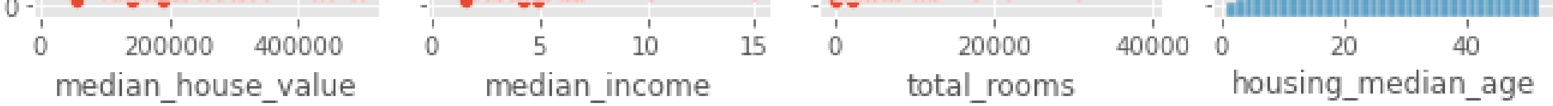
Out[13]:
```
median_house_value    1.000000
median_income         0.687151
total_rooms           0.135140
housing_median_age    0.114146
households            0.064590
total_bedrooms        0.047781
population           -0.026882
longitude            -0.047466
latitude             -0.142673
Name: median_house_value, dtype: float64
```

In [14]:
```
sns.pairplot(data[['median_house_value','median_income','total_rooms','housing_median_age']])
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x1fad7374400>
```



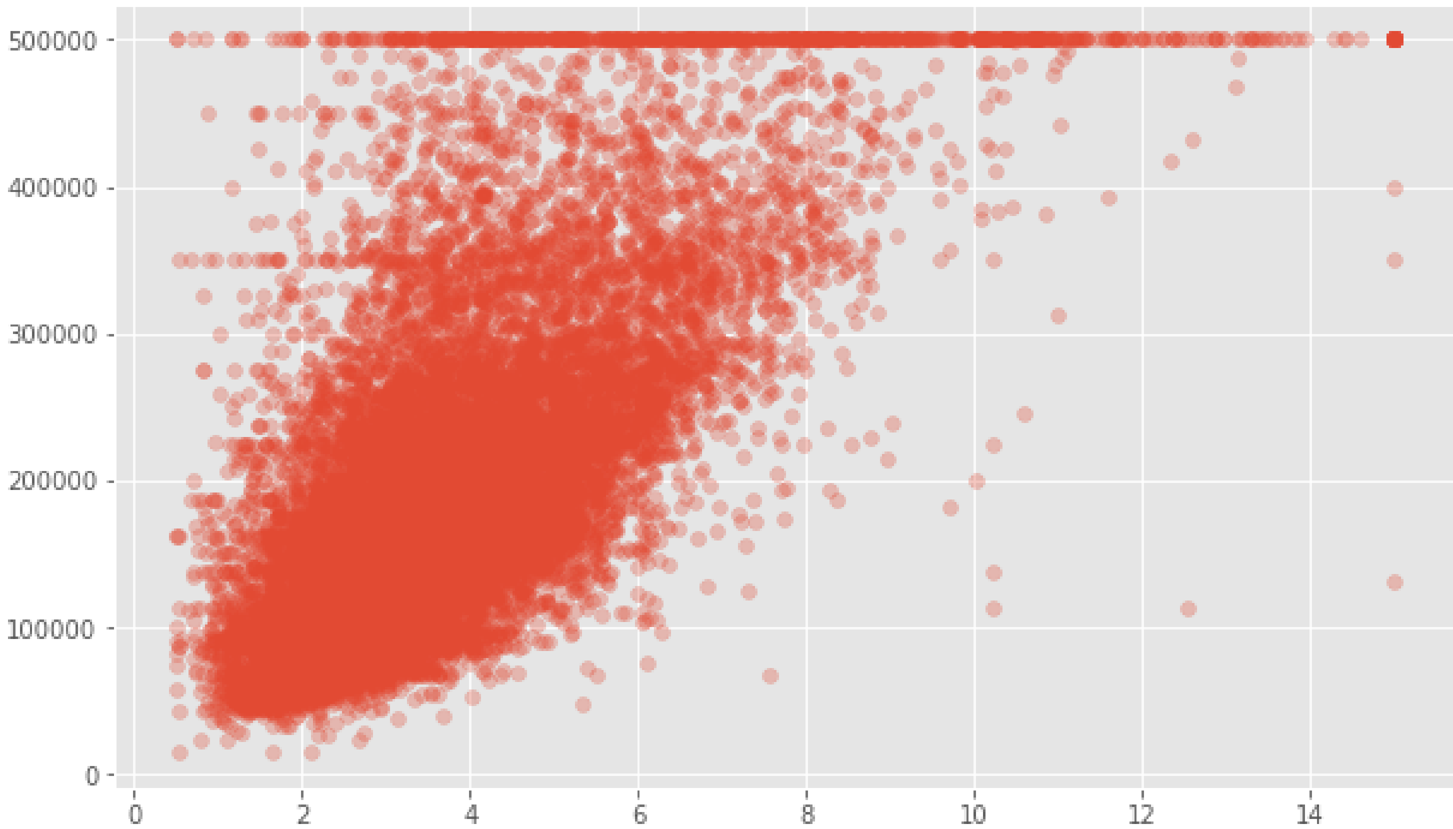
median_house_value
500000
400000
300000
200000
100000
0
median_income
15.0
12.5
10.0
7.5
5.0
2.5
0.0
total_rooms
40000
30000
20000
10000
0
housing_median_age
50
40
30
20
10



- Median Income is the most promising attribute to get Median Hosung Price

In [15]:
```
plt.figure(figsize=(10,6))
plt.scatter(y=data['median_house_value'],x=data['median_income'],alpha=0.3)
```

Out[15]:
```
<matplotlib.collections.PathCollection at 0x1fadb55e8e0>
```



- A clear line can be seen at 500k at which the data is capped. Similar lines can be seen around 450k,350k. This kind of data may degrade the performance of model.

4. Feature Engineering

Creating new features:

- rooms per household
- bedrooms per room
- population per household

In [16]:
```
data_copy = data.copy()
```

In [17]:
```
data_copy['rooms_per_household']=data_copy['total_rooms']/data_copy['households']
data_copy['bedrooms_per_room']=data_copy['total_bedrooms']/data_copy['total_rooms']
data_copy['population_per_household']=data_copy['population']/data_copy['households']
# data_copy.head()
```

In [18]:
```
corr_matrix=data_copy.corr()
corr_matrix['median_house_value'].sort_values(ascending=False)
```

Out[18]:
```
median_house_value          1.000000
median_income               0.687151
rooms_per_household         0.146255
total_rooms                 0.135140
housing_median_age          0.114146
households                  0.064590
total_bedrooms              0.047781
population_per_household   -0.021991
population                 -0.026882
longitude                  -0.047466
latitude                   -0.142673
bedrooms_per_room          -0.259952
Name: median_house_value, dtype: float64
```

5. Validation of new features and data

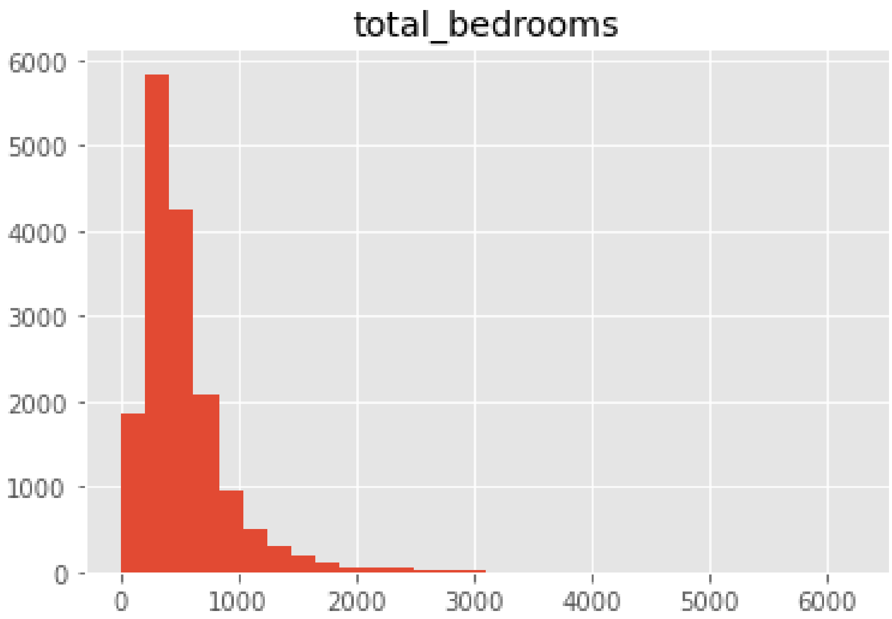
- It is clear that rooms_per_household and bedrooms_per_room have better correlation with median_house_value than total_rooms and total_bedrooms.
- Later Feature adder class needs to be created later during testing.(tranforming the data to have new features)

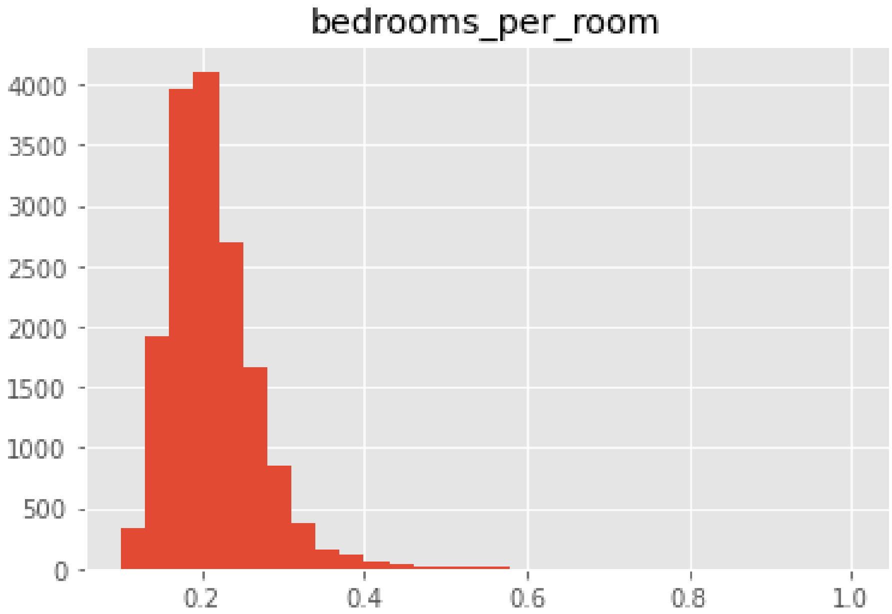
In [19]:
```
data.isnull().value_counts()
```

Out[19]:
```
longitude  latitude  housing_median_age  total_rooms  total_bedrooms  population  households  median_income  medi
an_house_value  ocean_proximity
False      False     False               False        False           False       False       False          Fals
e               False               16354
                                                      True            False       False       False          Fals
e               False                 158
dtype: int64
```

In [20]:
```
data_copy.hist(column='total_bedrooms',bins=30)
data_copy.hist(column='bedrooms_per_room',bins=30)
```

Out[20]:
```
array([[<AxesSubplot:title={'center':'bedrooms_per_room'}>]], dtype=object)
```

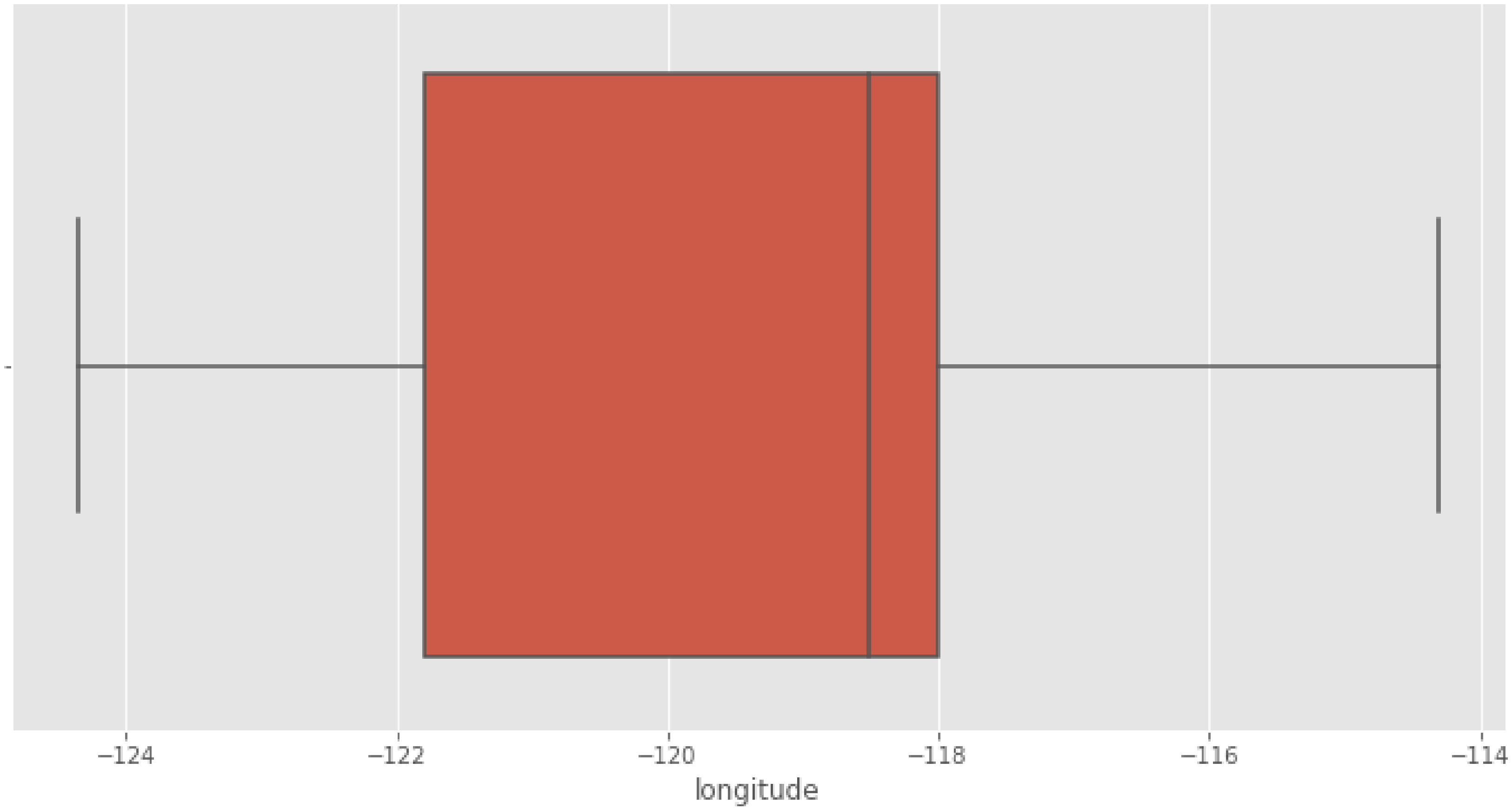


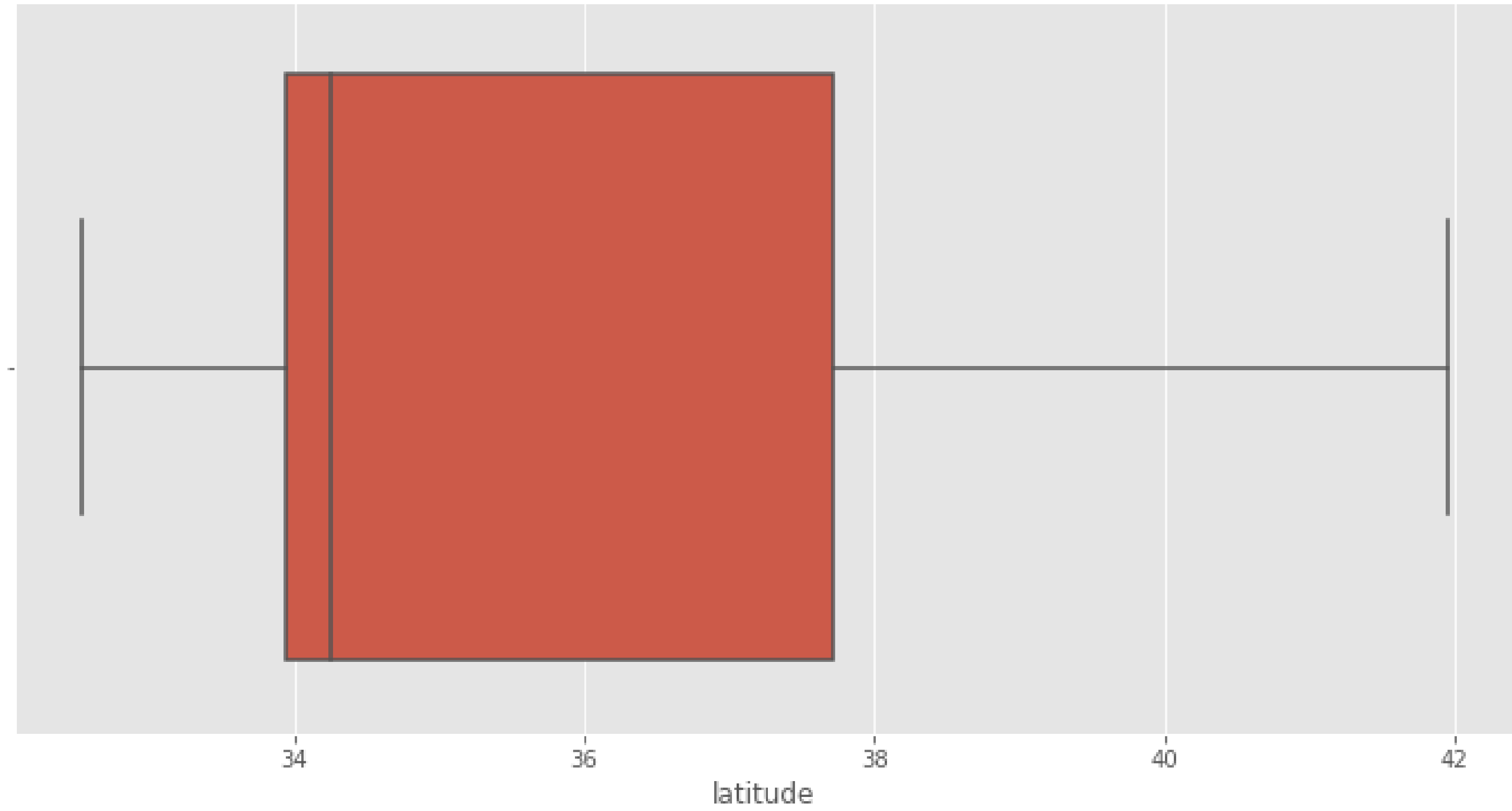


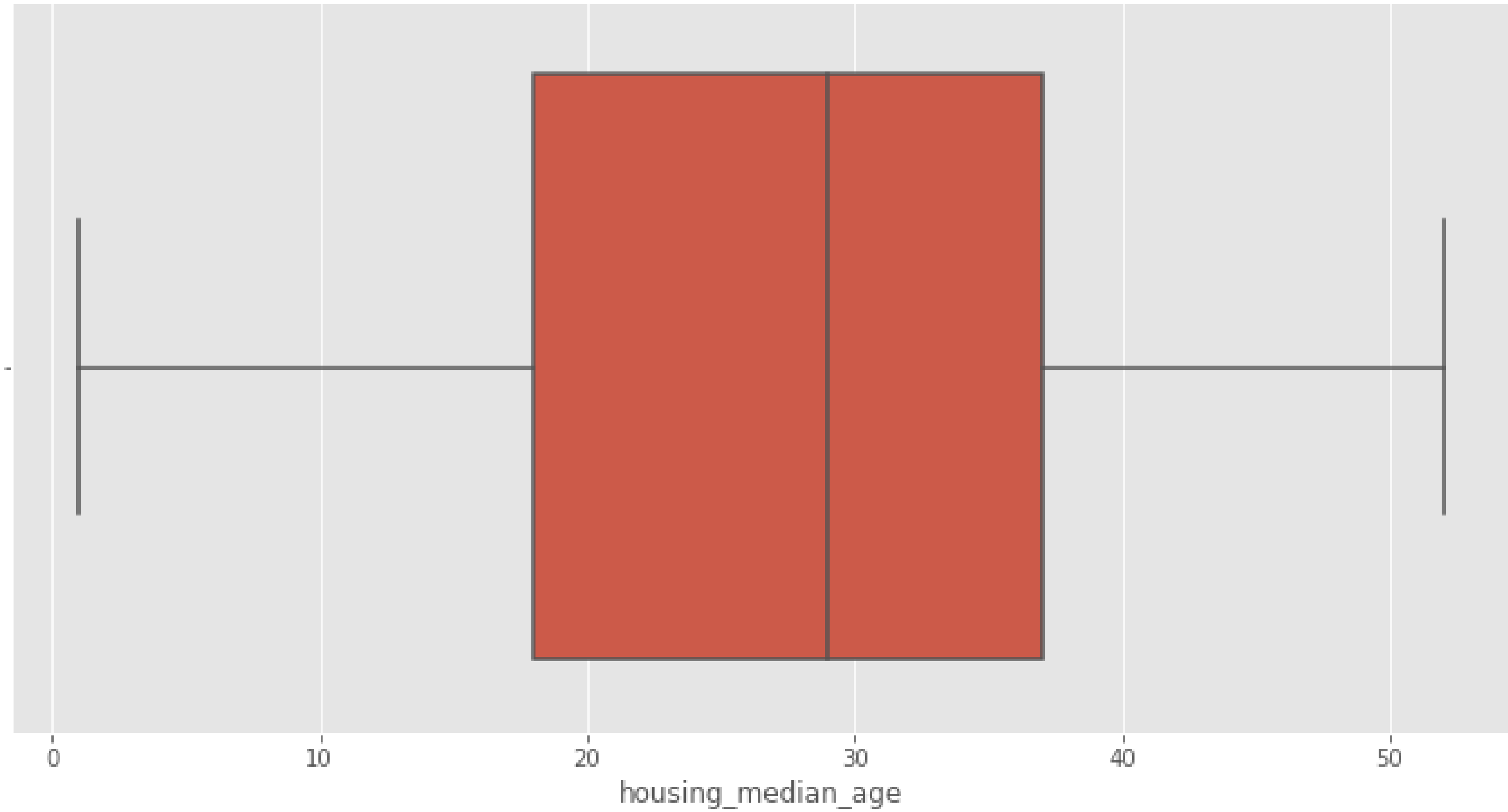
Cleaning the data

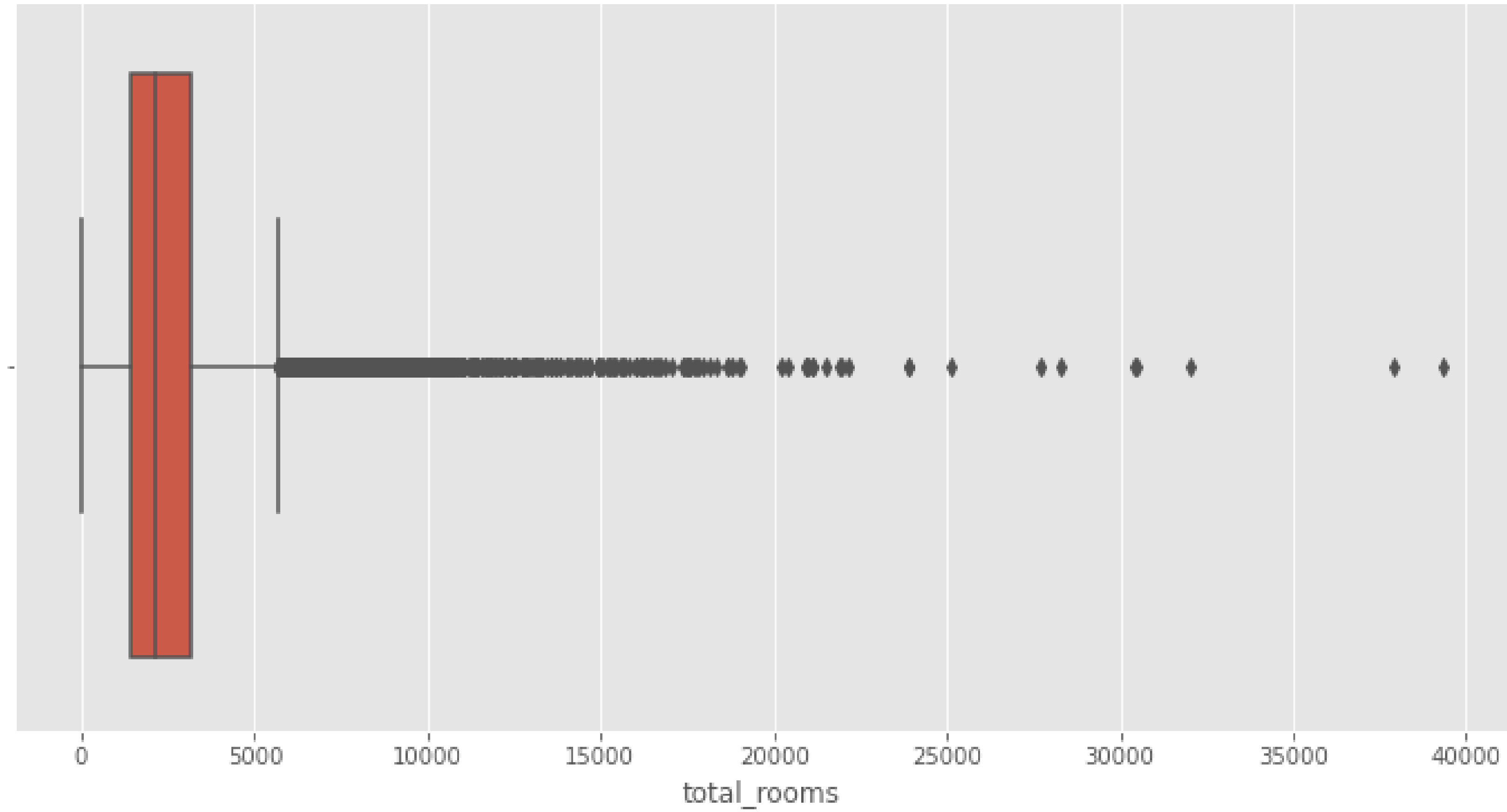
- Removing Outliers realised through boxplots

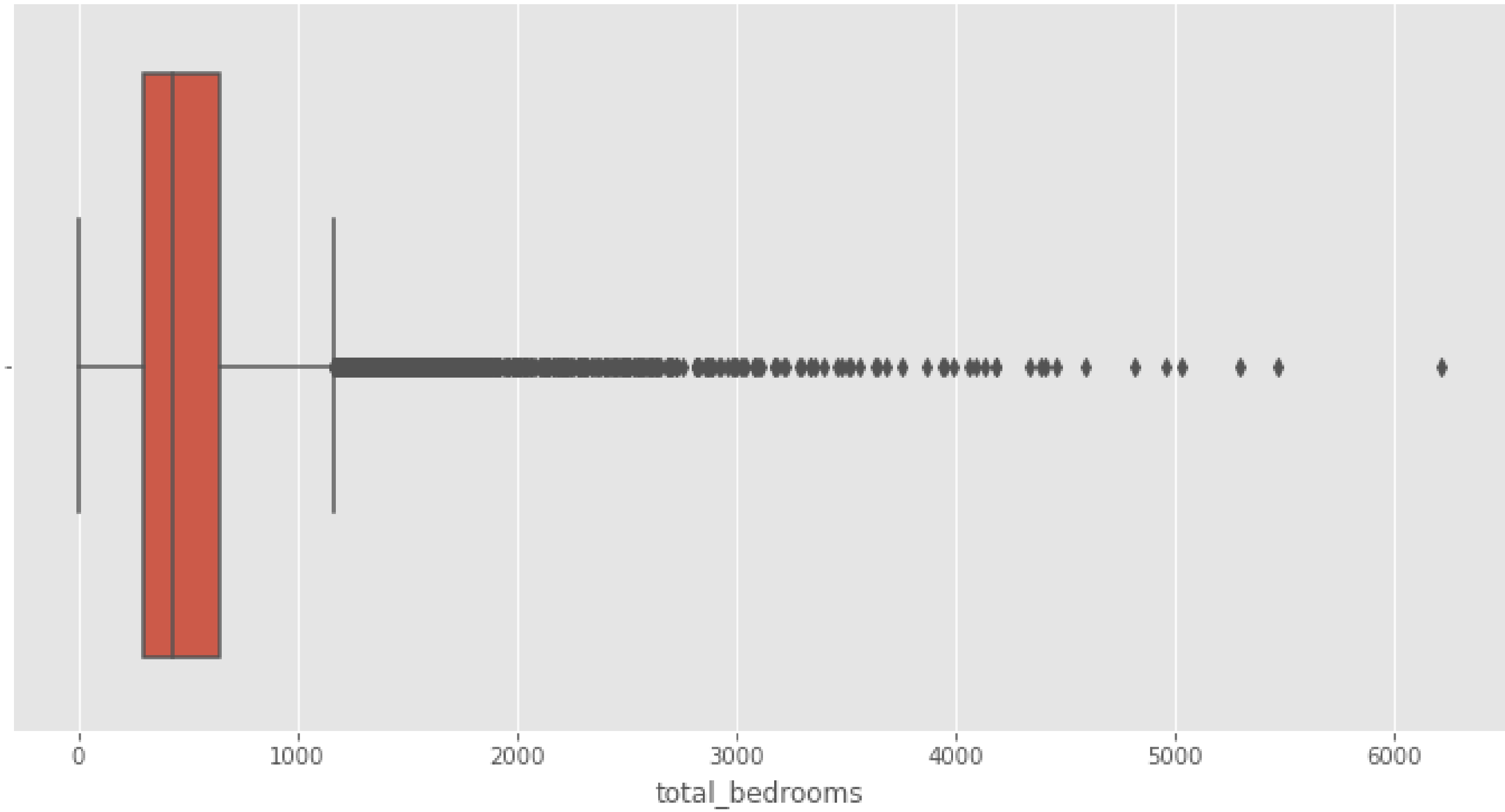
In [21]:
```
num_features=['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value', 'rooms_per_household',
       'bedrooms_per_room', 'population_per_household']

for i in num_features:
    fig, ax = plt.subplots()
    fig.set_size_inches(12,6)
    #plt.xlim(-10,10)
    sns.boxplot(x=i,data=data_copy,ax=ax)
```

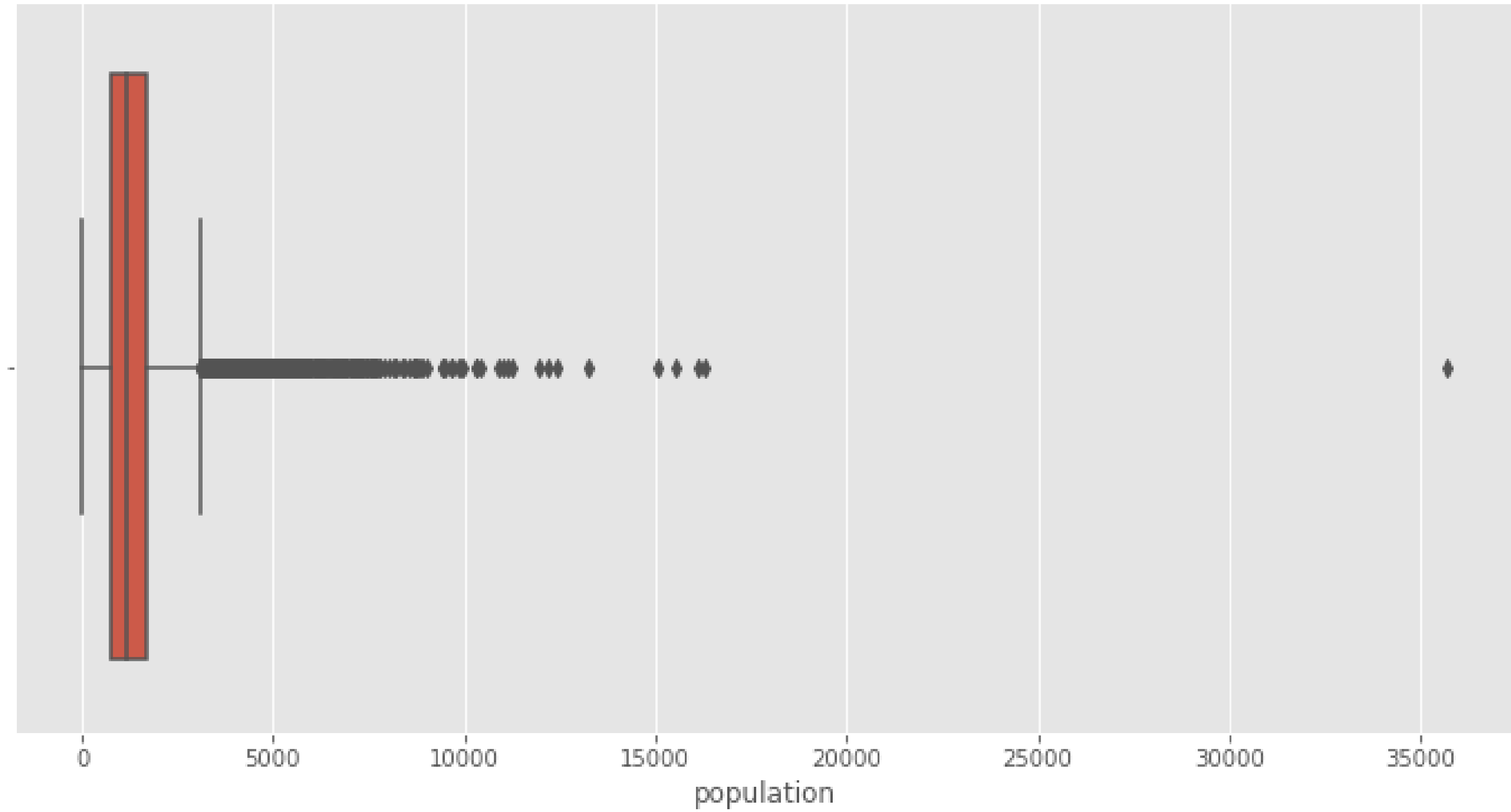

−124
−122
−120
−118
−116
−114
longitude

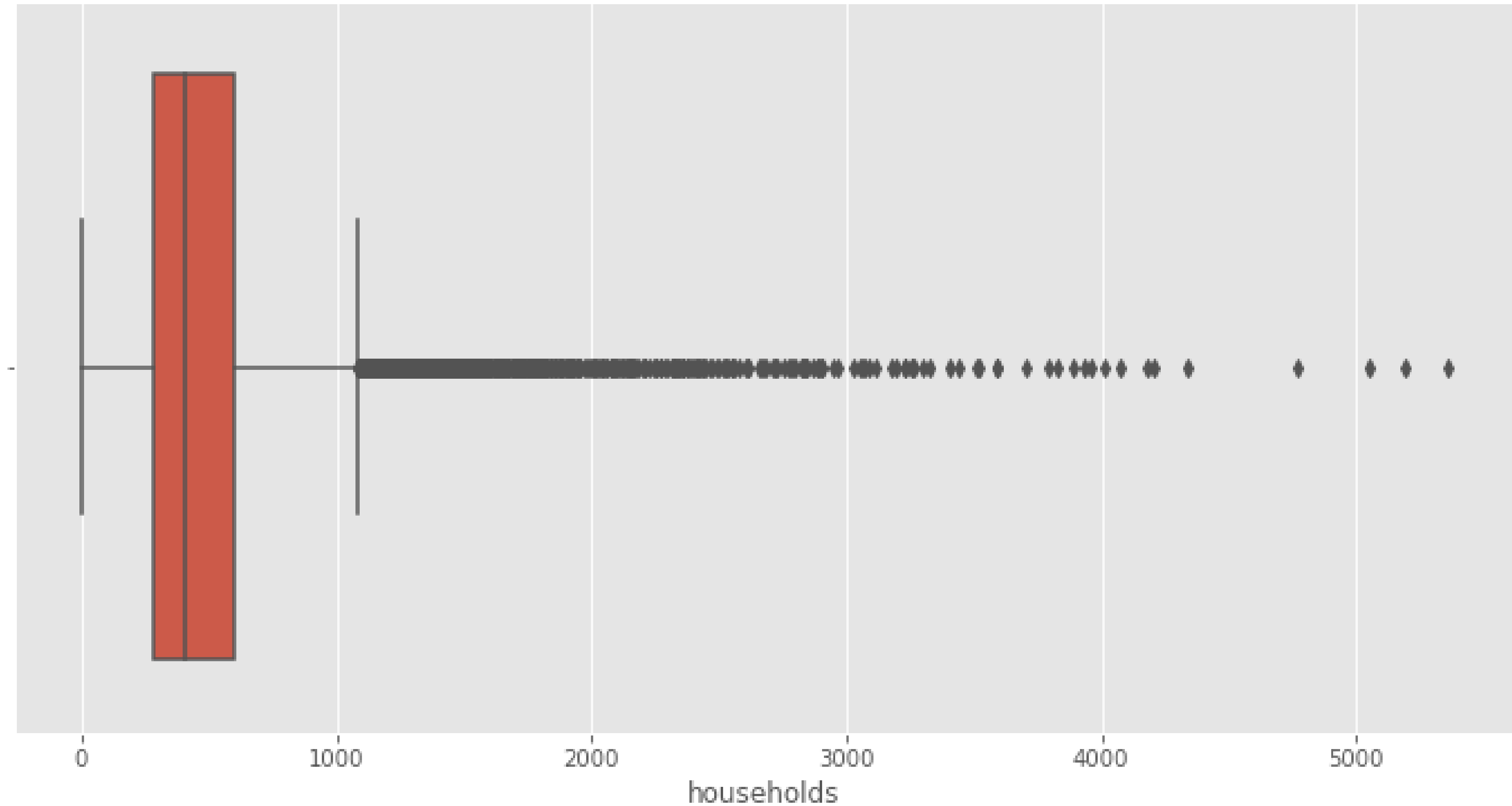

34
36
38
40
42
latitude

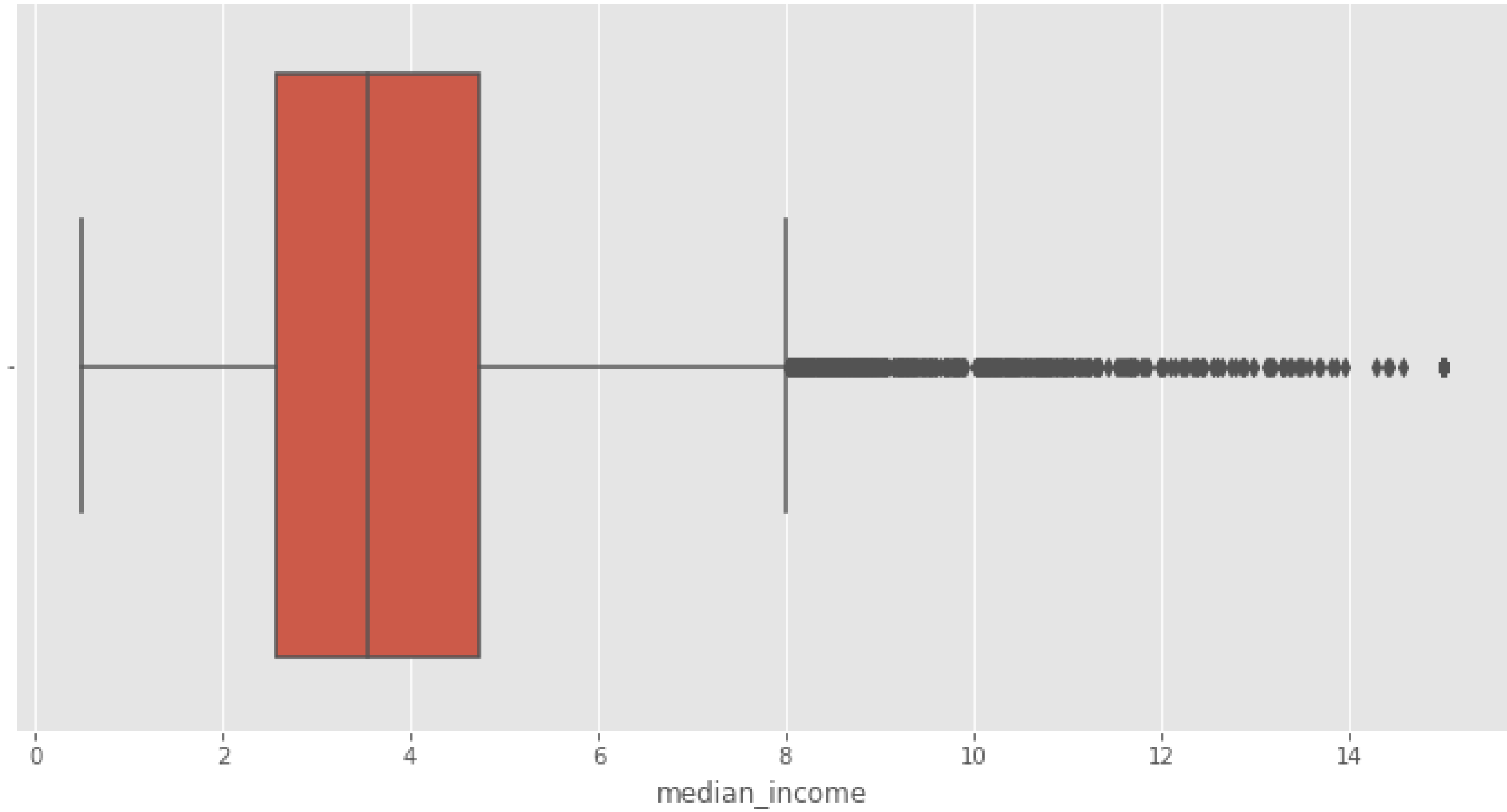

0
10
20
30
40
50
housing_median_age


0
5000
10000
15000
20000
25000
30000
35000
40000
total_rooms

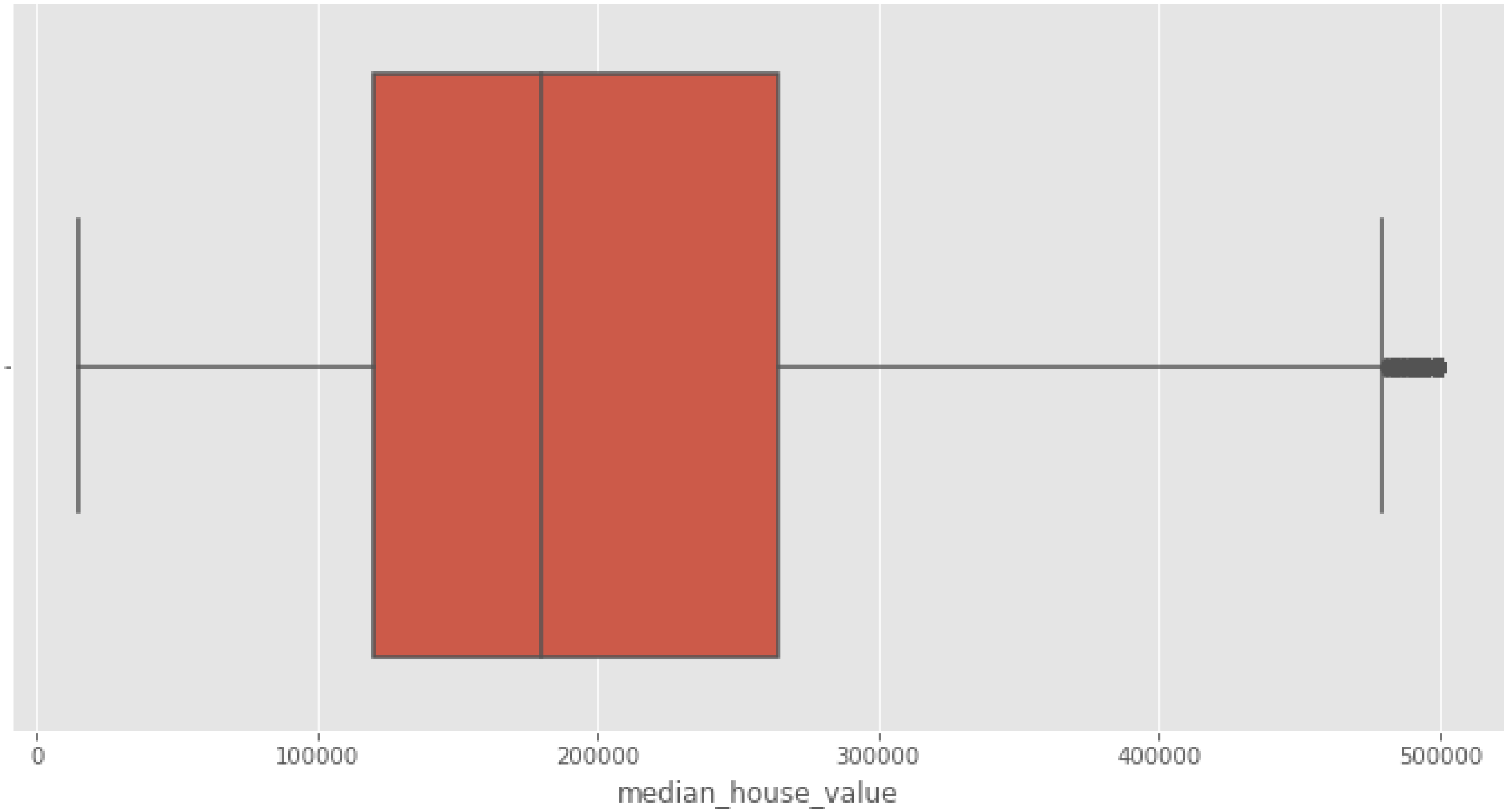

0
1000
2000
3000
4000
5000
6000
total_bedrooms

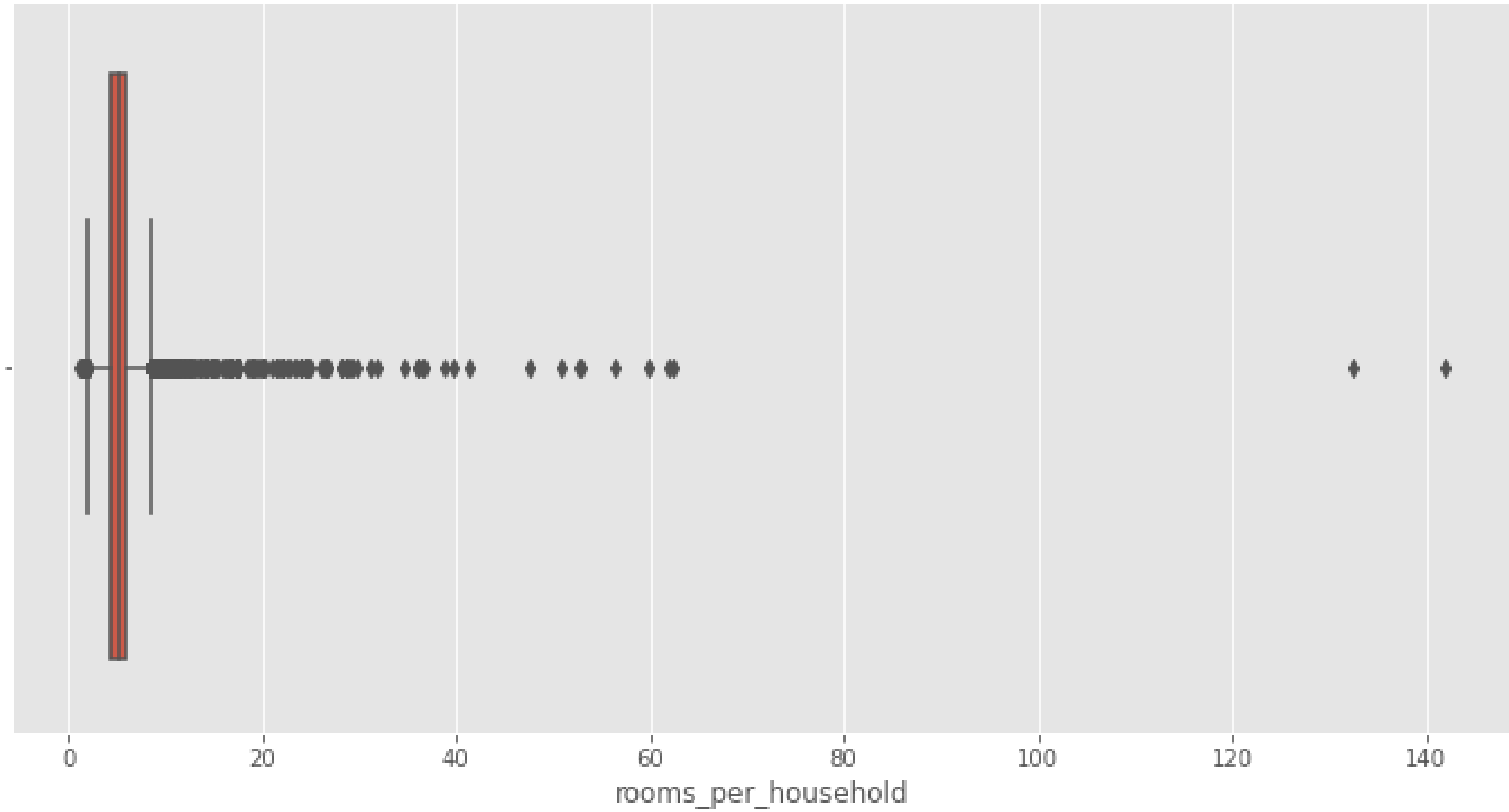

0
5000
10000
15000
20000
25000
30000
35000
population


0
1000
2000
3000
4000
5000
households

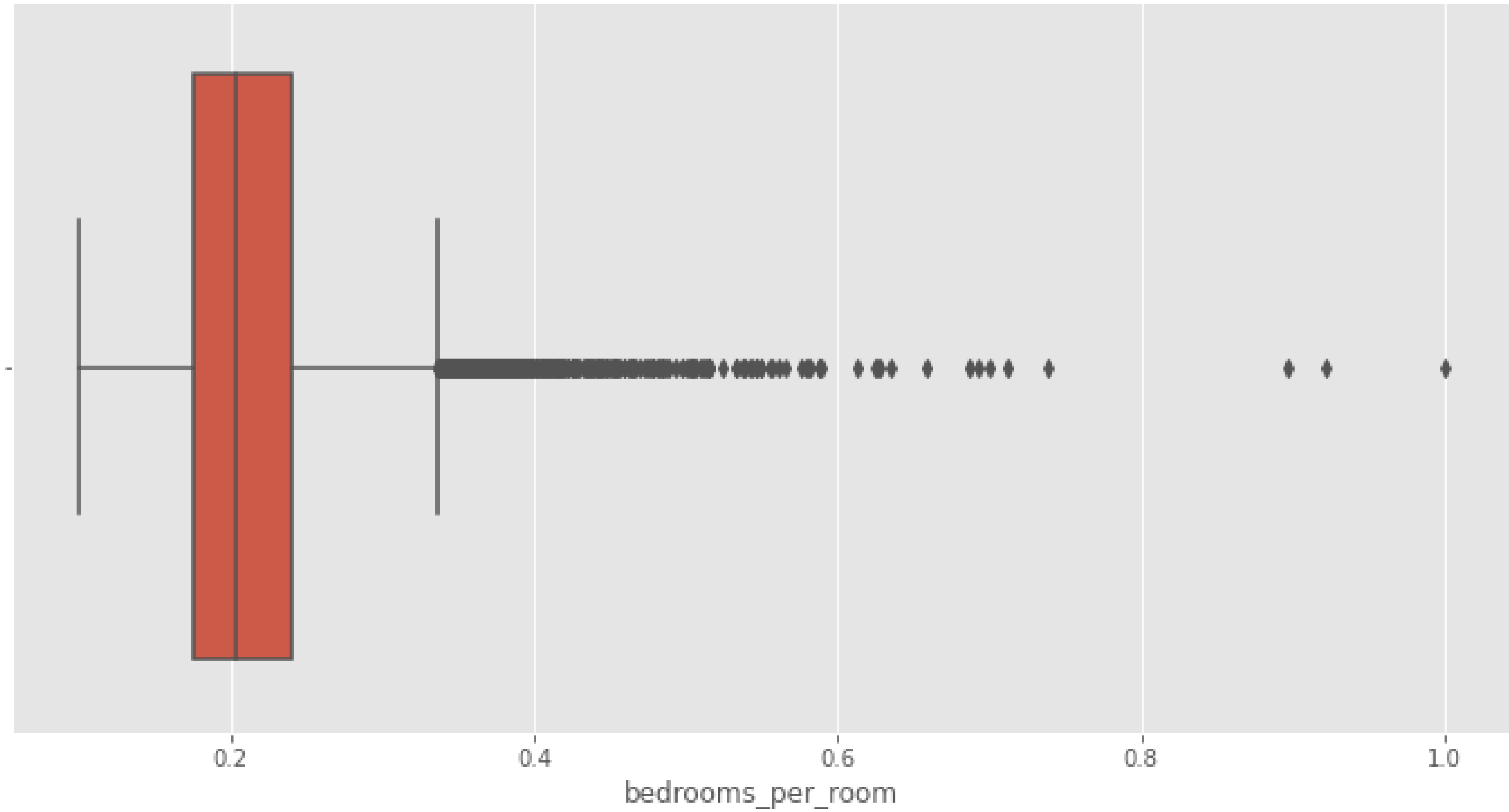


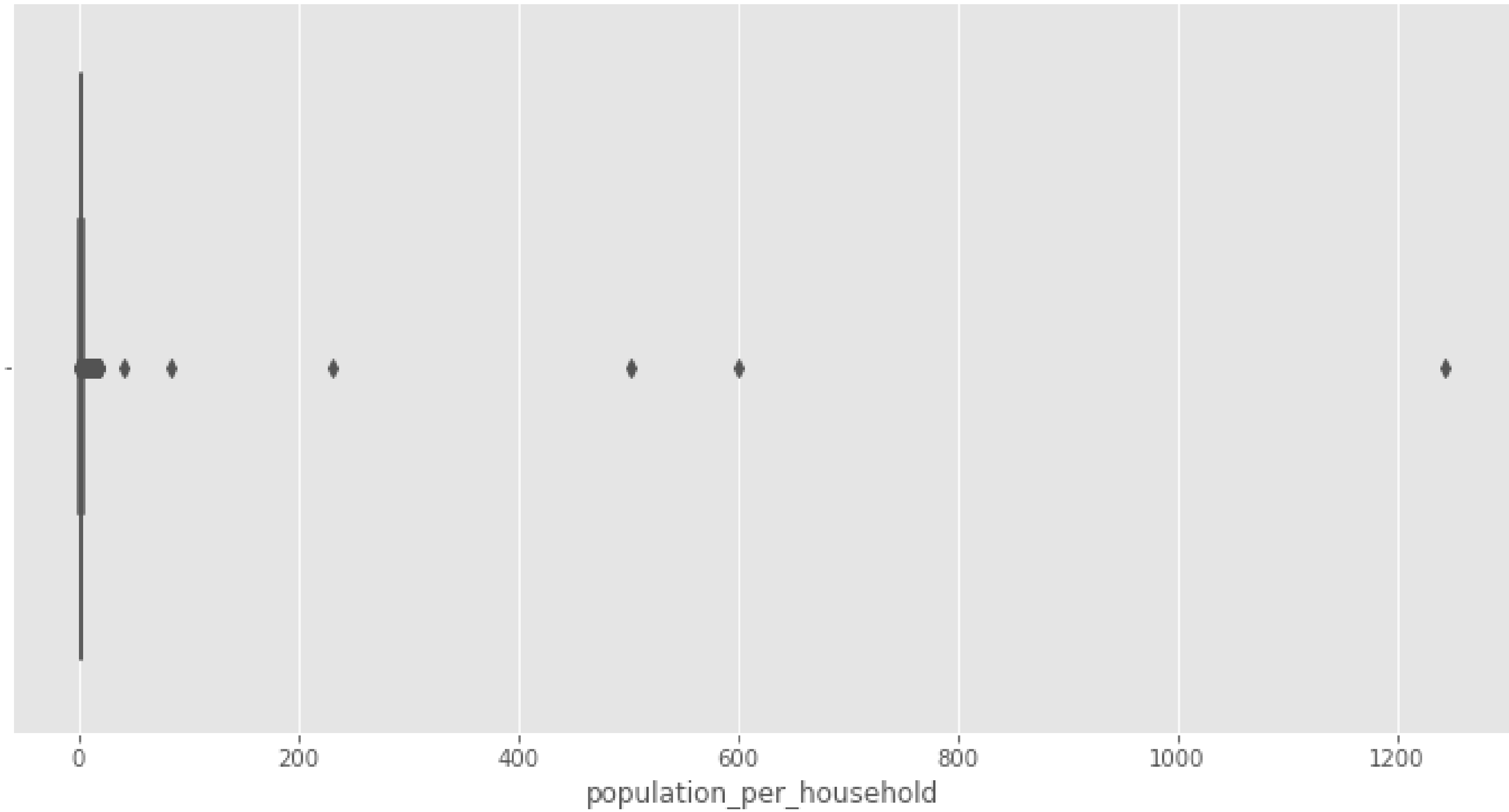
0
2
4
6
8
10
12
14
median_income


0
100000
200000
300000
400000
500000
median_house_value



0
20
40
60
80
100
120
140
rooms_per_household


0.2
0.4
0.6
0.8
1.0
bedrooms_per_room



In [22]:
```
from sklearn.base import BaseEstimator,TransformerMixin

class RemoveOutliers(BaseEstimator,TransformerMixin):
    """This class removes outliers from data.
    Note: Outlier values are hard coded
    """
    def fit (self,X,y=None):
        return self

    def transform(self,X,y=None):
        X=X[(X['median_house_value']!=500001) | (X['median_income']>=2)].reset_index(drop=True)
        X=X[X['median_income']<=11].reset_index(drop=True)
        X=X[(X['median_house_value']!=350000) | (X['median_income']>=1.5)].reset_index(drop=True)
        X=X[(X['median_house_value']!=450000) | (X['median_income']>=2)].reset_index(drop=True)
        X=X[(X['median_house_value']>=350000) | (X['median_income']<=9.5)].reset_index(drop=True)
        X=X[X['population']<=9000]
```

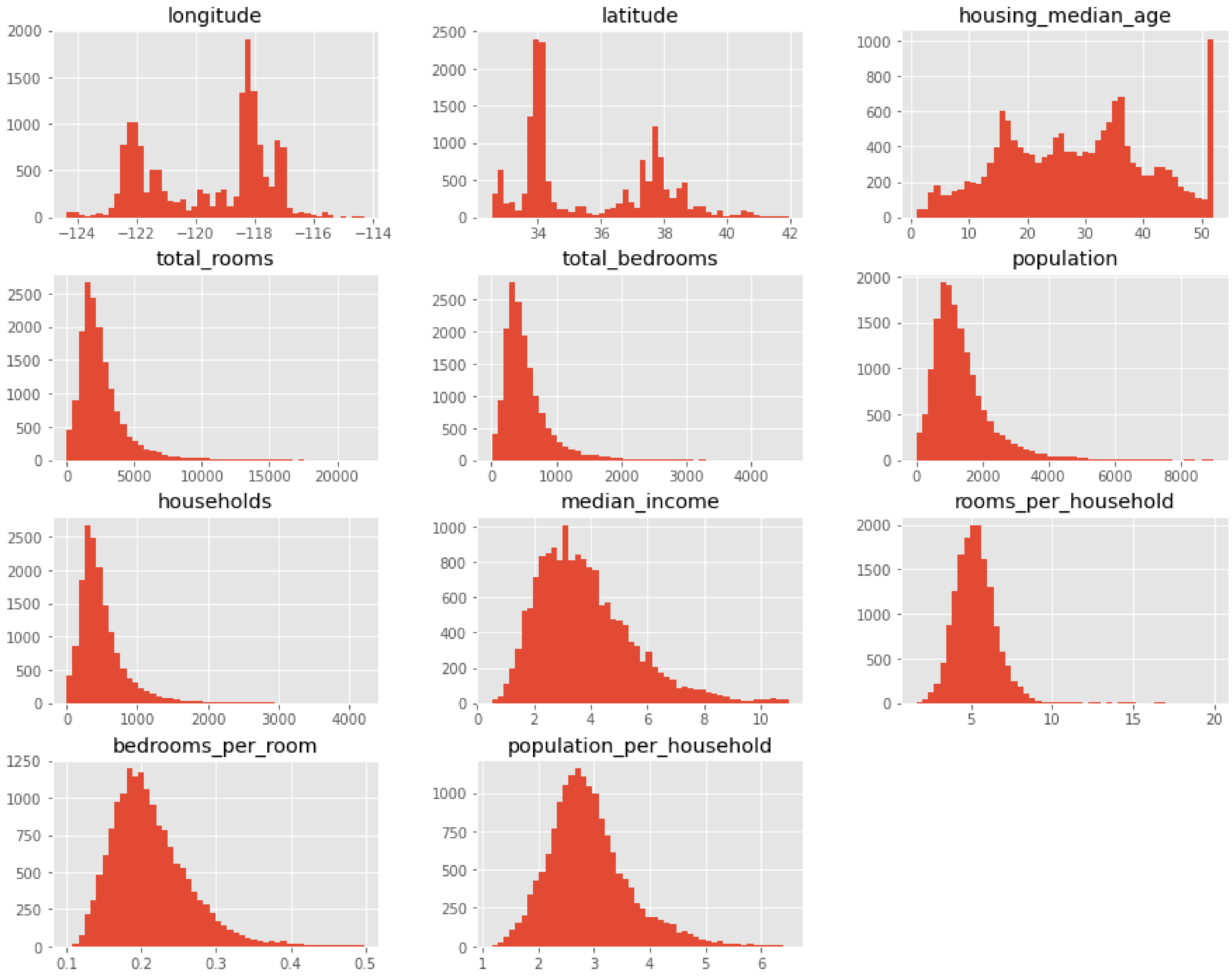
```python
        X=X[(X['population_per_household']>=1.15) & (X['population_per_household']<=6.5)]
        X=X[X['rooms_per_household']<20]
        X=X[X['bedrooms_per_room']<0.5].reset_index(drop=True)
        return X

data_copy=RemoveOutliers().fit_transform(data_copy)

data_labels=data_copy['median_house_value']
data_copy=data_copy.drop('median_house_value',axis=1)
```

In [23]:

```python
# data_copy.head()
# data_copy.isnull().value_counts()
data_copy.info()
data_copy.hist(bins=50,figsize=(15,12))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16005 entries, 0 to 16004
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   longitude                 16005 non-null  float64
 1   latitude                  16005 non-null  float64
 2   housing_median_age        16005 non-null  float64
 3   total_rooms               16005 non-null  float64
 4   total_bedrooms            16005 non-null  float64
 5   population                16005 non-null  float64
 6   households                16005 non-null  float64
 7   median_income             16005 non-null  float64
 8   ocean_proximity           16005 non-null  object 
 9   rooms_per_household       16005 non-null  float64
 10  bedrooms_per_room         16005 non-null  float64
 11  population_per_household  16005 non-null  float64
dtypes: float64(11), object(1)
memory usage: 1.5+ MB
```

Out[23]:
```
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':'rooms_per_household'}>],
       [<AxesSubplot:title={'center':'bedrooms_per_room'}>,
        <AxesSubplot:title={'center':'population_per_household'}>,
        <AxesSubplot:>]], dtype=object)
```


longitude
latitude
housing_median_age
total_rooms
total_bedrooms
population
households
median_income
rooms_per_household
bedrooms_per_room
population_per_household

6. Summary of steps taken

Steps undertaken for pre-processing the data:

- Data has been cleaned with no null values and outliers.
- We have further converted the categorical feature to numeric and scaled the data. ##### Further Analysis steps to be done:
- Remove skewness
- Can also use Get dummies to convert categorical feature of 'ocean_proximity'
- Check for Multi colinearity and scale the features further.