# califoenia

October 2, 2024

[20]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[21]:
```python
df = pd.read_csv('housing.csv')
df
```

[21]:
```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0        -122.23     37.88                41.0        880.0           129.0
1        -122.22     37.86                21.0       7099.0          1106.0
2        -122.24     37.85                52.0       1467.0           190.0
3        -122.25     37.85                52.0       1274.0           235.0
4        -122.25     37.85                52.0       1627.0           280.0
…             …         …                    …            …               …

20635    -121.09     39.48                25.0       1665.0           374.0
20636    -121.21     39.49                18.0        697.0           150.0
20637    -121.22     39.43                17.0       2254.0           485.0
20638    -121.32     39.43                18.0       1860.0           409.0
20639    -121.24     39.37                16.0       2785.0           616.0

       population  households  median_income
       median_house_value  \
0           322.0       126.0         8.3252          452600.0

1          2401.0      1138.0         8.3014          358500.0

2           496.0       177.0         7.2574          352100.0

3           558.0       219.0         5.6431          341300.0

4           565.0       259.0         3.8462          342200.0

…               …           …              …                 …

20635       845.0       330.0         1.5603           78100.0

20636       356.0       114.0         2.5568           77100.0

20637      1007.0       433.0         1.7000           92300.0

20638       741.0       349.0         1.8672           84700.0
```

```
20639    1387.0     530.0       2.3886           89400.0

      ocean_proximity
0           NEAR BAY
1           NEAR BAY
2           NEAR BAY
3           NEAR BAY
4           NEAR BAY
…              …
20635         INLAND
20636         INLAND
20637         INLAND
20638         INLAND
20639         INLAND

[20640 rows x 10 columns]
```

## 0.1 Task 1: Identifying Variable Types

[22]:
```
df.dtypes
```

[22]:
```
longitude          float64
latitude           float64
housing_median_age float64
total_rooms        float64
total_bedrooms     float64
population         float64
households         float64
median_income      float64
median_house_value float64
ocean_proximity     object
dtype: object
```

[23]:
```
df.info()
```

```
<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
```

[24]:
```
df.isnull().sum().sort_values(ascending=False)
```

[24]:
```
total_bedrooms     207
longitude            0
latitude             0
housing_median_age   0
total_rooms          0
population           0
households           0
median_income        0
median_house_value   0
ocean_proximity      0
dtype: int64
```

[25]:
```
numerical_columns = df.select_dtypes(np.number)
numerical_columns.sample(6)
```

[25]:
```
    longitude latitude housing_median_age total_rooms total_bedrooms \
4535  -118.21     34.03                44.0      1550.0          407.0
12750 -121.38     38.62                34.0      2352.0          610.0
8327  -118.30     33.94                36.0      2041.0          531.0
2709  -115.69     32.79                18.0      1564.0          340.0
15813 -122.42     37.76                52.0      4407.0         1192.0
7720  -118.11     33.94                37.0      1434.0          262.0

      population households median_income
      median_house_value
4535     1718.0      403.0       2.5268        141100.0

12750    1127.0      592.0       2.2000        116500.0

8327     1390.0      464.0       2.0114         99300.0

2709     1161.0      343.0       2.1792         55200.0

15813    2280.0     1076.0       3.3937        270000.0
```

```
7720       786.0      256.0       4.4375           244900.0
```

[26]:
```python
categoric = df.select_dtypes(object)
categoric.sample(5)
```

[26]:
```
    ocean_proximity
2416         INLAND
4930 <1H OCEAN 20229
NEAR OCEAN
5250 <1H OCEAN 8114
NEAR OCEAN
```

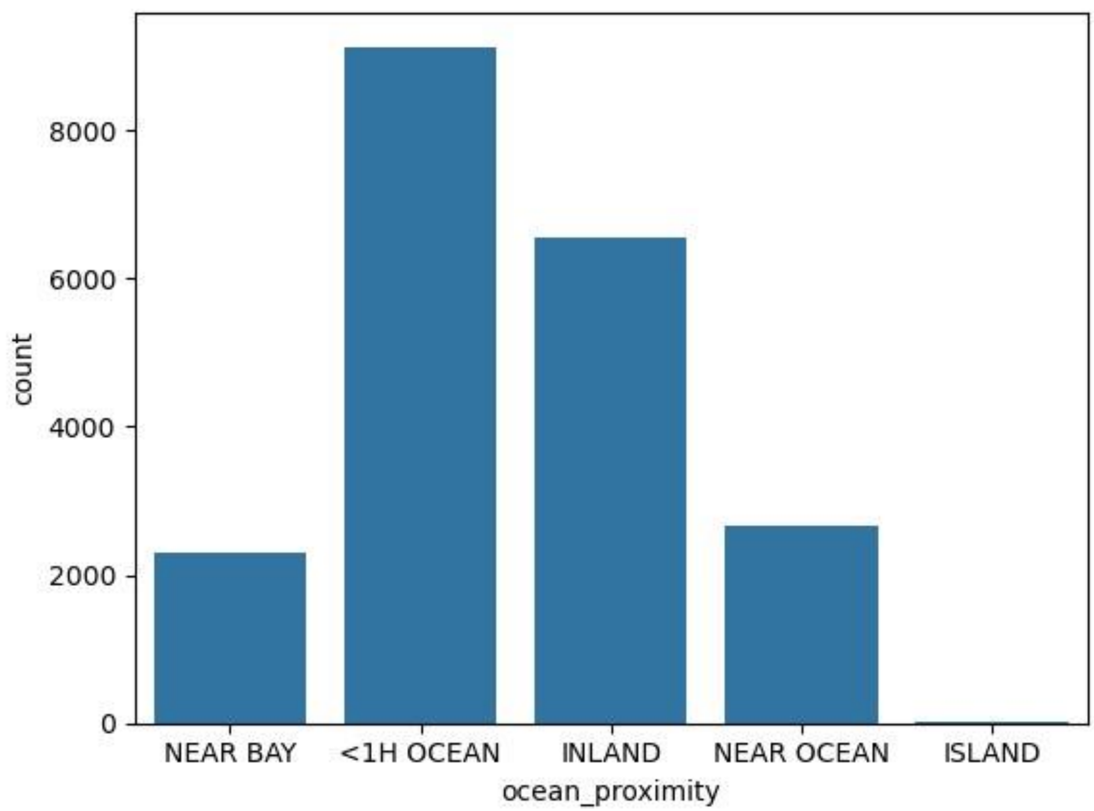
[27]:
```python
categoric.value_counts()
```

[27]:
```
ocean_proximity

<1H OCEAN       9136
INLAND          6551
NEAR OCEAN      2658
NEAR BAY        2290
ISLAND             5
Name: count, dtype: int64
```

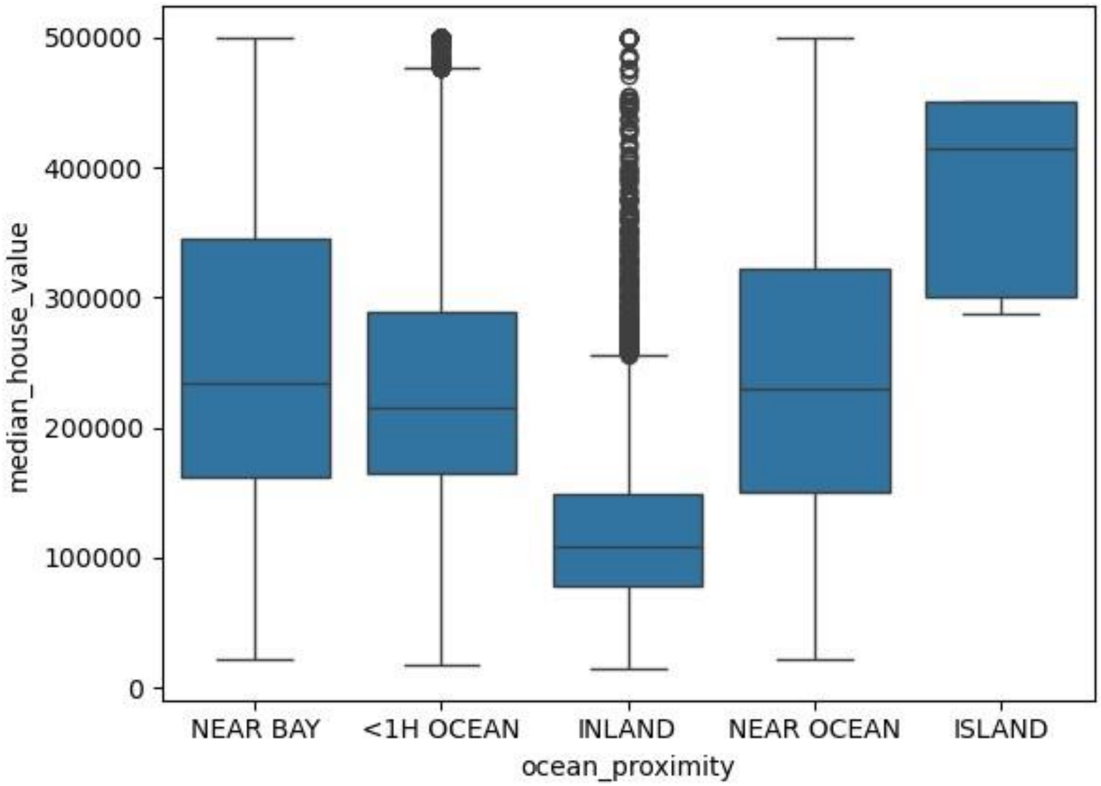
[28]:
```python
sns.countplot(x='ocean_proximity', data=df)
```

[28]:
```
<Axes: xlabel='ocean_proximity', ylabel='count'>
```



[29]:
```
sns.boxplot(x='ocean_proximity', y='median_house_value', data=df)
```

[29]: <Axes: xlabel='ocean_proximity', ylabel='median_house_value'>




## 0.2 Feature Scaling

### 0.2.1 Standardization

Standardization transforms the features to have a mean of 0 and a standard deviation of 1.

[30]:
```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(numerical_columns)

standardized_df = pd.DataFrame(scaled_data, columns=numerical_columns.columns)
```

[31]:
```
standardized_df
```

```
from sklearn.preprocessing import StandardScaler
```

[31]:
```
    longitude latitude housing_median_age total_rooms total_bedrooms \
```

| | | | | | |
|---|---|---|---|---|---|
| 0 | -1.327835 | 1.052548 | 0.982143 | -0.804819 | -0.970325 |
| 1 | -1.322844 | 1.043185 | -0.607019 | 2.045890 | 1.348276 |
| 2 | -1.332827 | 1.038503 | 1.856182 | -0.535746 | -0.825561 |
| 3 | -1.337818 | 1.038503 | 1.856182 | -0.624215 | -0.718768 |
| 4 | -1.337818 | 1.038503 | 1.856182 | -0.462404 | -0.611974 |
| … | … | … | … | … | … |
| 20635 | -0.758826 | 1.801647 | -0.289187 | -0.444985 | -0.388895 |
| 20636 | -0.818722 | 1.806329 | -0.845393 | -0.888704 | -0.920488 |
| 20637 | -0.823713 | 1.778237 | -0.924851 | -0.174995 | -0.125472 |
| 20638 | -0.873626 | 1.778237 | -0.845393 | -0.355600 | -0.305834 |
| 20639 | -0.833696 | 1.750146 | -1.004309 | 0.068408 | 0.185416 |

| | population | households | median_income | median_house_value |
|---|---|---|---|---|
| 0 | -0.974429 | -0.977033 | 2.344766 | 2.129631 |
| 1 | 0.861439 | 1.669961 | 2.332238 | 1.314156 |
| 2 | -0.820777 | -0.843637 | 1.782699 | 1.258693 |
| 3 | -0.766028 | -0.733781 | 0.932968 | 1.165100 |
| 4 | -0.759847 | -0.629157 | -0.012881 | 1.172900 |
| … | … | … | … | … |
| 20635 | -0.512592 | -0.443449 | -1.216128 | -1.115804 |
| 20636 | -0.944405 | -1.008420 | -0.691593 | -1.124470 |
| 20637 | -0.369537 | -0.174042 | -1.142593 | -0.992746 |
| 20638 | -0.604429 | -0.393753 | -1.054583 | -1.058608 |
| 20639 | -0.033977 | 0.079672 | -0.780129 | -1.017878 |

[20640 rows x 9 columns]

### 0.2.2 Normalization

Normalization scales the data between 0 and 1. This is useful when the scale of features differs, and you want to maintain relative differences.

[32]:
```python
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaled_data = scaler.fit_transform(numerical_columns)

normalized_df = pd.DataFrame(scaled_data, columns=numerical_columns.columns)

normalized_df
```

[32]:

| | longitude | latitude | housing_median_age | total_rooms | total_bedrooms \ |
|---|---|---|---|---|---|
| 0 | 0.211155 | 0.567481 | 0.784314 | 0.022331 | 0.019863 |
| 1 | 0.212151 | 0.565356 | 0.392157 | 0.180503 | 0.171477 |
| 2 | 0.210159 | 0.564293 | 1.000000 | 0.037260 | 0.029330 |
| 3 | 0.209163 | 0.564293 | 1.000000 | 0.032352 | 0.036313 |
| 4 | 0.209163 | 0.564293 | 1.000000 | 0.041330 | 0.043296 |
| … | … | … | … | … | … |
| 20635 | 0.324701 | 0.737513 | 0.470588 | 0.042296 | 0.057883 |
| 20636 | 0.312749 | 0.738576 | 0.333333 | 0.017676 | 0.023122 |
| 20637 | 0.311753 | 0.732200 | 0.313725 | 0.057277 | 0.075109 |
| 20638 | 0.301793 | 0.732200 | 0.333333 | 0.047256 | 0.063315 |
| 20639 | 0.309761 | 0.725824 | 0.294118 | 0.070782 | 0.095438 |

| | population | households | median_income | median_house_value |
|---|---|---|---|---|
| 0 | 0.008941 | 0.020556 | 0.539668 | 0.902266 |
| 1 | 0.067210 | 0.186976 | 0.538027 | 0.708247 |
| 2 | 0.013818 | 0.028943 | 0.466028 | 0.695051 |
| 3 | 0.015555 | 0.035849 | 0.354699 | 0.672783 |
| 4 | 0.015752 | 0.042427 | 0.230776 | 0.674638 |
| … | … | … | … | … |
| 20635 | 0.023599 | 0.054103 | 0.073130 | 0.130105 |
| 20636 | 0.009894 | 0.018582 | 0.141853 | 0.128043 |
| 20637 | 0.028140 | 0.071041 | 0.082764 | 0.159383 |
| 20638 | 0.020684 | 0.057227 | 0.094295 | 0.143713 |

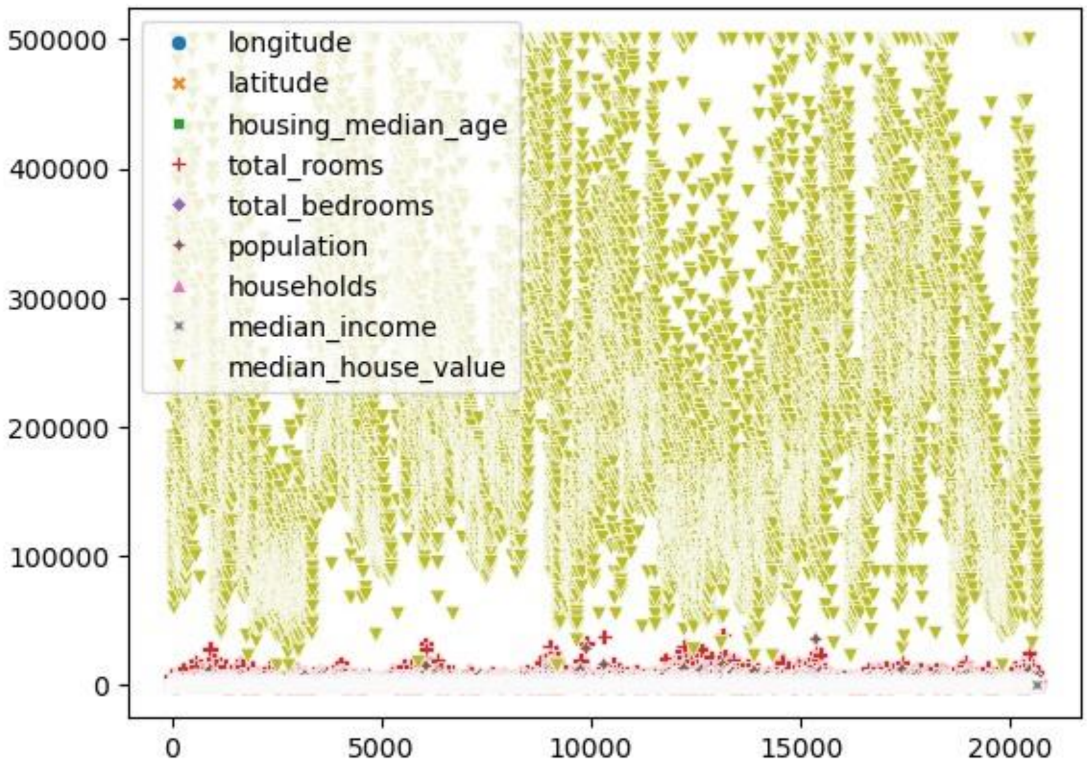
```
20639   0.038790   0.086992     0.130253         0.153403

[20640 rows x 9 columns]
```

### 0.2.3 Impact of Scaling

[33]:
```
# Before Scaling
sns.scatterplot(data=numerical_columns)
```

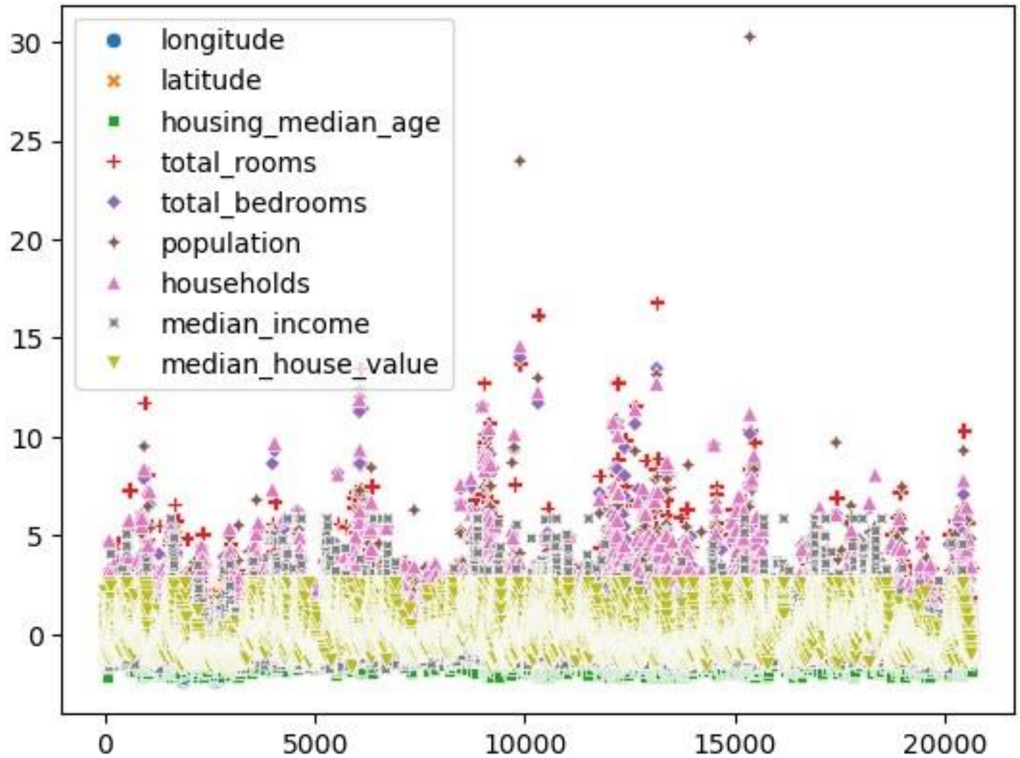
[33]: <Axes: >




[34]:
```
# After Scaling (Standardization)
sns.scatterplot(data=standardized_df)
```

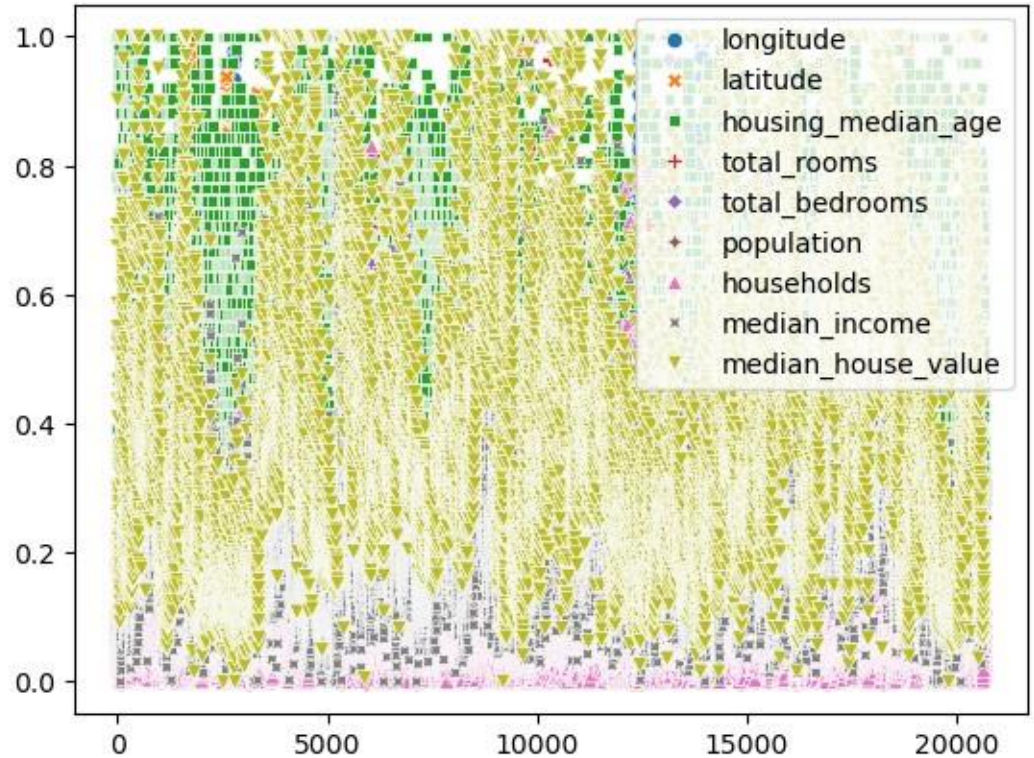
[34]: <Axes: >

```
c:\Users\Abuzar\miniconda3\envs\ML\lib\sitepackages\IPython\core\pylab
tools.py:152: UserWarning: Creating legend with loc="best" can be slow
with large amounts of data. fig.canvas.print_figure(bytes_io, **kw)
```




[35]:
```
# After Normalization
sns.scatterplot(data=normalized_df)
```

[35]: <Axes: >



1.0
0.8
0.6
0.4
0.2
0.0
0
5000
10000
15000
20000
longitude
latitude
housing_median_age
total_rooms
total_bedrooms
population
households
median_income
median_house_value


[ ]: