ug2x7rii4

April 16, 2025

Califonia House Price Dataset:

[1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```python
df = pd.read_csv('housing.csv.zip')
df
```

[2]:
```
       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]
```

[3]:

```python
df.head()
```

[3]:

```
   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

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
```

[4]:

```python
df.tail()
```

[4]:

```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
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  \
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
20635          INLAND
20636          INLAND
20637          INLAND
20638          INLAND
20639          INLAND
```

[5]:
```
df.shape
```

[5]:
```
(20640, 10)
```

[6]:
```
df.describe()
```

[6]:
```
          longitude      latitude  housing_median_age   total_rooms  \
count  20640.000000  20640.000000        20640.000000  20640.000000
mean    -119.569704     35.631861           28.639486   2635.763081
std        2.003532      2.135952           12.585558   2181.615252
min     -124.350000     32.540000            1.000000      2.000000
25%     -121.800000     33.930000           18.000000   1447.750000
50%     -118.490000     34.260000           29.000000   2127.000000
75%     -118.010000     37.710000           37.000000   3148.000000
max     -114.310000     41.950000           52.000000  39320.000000

       total_bedrooms    population    households  median_income  \
count    20433.000000  20640.000000  20640.000000   20640.000000
mean       537.870553   1425.476744    499.539680       3.870671
std        421.385070   1132.462122    382.329753       1.899822
min          1.000000      3.000000      1.000000       0.499900
25%        296.000000    787.000000    280.000000       2.563400
50%        435.000000   1166.000000    409.000000       3.534800
75%        647.000000   1725.000000    605.000000       4.743250
max       6445.000000  35682.000000   6082.000000      15.000100

       median_house_value
count        20640.000000
mean        206855.816909
std         115395.615874
min          14999.000000
25%         119600.000000
50%         179700.000000
75%         264725.000000
max         500001.000000
```

[7]:
```
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
```

[8]:

```
df.isnull().sum()
```

[8]:

```
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
```

[9]:

```
df['total_bedrooms'].fillna(df['total_bedrooms'].mean(), inplace=True)
df.isnull().sum()
```

[9]:

```
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
```

[10]:

```
n = df.select_dtypes(exclude='object')
n
```

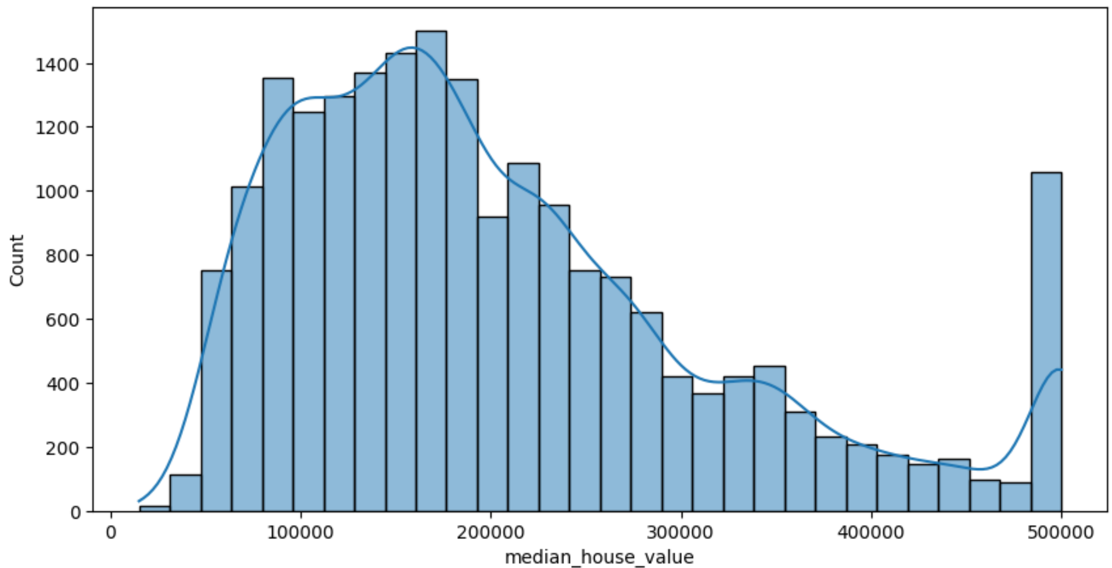
[10]:
```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0        -122.23     37.88                41.0        880.0           129.0
1        -122.22     37.86                21.0       7099.0          1106.0
2        -122.24     37.85                52.0       1467.0           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

[20640 rows x 9 columns]
```

[11]:
```python
c = df.select_dtypes(include='object')
c
```

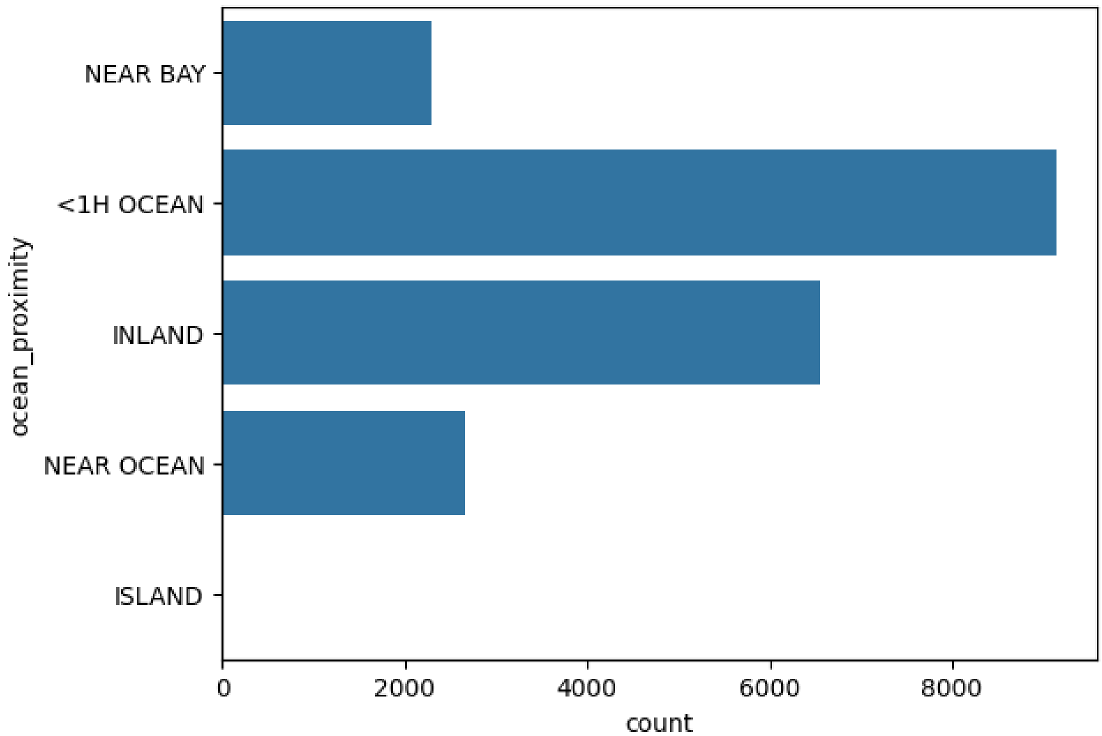
[11]:
```
      ocean_proximity
0            NEAR BAY
1            NEAR BAY
2            NEAR BAY
3            NEAR BAY
4            NEAR BAY
…                 …
20635          INLAND
20636          INLAND
20637          INLAND
20638          INLAND
20639          INLAND

[20640 rows x 1 columns]
```

UNIVARIATE ANALYSIS

[12]:
```
plt.figure(figsize=(10,5))
sns.histplot(df['median_house_value'], bins=30, kde=True)
plt.show()
```



[13]:
```
sns.countplot(df['ocean_proximity'])
plt.show()

df['ocean_proximity'].value_counts()
```



[13]:
```
ocean_proximity
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: count, dtype: int64
```

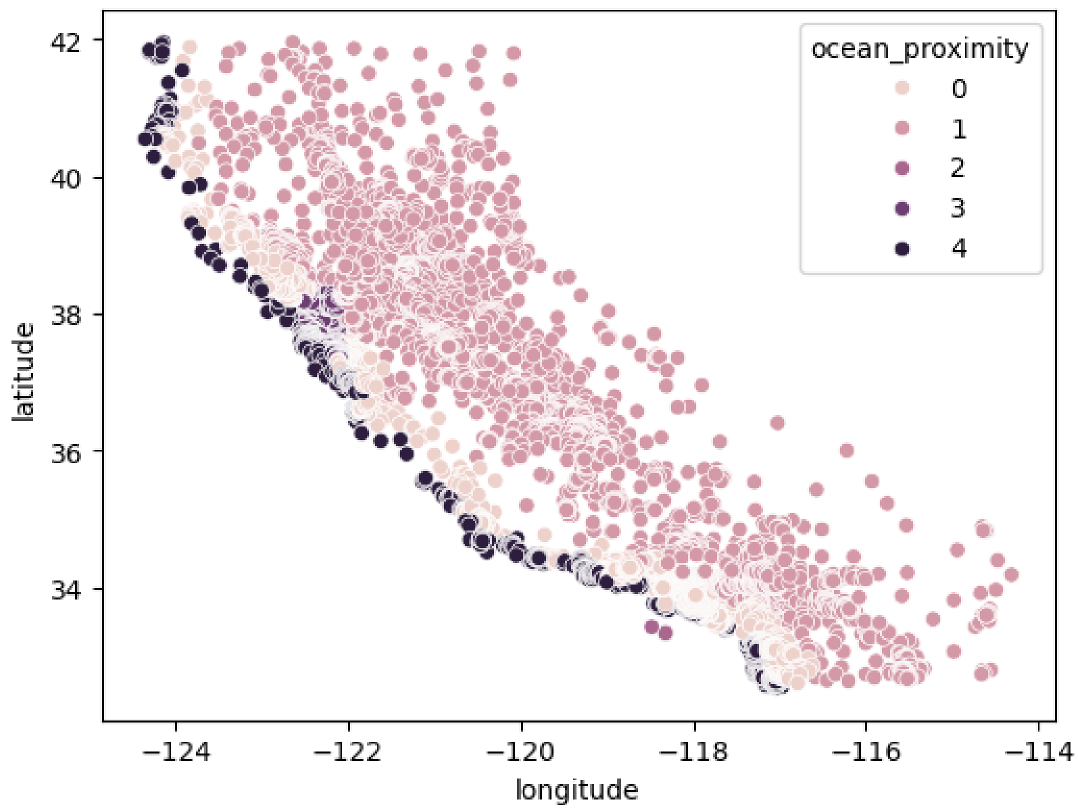
[53]:
```python
df = df[df['ocean_proximity'] != 'ISLAND']
```

[54]:
```python
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
df['ocean_proximity'] = label.fit_transform(df['ocean_proximity'])
df['ocean_proximity']
```

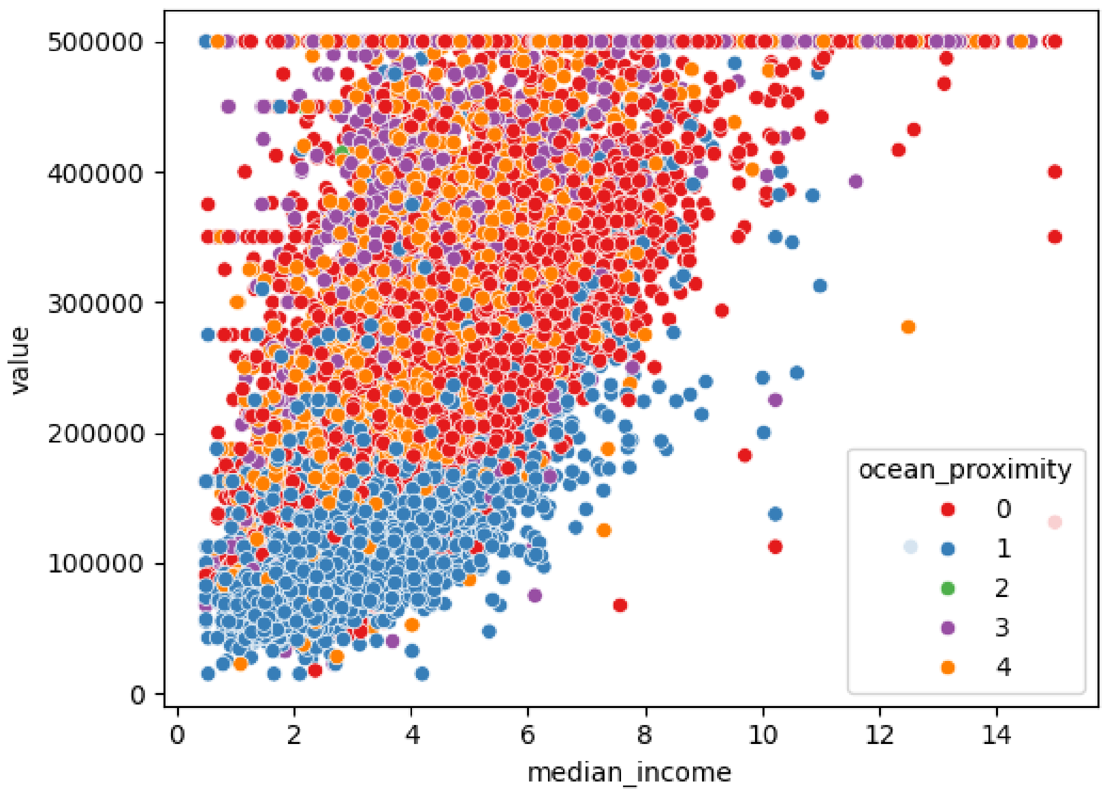
[54]:
```
0        3
1        3
2        3
3        3
4        3
        ..
20635    1
20636    1
```

```
20637    1
20638    1
20639    1
Name: ocean_proximity, Length: 20640, dtype: int64
```

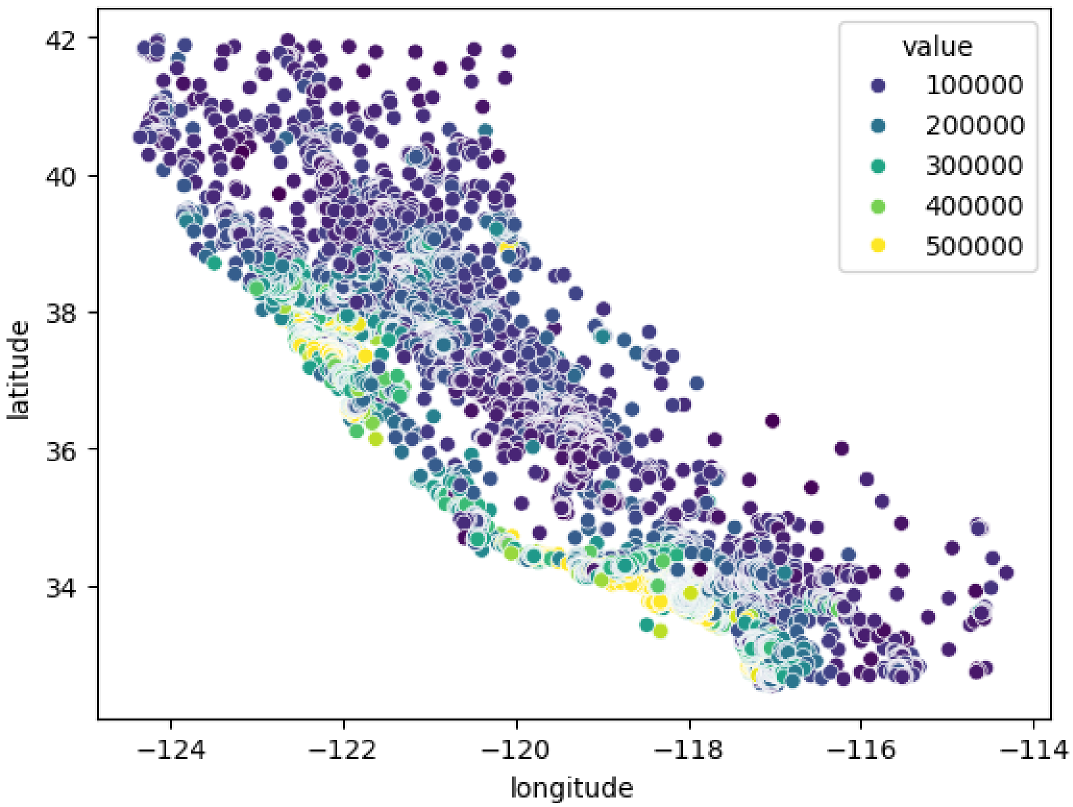
[55]:
```
sns.scatterplot(x=df['longitude'], y= df['latitude'], hue=df['ocean_proximity'])
plt.show()
```



[58]:
```
sns.scatterplot(x=df['median_income'], y= df['value'],␣
↪hue=df['ocean_proximity'], palette='Set1')
plt.show()
```



[59]:
```
sns.scatterplot(x=df['longitude'], y= df['latitude'], hue=df['value'], palette␣
↪= 'viridis')
plt.show()
```



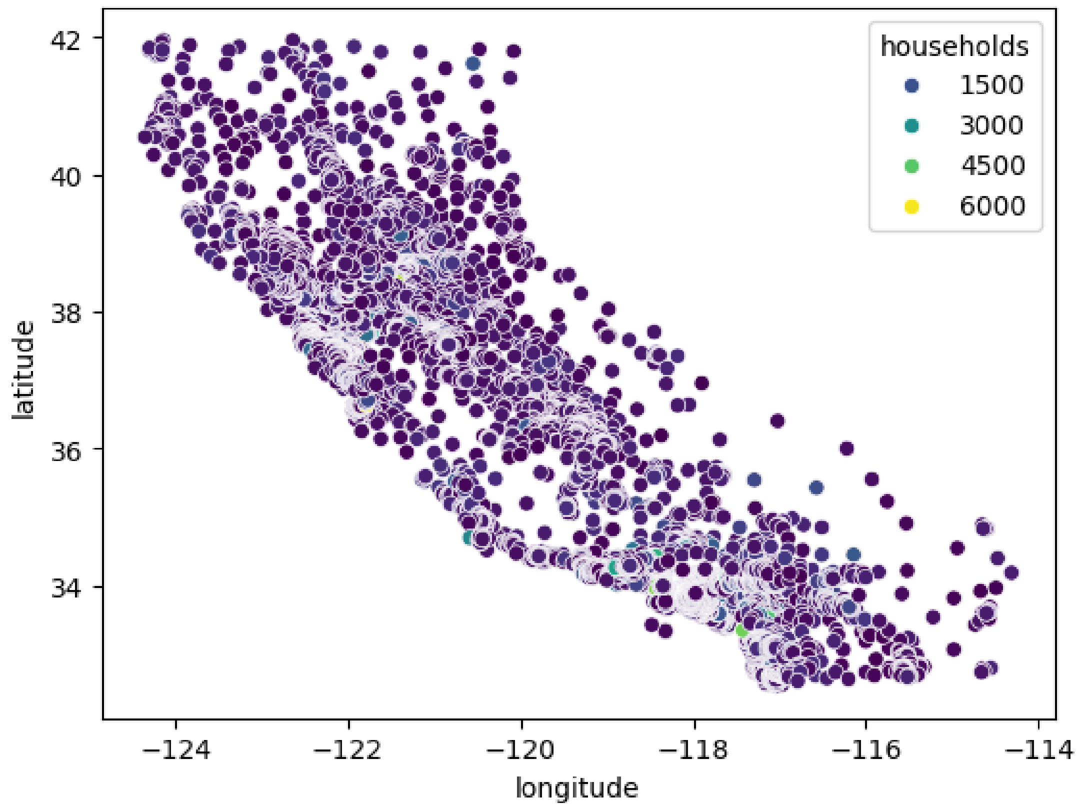
[19]:
```
c.columns
```

[19]:
```
Index(['ocean_proximity'], dtype='object')
```

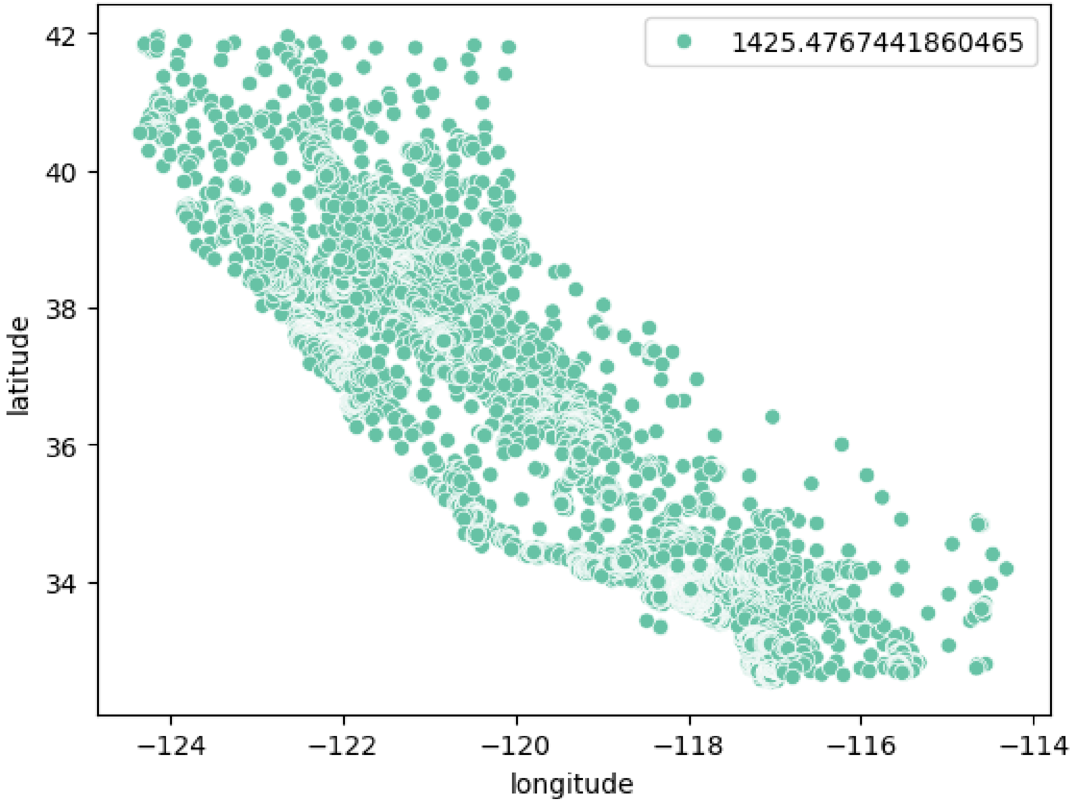
[20]:
```
n.columns
```

[20]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

[21]:
```
sns.scatterplot(x=df['longitude'], y= df['latitude'], hue=df['households'],␣
 ↪palette = 'viridis')
plt.show()
```



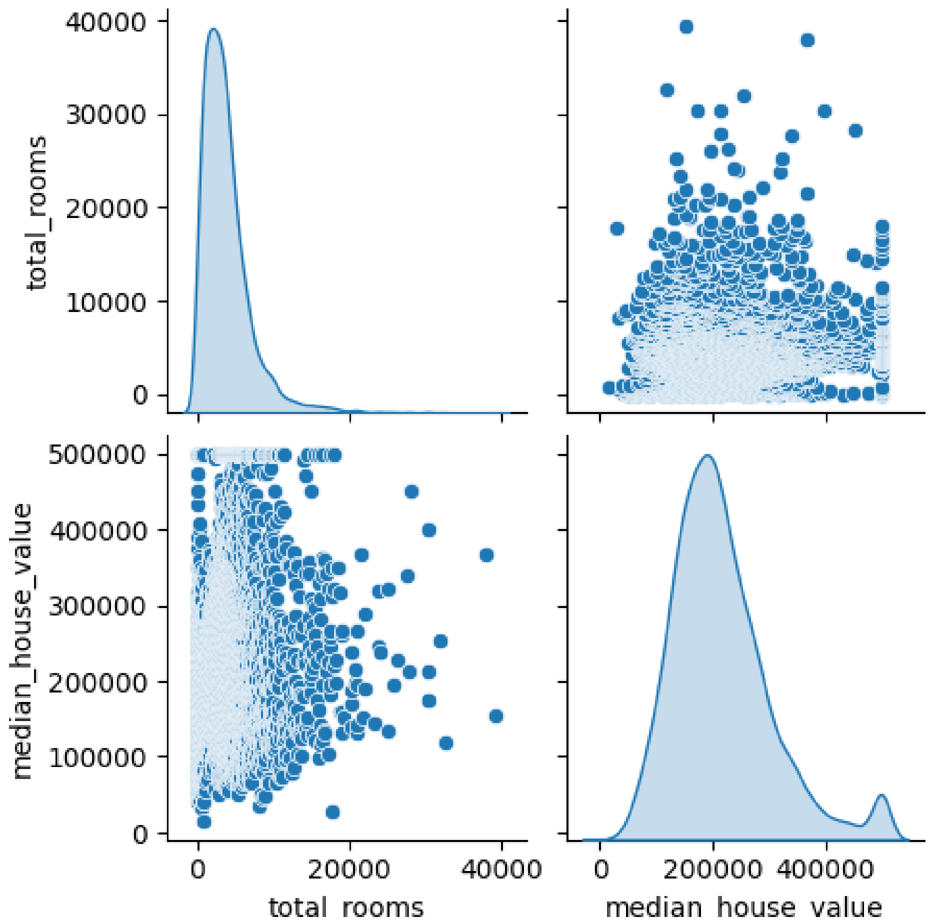
[22]:
```
sns.scatterplot(x=df['longitude'], y= df['latitude'], hue=df['population'].
    ↪mean(), palette = 'Set2')
plt.show()
```



[64]:
```
house_value1 = n.groupby('total_rooms')['median_house_value'].mean().
↪reset_index() # Use the actual column name 'median_house_value' instead of␣
↪the undefined variable 'value'.
house_value1
```

[64]:
```
      total_rooms  median_house_value
0             2.0            137500.0
1             6.0             55000.0
2             8.0            500001.0
3            11.0            162500.0
4            12.0             67500.0
…             …                   …
5921      30450.0            174300.0
5922      32054.0            253900.0
5923      32627.0            118800.0
5924      37937.0            366300.0
5925      39320.0            153700.0

[5926 rows x 2 columns]
```

[24]:
```
sns.pairplot(house_value1, diag_kind='kde')
plt.show()
```



[65]:
```
house_value2 = n.groupby('total_bedrooms')['median_house_value'].mean().
↪reset_index()
house_value2
```

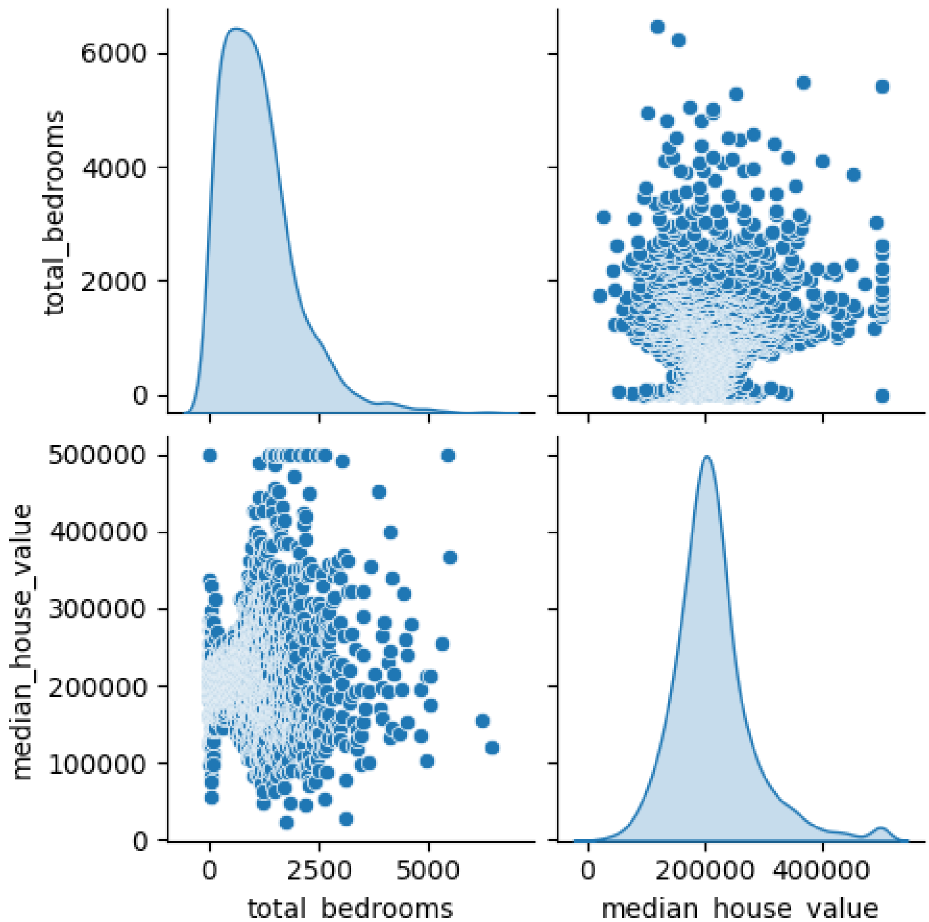
[65]:

	total_bedrooms	median_house_value
0	1.0	500001.000000
1	2.0	96250.000000
2	3.0	285000.000000
3	4.0	186428.714286
4	5.0	238550.000000
…	…	…
1919	5290.0	253900.000000
1920	5419.0	500001.000000
1921	5471.0	366300.000000

```
1922           6210.0       153700.000000
1923           6445.0       118800.000000

[1924 rows x 2 columns]
```

[26]:

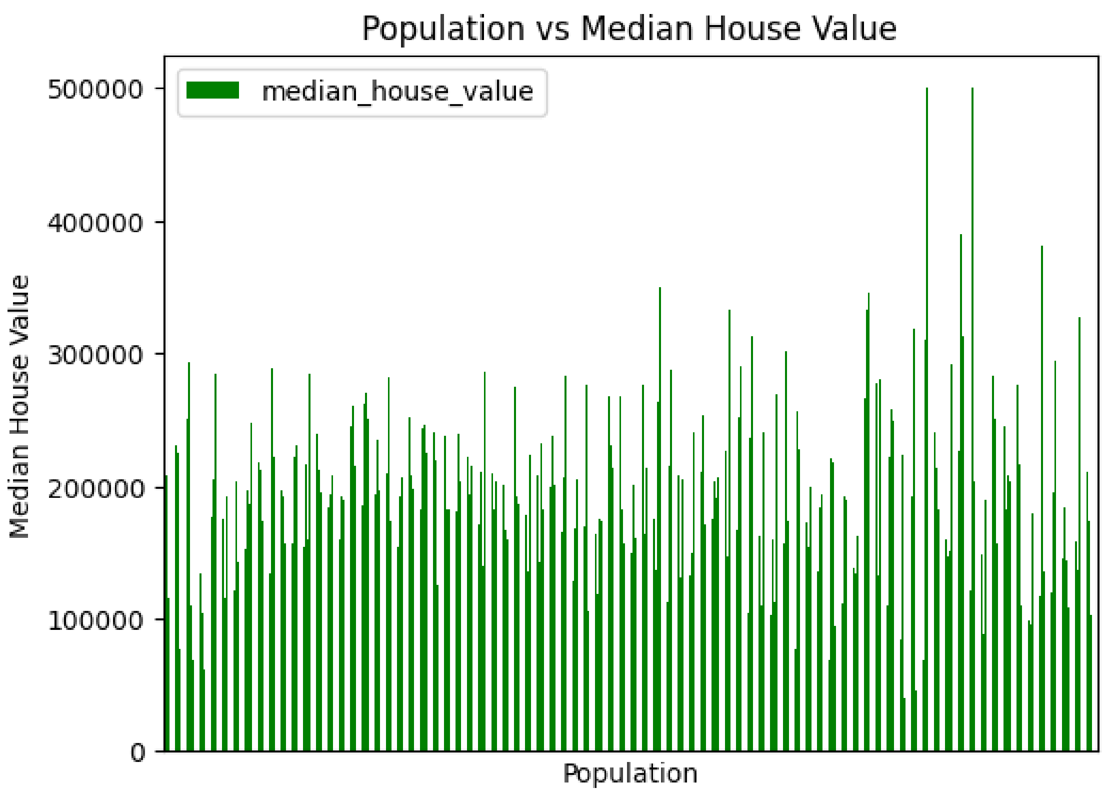
```
sns.pairplot(house_value2, diag_kind = 'kde', palette = 'viridis')
plt.show()
```

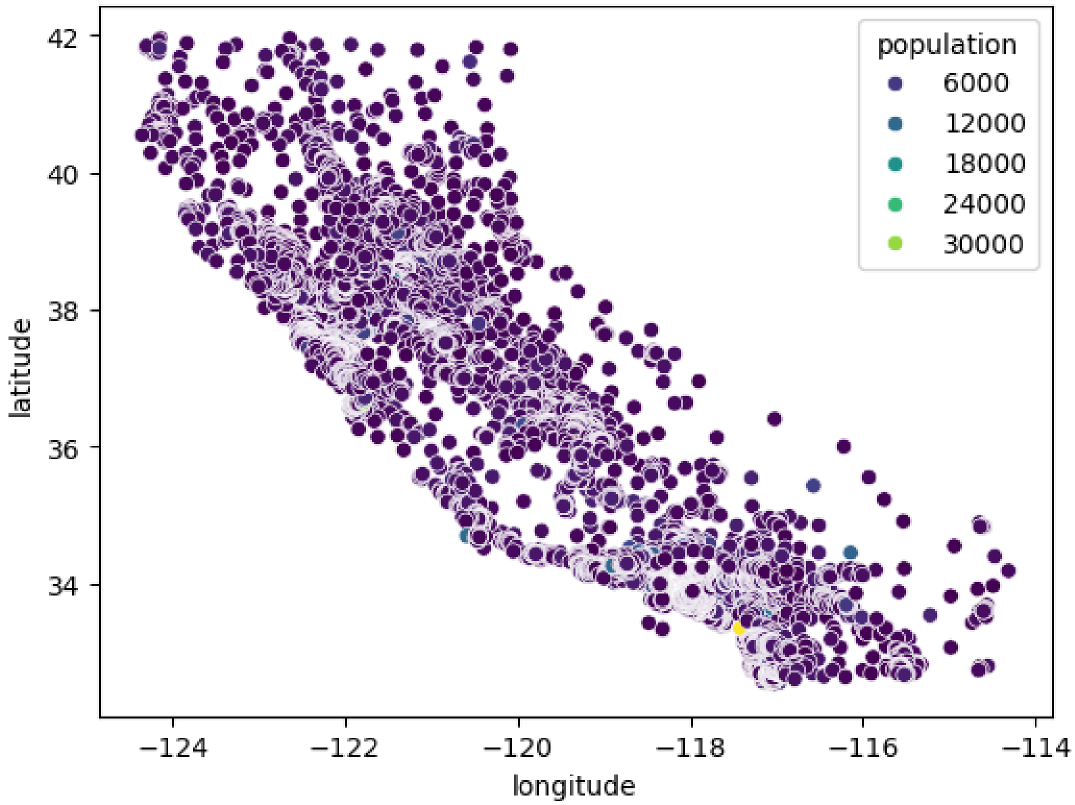


[66]:

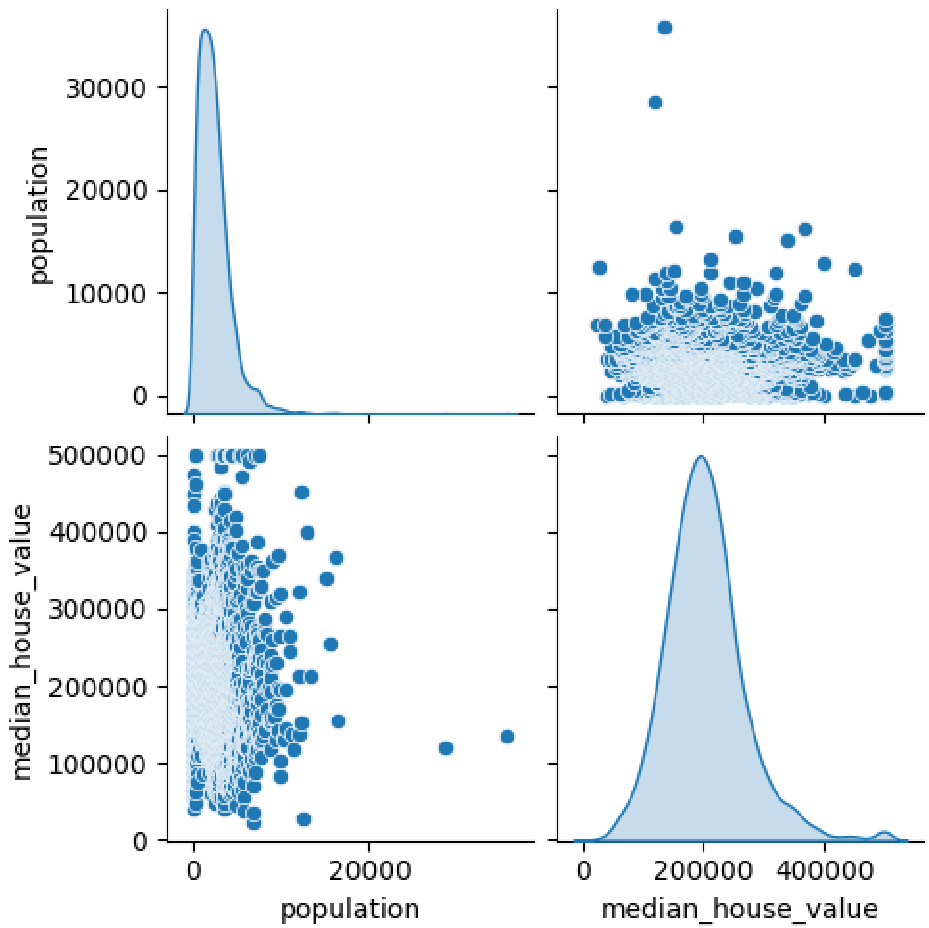
```
house_value3 = n.groupby('population')['median_house_value'].mean().
 ↪reset_index()
house_value3
population = df['population'].sample(n=50)
house_value3.plot(x='population', y='median_house_value', kind='bar',␣
 ↪color='green')
plt.xlabel('Population')
plt.ylabel('Median House Value')
plt.title('Population vs Median House Value')
```

```
plt.xticks([])
plt.show()

sns.scatterplot(x=df['longitude'], y= df['latitude'], hue=df['population'],␣
 ↪palette = 'viridis')
plt.show()
```





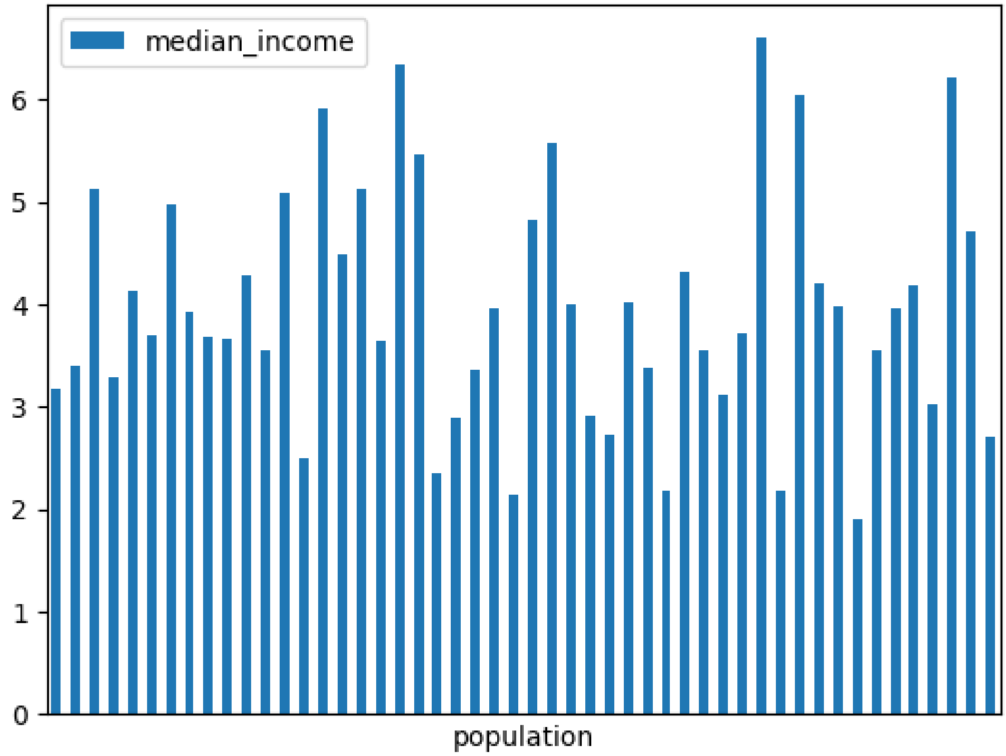
[28]:
```
sns.pairplot(house_value3, diag_kind = 'kde', palette = 'viridis')
plt.show()
```



[29]:
```
n.columns
```

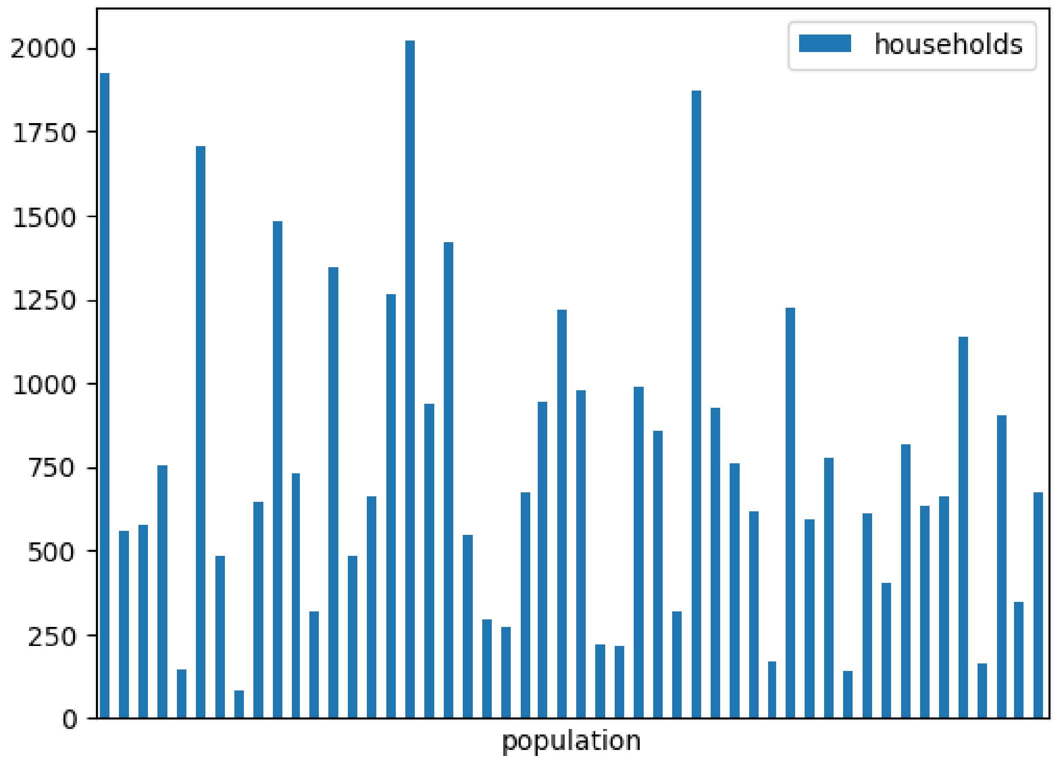
[29]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

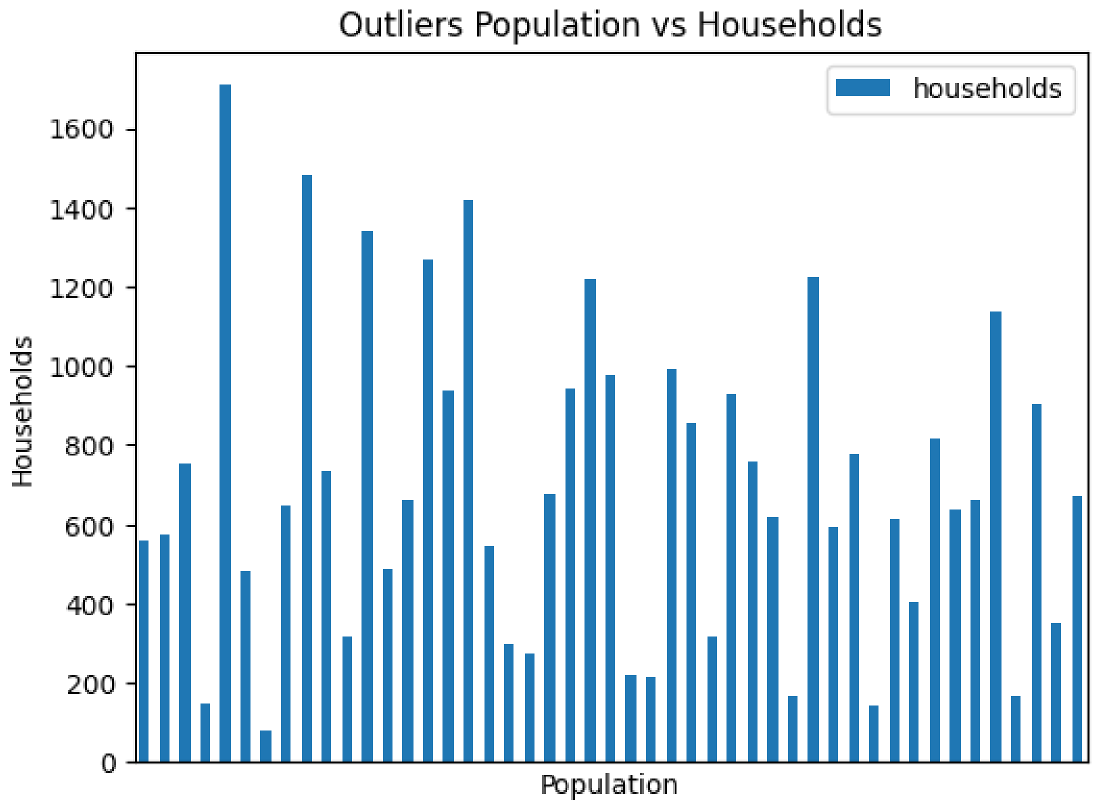
[30]:
```
pop_income = n.groupby('population')['median_income'].mean().reset_index()
pop_income

pop_income = pop_income.sample(n=50)
pop_income.plot(x='population', y='median_income', kind='bar')
plt.xticks([])
plt.show()
```

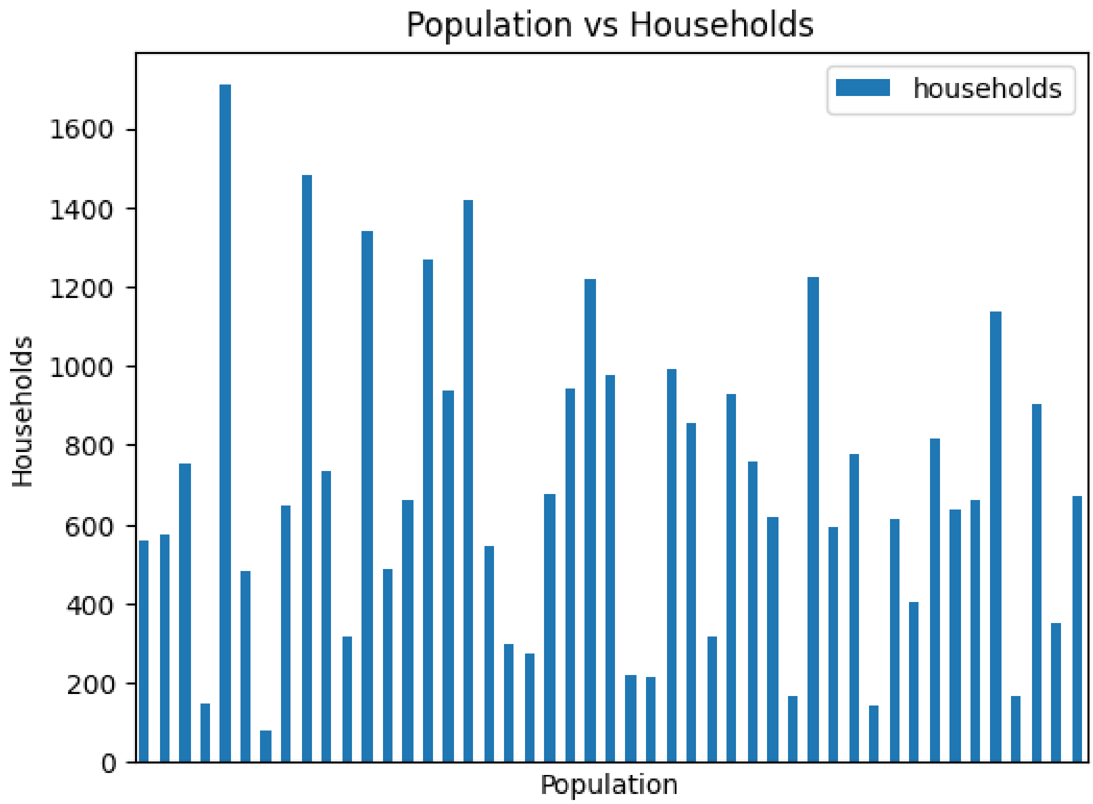


[31]:
```
pop_households = n.groupby('population')['households'].mean().reset_index()
pop_households

pop_households = pop_households.sample(n=50)
pop_households.plot(x='population', y='households', kind='bar')
plt.xticks([])
plt.show()

# remove outliers

q1 = pop_households['households'].quantile(0.25)
q3 = pop_households['households'].quantile(0.75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
pop_households = pop_households[(pop_households['households'] >= lower_bound) &
 ↪(pop_households['households'] <= upper_bound)]
pop_households

outliers = pop_households[pop_households['households'] > lower_bound ]
outliers.plot(x='population', y='households', kind='bar')
plt.xlabel('Population')
```

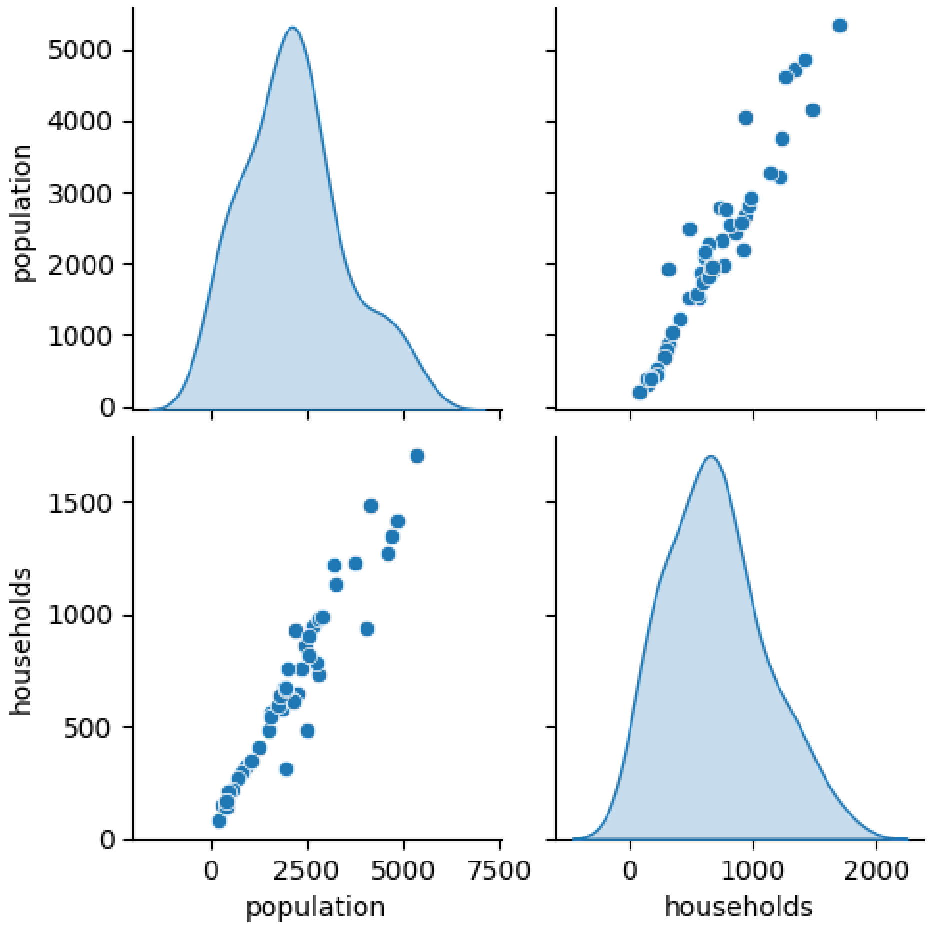
```
plt.ylabel('Households')
plt.title('Outliers Population vs Households')
plt.xticks([])
plt.show()


pop_households.plot(x='population', y='households', kind='bar')
plt.xlabel('Population')
plt.ylabel('Households')
plt.title('Population vs Households')
plt.xticks([])
plt.show()
```




Outliers Population vs Households
households
Households
Population
0
200
400
600
800
1000
1200
1400
1600



[32]:
```
sns.pairplot(pop_households, diag_kind = 'kde', palette = 'viridis')
plt.show()
```

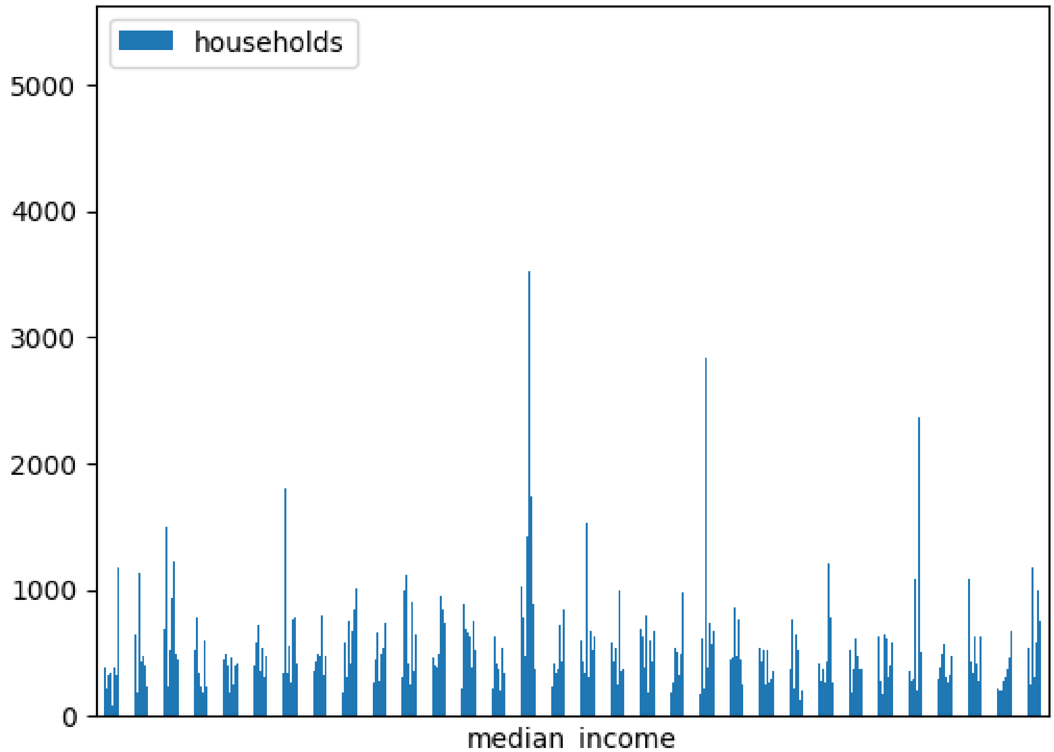


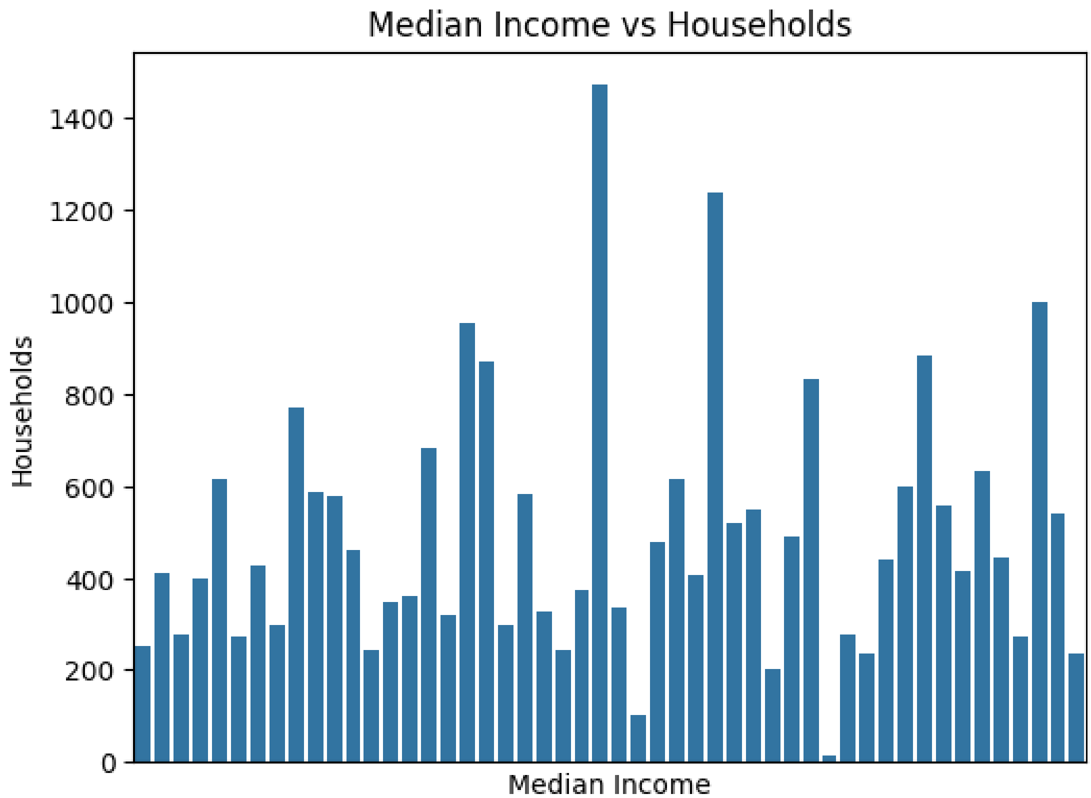
[33]:
```
n.columns
```

[33]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'median_house_value'],
      dtype='object')
```

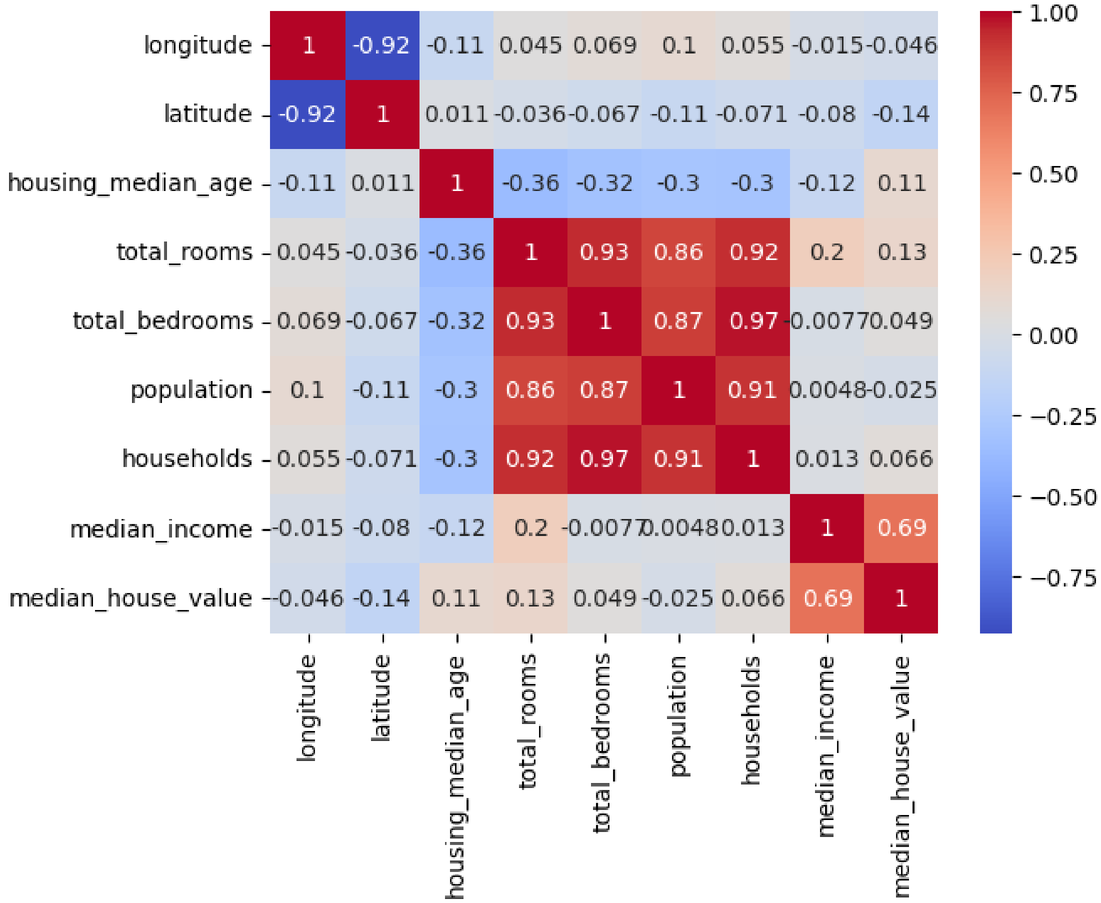
[34]:
```
income_households = n.groupby('median_income')['households'].mean().
 ↪reset_index()
income_households.plot(x='median_income', y='households', kind='bar')
plt.xticks([])
plt.show()

income_households = income_households.sample(n=50)
sns.barplot(x='median_income', y='households', data=income_households)
plt.xlabel('Median Income')
plt.ylabel('Households')
```

```
plt.title('Median Income vs Households')
plt.xticks([])
plt.show()
```





[35]:
```
corr_matrix = n.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```



[36]:
```
corr_matrix = corr_matrix.style.background_gradient(cmap='coolwarm')
corr_matrix
```

[36]: <pandas.io.formats.style.Styler at 0x7fe9dc6cf7d0>

[37]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import *
from sklearn.metrics import *
```

[38]:
```
from sklearn import *
```

[39]:
```
df = df.rename(columns={'median_house_value': 'value'})
df
```

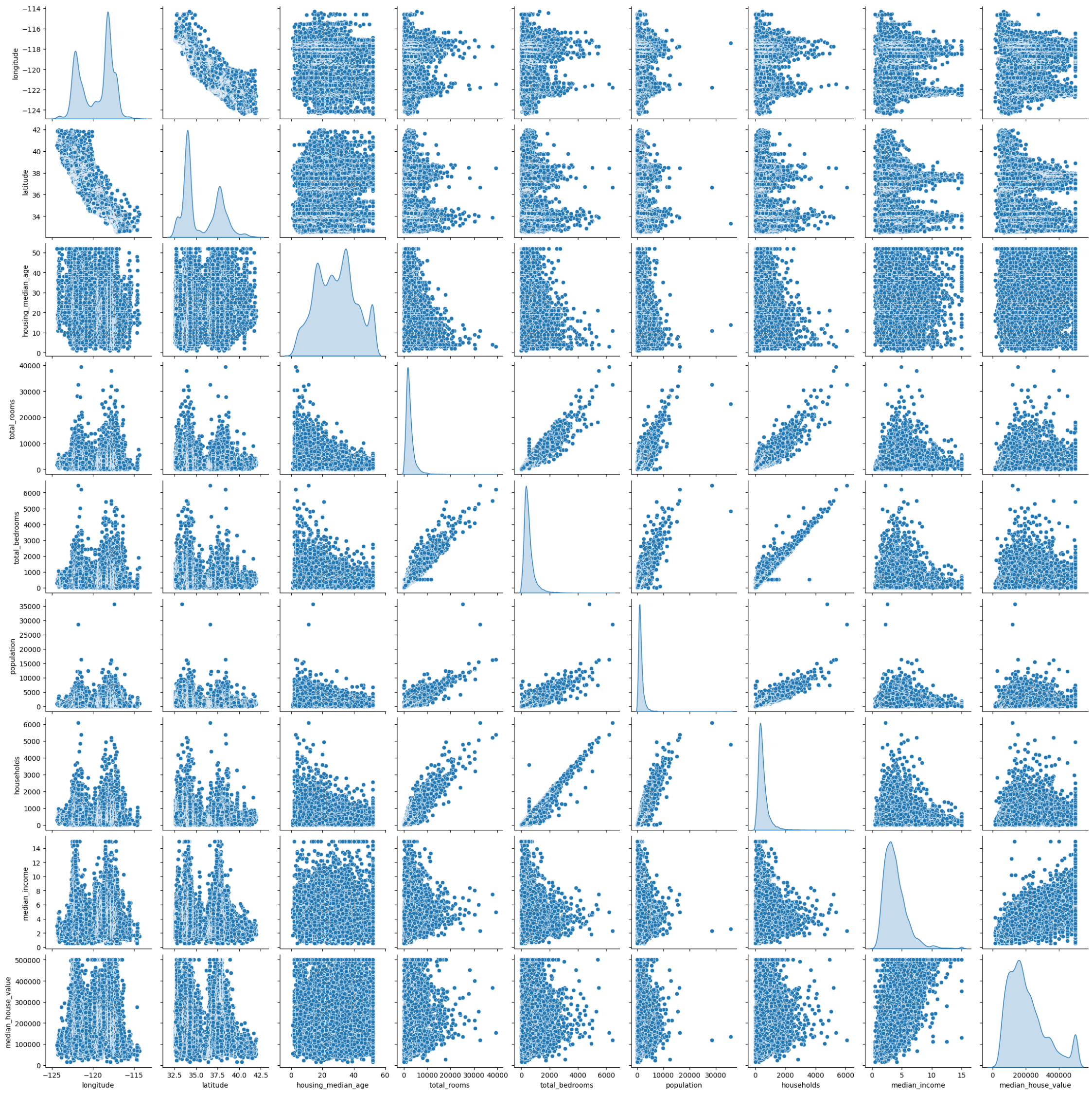
[39]:
```
       longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0        -122.23     37.88                41.0        880.0           129.0
1        -122.22     37.86                21.0       7099.0          1106.0
2        -122.24     37.85                52.0       1467.0           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     value  ocean_proximity
0           322.0       126.0         8.3252  452600.0                3
1          2401.0      1138.0         8.3014  358500.0                3
2           496.0       177.0         7.2574  352100.0                3
3           558.0       219.0         5.6431  341300.0                3
4           565.0       259.0         3.8462  342200.0                3
…             …           …              …         …              …
20635       845.0       330.0         1.5603   78100.0                1
20636       356.0       114.0         2.5568   77100.0                1
20637      1007.0       433.0         1.7000   92300.0                1
20638       741.0       349.0         1.8672   84700.0                1
20639      1387.0       530.0         2.3886   89400.0                1

[20640 rows x 10 columns]
```

[40]:
```
label_encoder = LabelEncoder()
df['ocean_proximity'] = label_encoder.fit_transform(df['ocean_proximity'])
op = df['ocean_proximity']
op
```

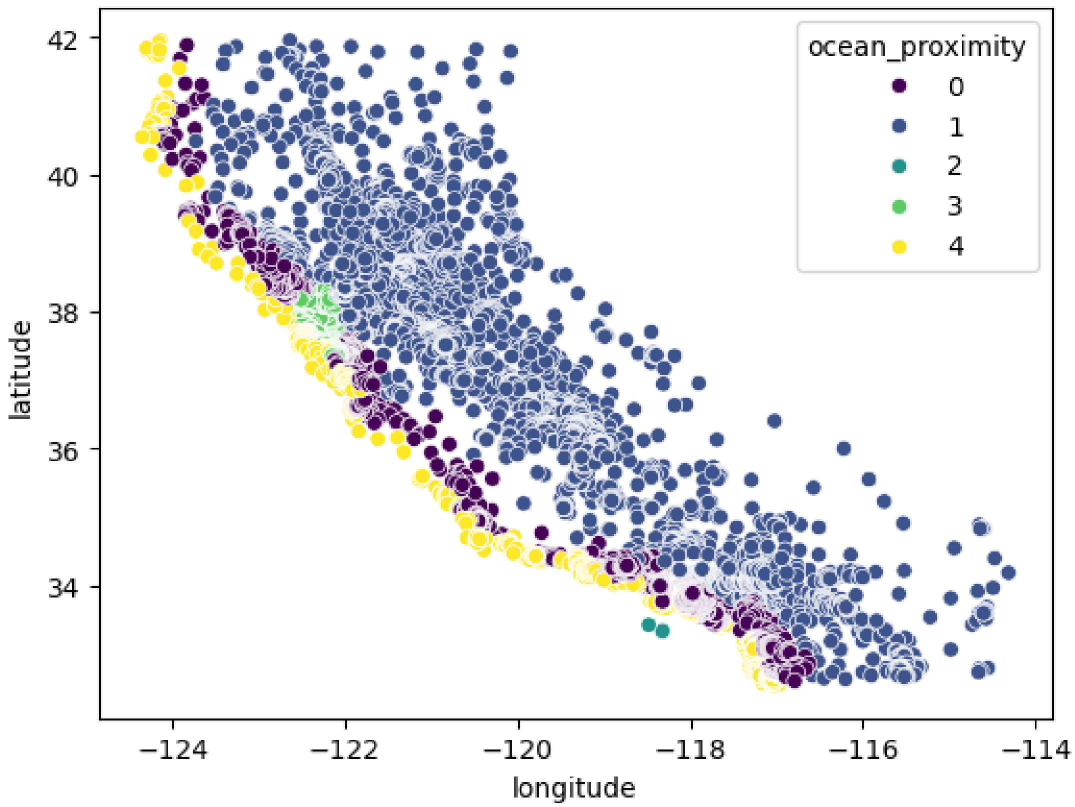
[40]:
```
0        3
1        3
2        3
3        3
4        3
        ..
20635    1
20636    1
20637    1
20638    1
20639    1
```

```
Name: ocean_proximity, Length: 20640, dtype: int64
```

[41]:
```
sns.pairplot(n, diag_kind='kde')
plt.show()
```



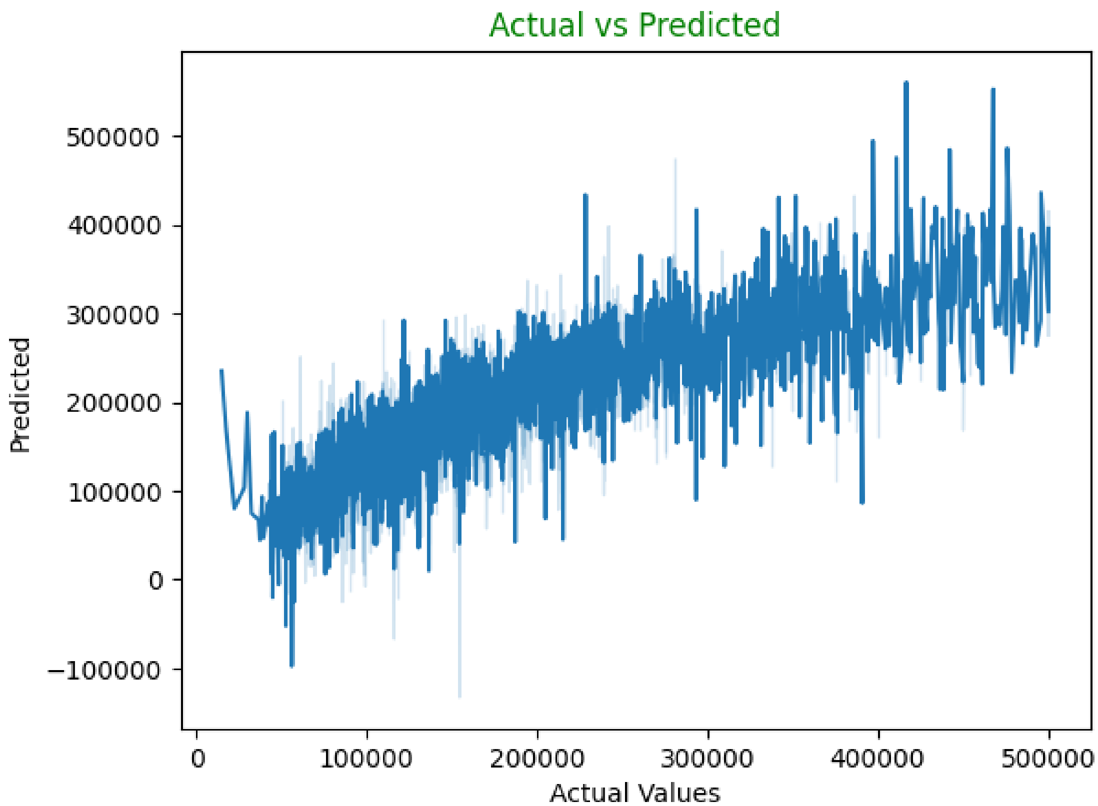
[42]:
```
sns.scatterplot(x=df['longitude'], y= df['latitude'], hue=op, palette =␣
↪'viridis')
plt.show()
```



[43]:
```
x = df.drop('value', axis=1)
y = df['value']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
↪random_state=42)

model = LinearRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 5055025116.165619
R-squared: 0.6142406531011781
```

[44]:
```
sns.lineplot(x=y_test, y=y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted', color = 'green')
plt.show()
```



[45]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score

model = DecisionTreeRegressor()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

```
Mean Squared Error: 4634878555.809109
```

R-squared: 0.6463028998755034

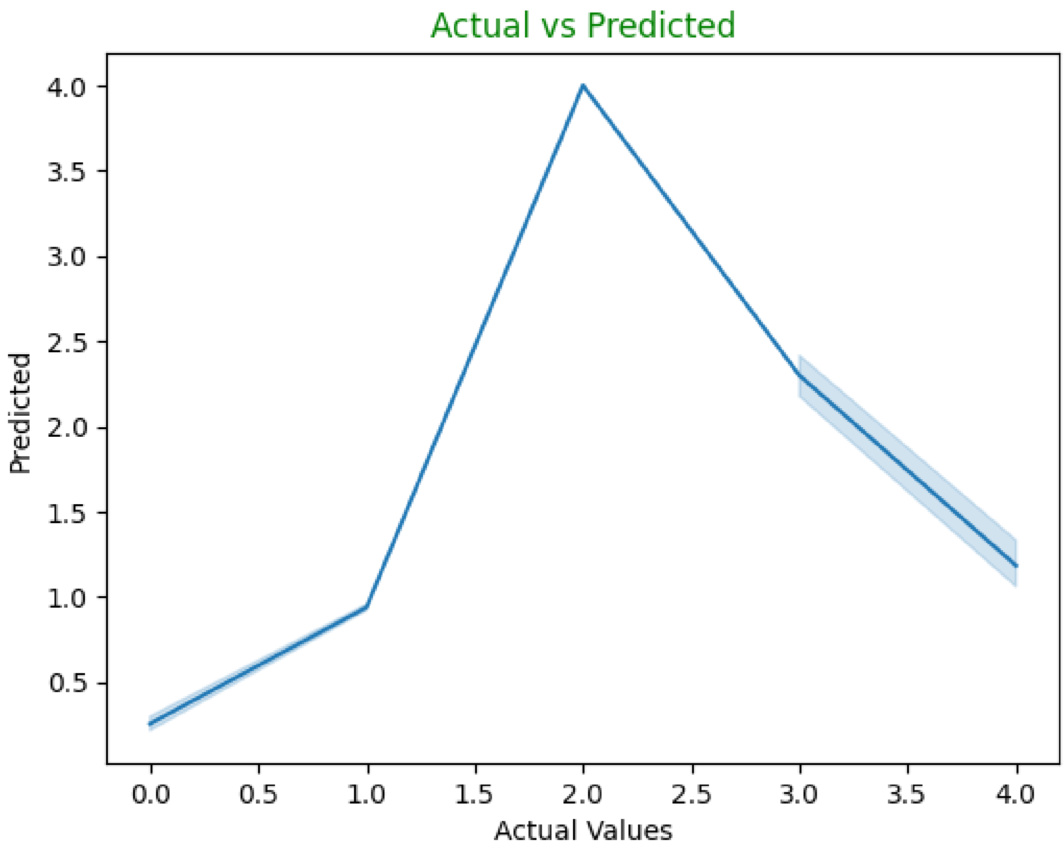
[46]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
```

[47]:
```
x = df.drop('ocean_proximity', axis=1)
y = df['ocean_proximity']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
↪random_state=42)

scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

model = LogisticRegression()
model.fit(x_train_scaled, y_train)

y_pred = model.predict(x_test_scaled)

accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Classification Report:\n", classification_rep)
```

```
Accuracy: 0.7950581395348837
Classification Report:
               precision    recall  f1-score   support

           0       0.74      0.92      0.82      1795
           1       0.97      0.91      0.94      1324
           2       0.00      0.00      0.00         1
           3       0.65      0.76      0.70       436
           4       0.63      0.17      0.26       572

    accuracy                           0.80      4128
   macro avg       0.60      0.55      0.55      4128
weighted avg       0.79      0.80      0.77      4128
```

[48]:
```
sns.lineplot(x=y_test, y=y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted')
```

```
plt.title('Actual vs Predicted', color = 'green')
plt.show()
```



[49]:

```
x = df.drop('total_rooms', axis=1)
y = df['total_rooms']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
↪random_state=42)

scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

model = LinearRegression()
model.fit(x_train_scaled, y_train)

y_pred = model.predict(x_test_scaled)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```python
print("Mean Squared Error:", mse)
print("R-squared:", r2)

sns.lineplot(x=y_test, y=y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted')
plt.title('Actual vs Predicted', color = 'green')
plt.show()
```

```
Mean Squared Error: 494760.2558706711
R-squared: 0.8986164708974816
```