```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [2]: data = pd.read_csv("C:/Users/Sumant kumar/Documents/housing.csv")
```

```
In [3]: data
```

Out[3]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY
...	...	...	...	...	...	...	...	...	...	...
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	330.0	1.5603	78100.0	INLAND
20636	-121.21	39.49	18.0	697.0	150.0	356.0	114.0	2.5568	77100.0	INLAND
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	433.0	1.7000	92300.0	INLAND
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	349.0	1.8672	84700.0	INLAND
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	530.0	2.3886	89400.0	INLAND

20640 rows × 10 columns

```
In [4]: data.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
```

```
In [5]: data.dropna(inplace = True)
```

```
In [6]: data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 20433 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20433 non-null  float64
 1   latitude            20433 non-null  float64
 2   housing_median_age  20433 non-null  float64
 3   total_rooms         20433 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20433 non-null  float64
 6   households          20433 non-null  float64
 7   median_income       20433 non-null  float64
 8   median_house_value  20433 non-null  float64
 9   ocean_proximity     20433 non-null  object 
dtypes: float64(9), object(1)
memory usage: 1.7+ MB
```

```
In [7]: from sklearn.model_selection import train_test_split
        x = data.drop(['median_house_value'],axis=1)
        y =  data['median_house_value']
```

```
In [8]: x_train, x_test ,y_train ,y_test = train_test_split(x ,y ,test_size = 0.2)
```

```
In [9]: train_data = x_train.join(y_train)
```

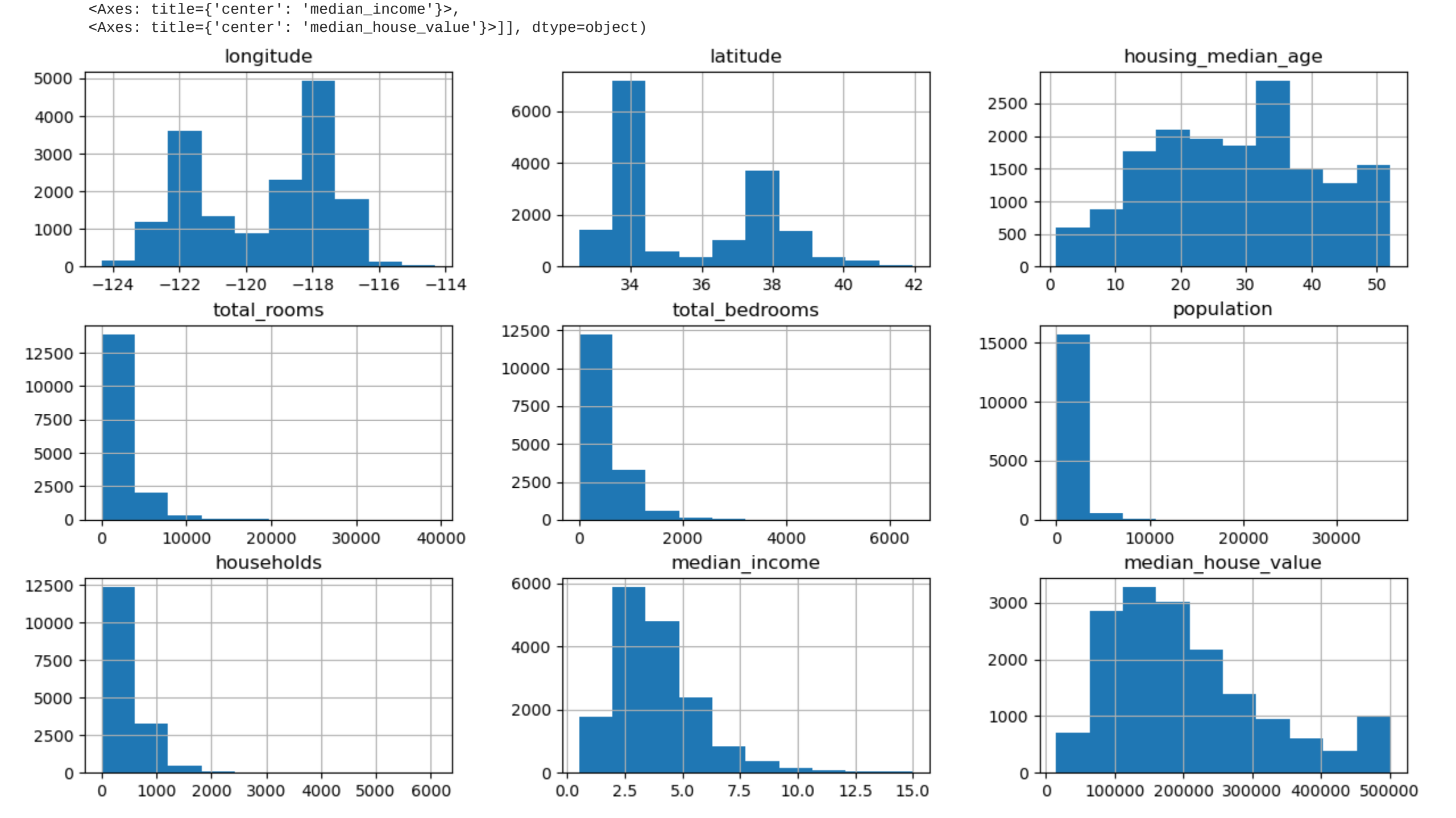
```
In [10]: train_data
```

Out[10]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity	median_house_value
10577	-117.69	33.63	23.0	1444.0	260.0	792.0	253.0	4.9079	<1H OCEAN	273800.0
15025	-117.01	32.77	24.0	2311.0	536.0	1005.0	525.0	2.9000	<1H OCEAN	185200.0
14551	-117.13	32.98	5.0	2276.0	311.0	1158.0	317.0	6.4321	<1H OCEAN	271900.0
12694	-121.40	38.56	22.0	2623.0	357.0	838.0	368.0	7.1430	INLAND	327800.0
8280	-118.15	33.78	12.0	4436.0	1133.0	2176.0	1002.0	3.5812	NEAR OCEAN	198600.0
...	...	...	...	...	...	...	...	...	...	...
1516	-122.07	37.93	45.0	1544.0	244.0	614.0	238.0	5.0255	NEAR BAY	226000.0
11410	-117.93	33.71	10.0	2775.0	717.0	1581.0	633.0	4.1366	<1H OCEAN	158800.0
4739	-118.38	34.05	40.0	2352.0	598.0	1133.0	563.0	3.2380	<1H OCEAN	287500.0
10207	-117.92	33.88	52.0	1270.0	276.0	609.0	211.0	3.7500	<1H OCEAN	232500.0
8401	-118.36	33.94	39.0	1390.0	410.0	1666.0	371.0	3.3056	<1H OCEAN	156800.0

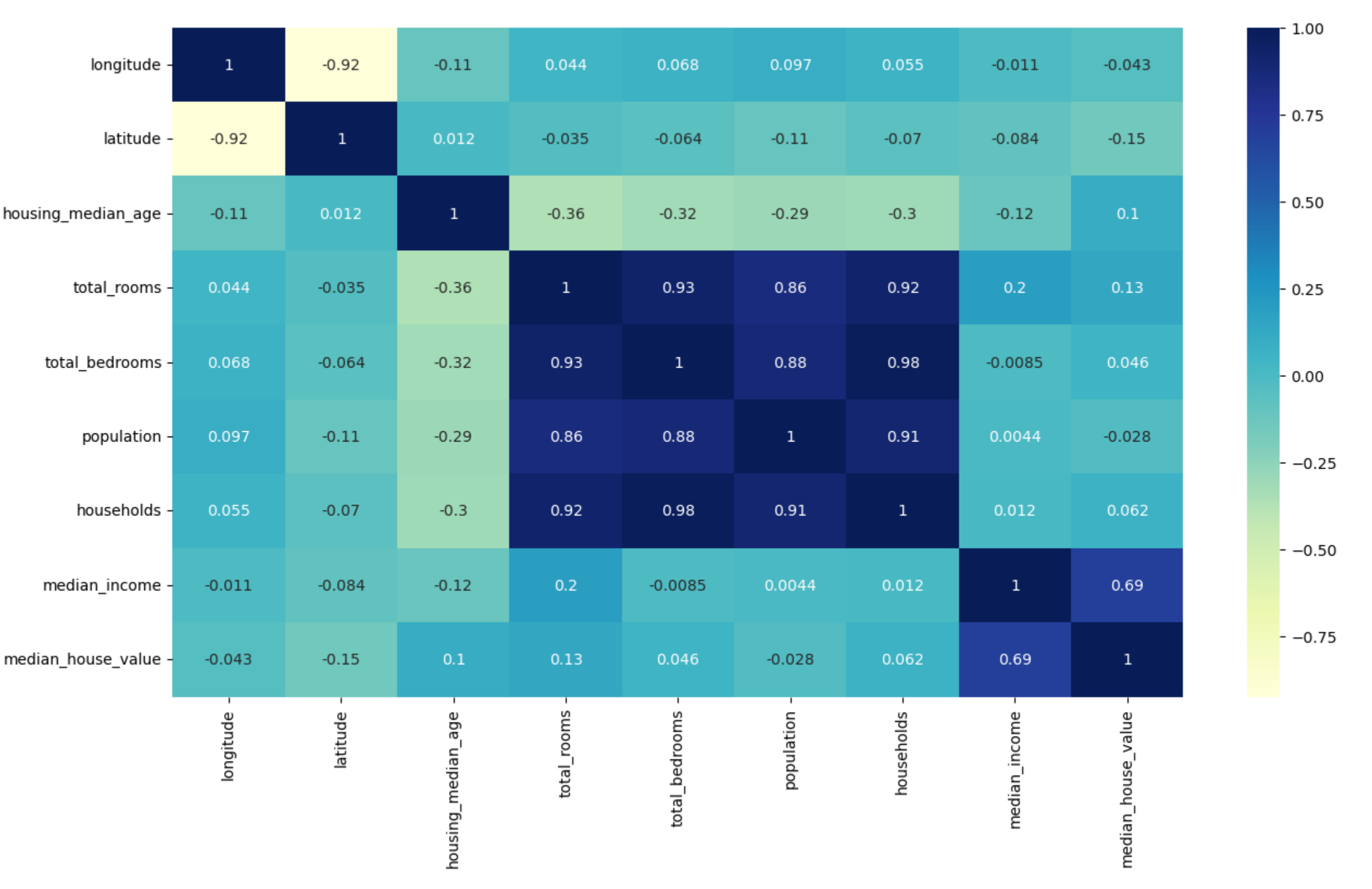
16346 rows × 10 columns

```
In [11]: train_data.hist(figsize = (15,8))

Out[11]: array([[<Axes: title={'center': 'longitude'}>,
        <Axes: title={'center': 'latitude'}>,
        <Axes: title={'center': 'housing_median_age'}>],
       [<Axes: title={'center': 'total_rooms'}>,
        <Axes: title={'center': 'total_bedrooms'}>,
        <Axes: title={'center': 'population'}>],
       [<Axes: title={'center': 'households'}>,
        <Axes: title={'center': 'median_income'}>,
        <Axes: title={'center': 'median_house_value'}>]], dtype=object)
```



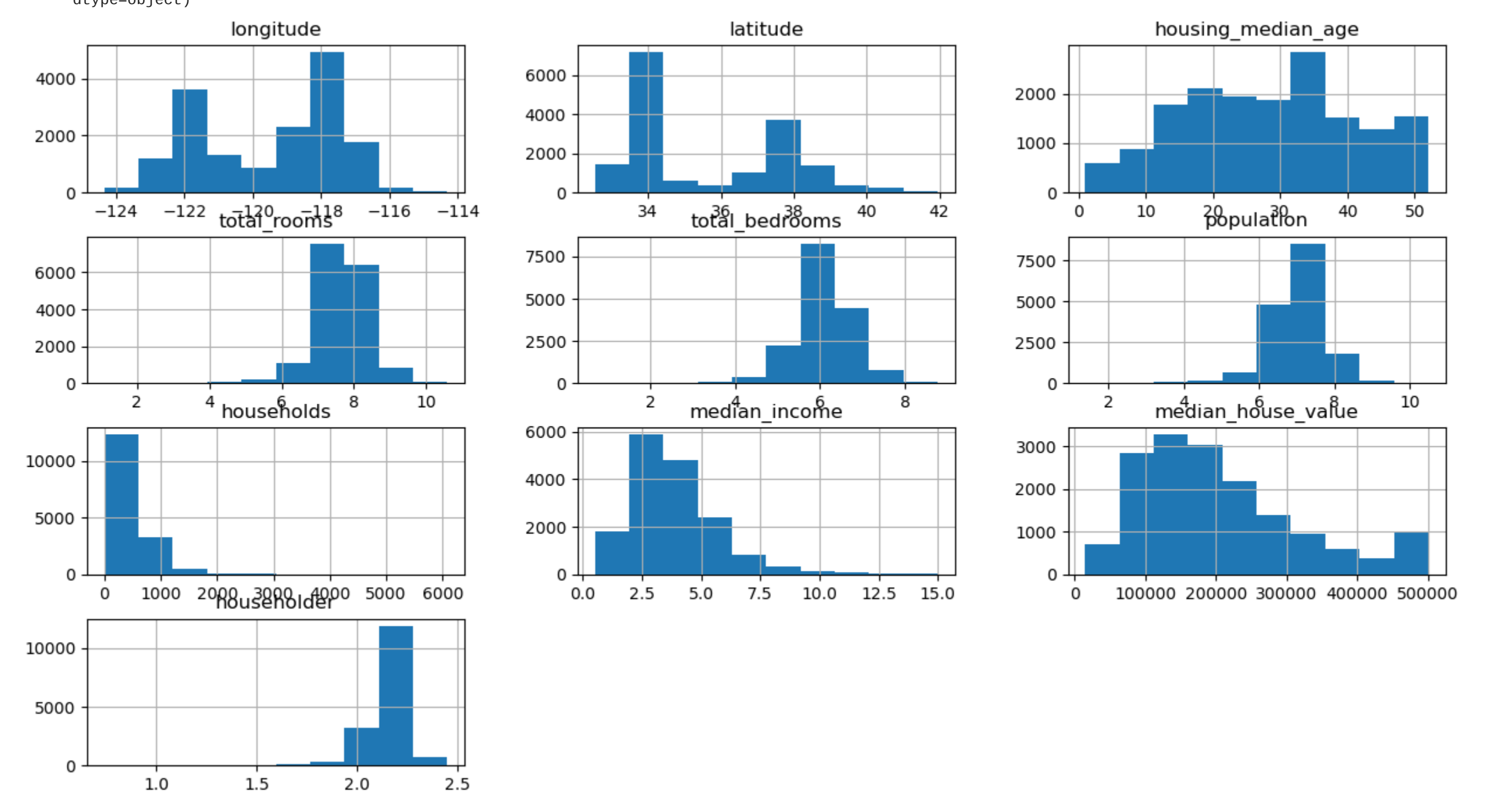

```
In [12]: plt.figure(figsize = (15,8))
         sns.heatmap(train_data.corr(),annot =True,cmap="YlGnBu")
```

```
C:\Users\Sumant kumar\AppData\Local\Temp\ipykernel_10968\3601385669.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(train_data.corr(),annot =True,cmap="YlGnBu")
Out[12]: <Axes: >
```




```
In [13]: train_data['total_rooms'] =np.log(train_data['total_rooms'] +1)
         train_data['total_bedrooms'] =np.log(train_data['total_bedrooms'] +1)
         train_data['population'] =np.log(train_data['population'] +1)
         train_data['householder'] =np.log(train_data['total_rooms'] +1)
```

```
In [14]: train_data.hist(figsize=(15,8))

Out[14]: array([[<Axes: title={'center': 'longitude'}>,
        <Axes: title={'center': 'latitude'}>,
        <Axes: title={'center': 'housing_median_age'}>],
       [<Axes: title={'center': 'total_rooms'}>,
        <Axes: title={'center': 'total_bedrooms'}>,
        <Axes: title={'center': 'population'}>],
       [<Axes: title={'center': 'households'}>,
        <Axes: title={'center': 'median_income'}>,
        <Axes: title={'center': 'median_house_value'}>],
       [<Axes: title={'center': 'householder'}>, <Axes: >, <Axes: >]],
      dtype=object)
```




```
In [15]: train_data=train_data.join(pd.get_dummies(train_data.ocean_proximity)).drop(['ocean_proximity'],axis=1)
```

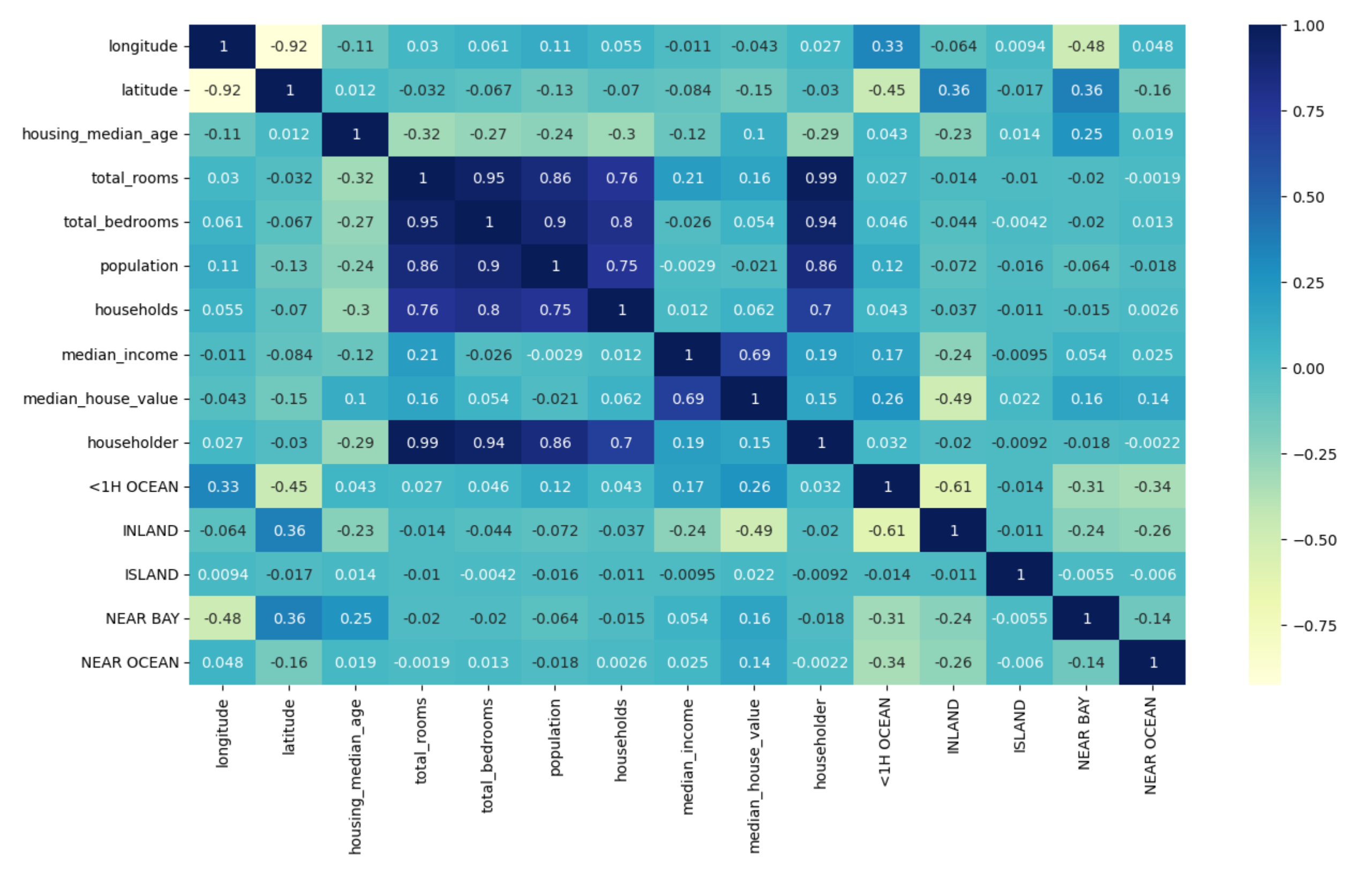
```
In [16]: train_data
```

Out[16]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	householder	<1H OCEAN	INLAND	ISLAND	NEAR BAY	NEAR OCEAN
10577	-117.69	33.63	23.0	7.275865	5.564520	6.675823	253.0	4.9079	273800.0	2.113343	1	0	0	0	0
15025	-117.01	32.77	24.0	7.745868	6.285998	6.913737	525.0	2.9000	185200.0	2.168581	1	0	0	0	0
14551	-117.13	32.98	5.0	7.730614	5.743003	7.055313	317.0	6.4321	271900.0	2.166836	1	0	0	0	0
12694	-121.40	38.56	22.0	7.872455	5.880533	6.732211	368.0	7.1430	327800.0	2.182952	0	1	0	0	0
8280	-118.15	33.78	12.0	8.397734	7.033506	7.685703	1002.0	3.5812	198600.0	2.240469	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1516	-122.07	37.93	45.0	7.342779	5.501258	6.421622	238.0	5.0255	226000.0	2.121396	0	0	0	1	0
11410	-117.93	33.71	10.0	7.928766	6.576470	7.366445	633.0	4.1366	158800.0	2.189278	1	0	0	0	0
4739	-118.38	34.05	40.0	7.763446	6.395262	7.033506	563.0	3.2380	287500.0	2.170589	1	0	0	0	0
10207	-117.92	33.88	52.0	7.147559	5.624018	6.413459	211.0	3.7500	232500.0	2.097718	1	0	0	0	0
8401	-118.36	33.94	39.0	7.237778	6.018593	7.418781	371.0	3.3056	156800.0	2.108731	1	0	0	0	0

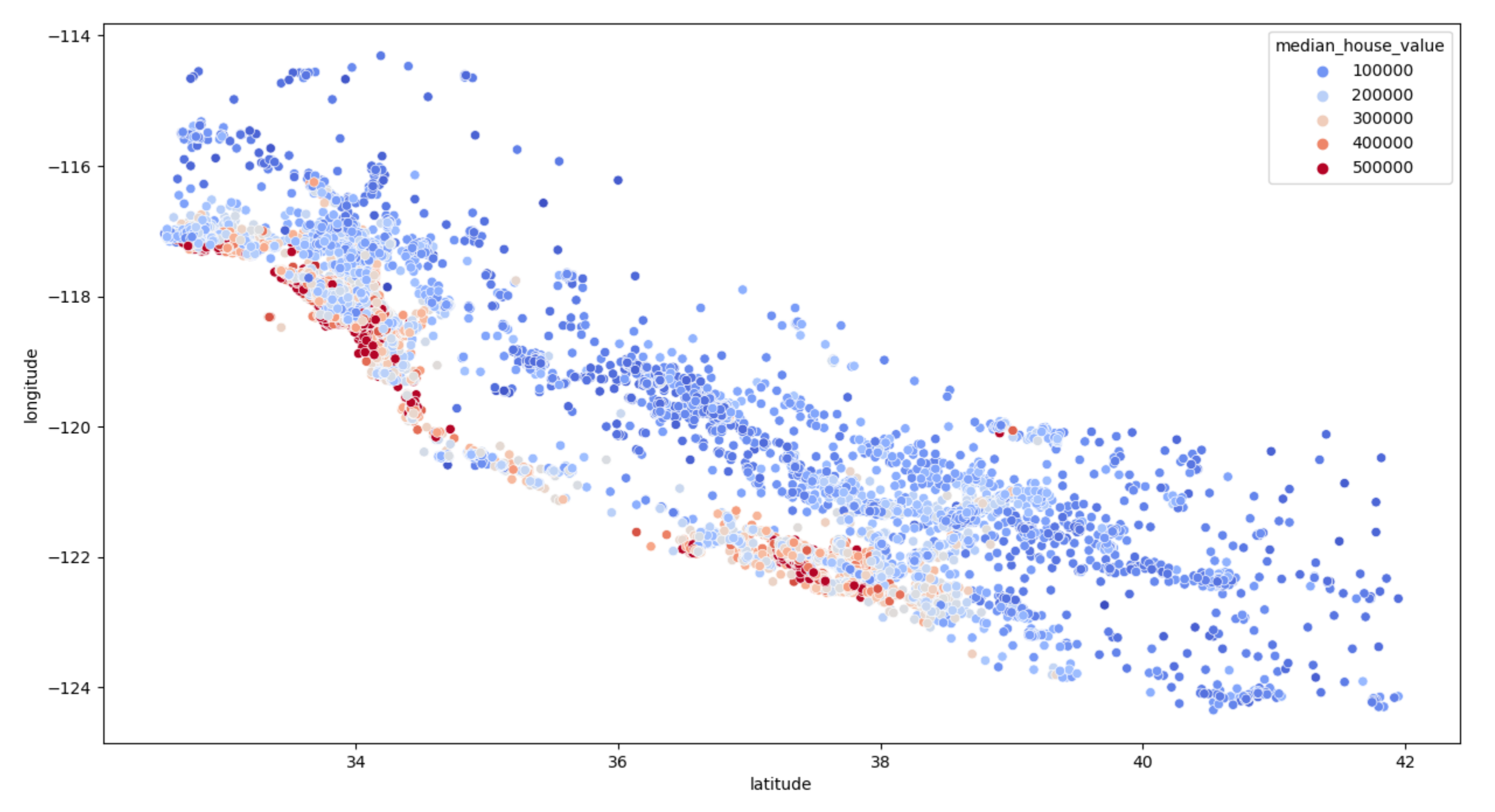
16346 rows × 15 columns

```
In [17]: plt.figure(figsize = (15,8))
         sns.heatmap(train_data.corr(),annot =True,cmap="YlGnBu")

Out[17]: <Axes: >
```



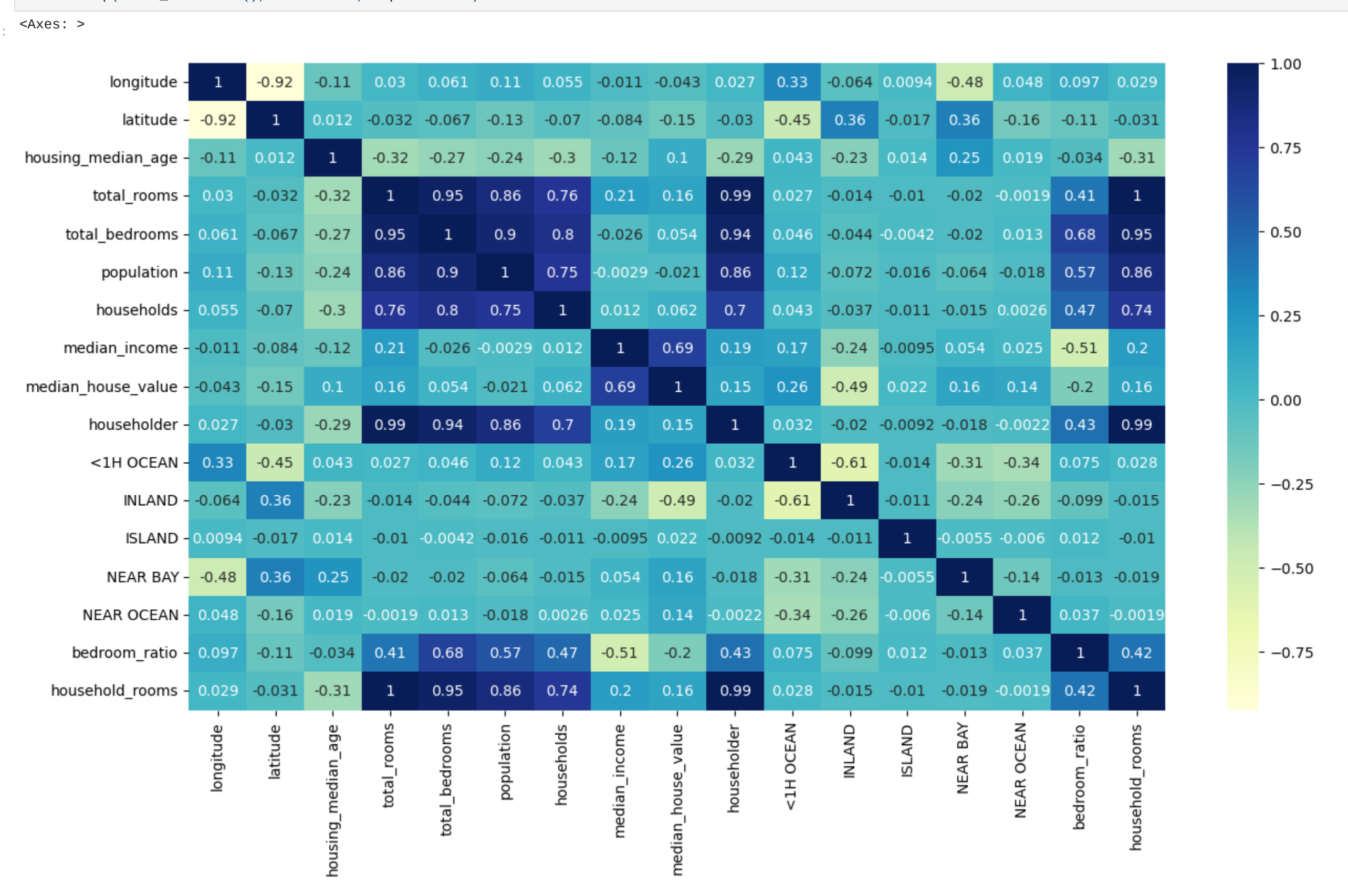

```
In [ ]: 
```

```
In [18]: plt.figure(figsize = (15,8))
         sns.scatterplot(x='latitude',y='longitude',data=train_data,hue ="median_house_value",palette='coolwarm')

Out[18]: <Axes: xlabel='latitude', ylabel='longitude'>
```




```
In [19]: train_data['bedroom_ratio'] =train_data['total_bedrooms']/train_data['total_rooms']
         train_data['household_rooms'] =train_data['total_rooms']/train_data['householder']
```

```
In [20]: plt.figure(figsize = (15,8))
         sns.heatmap(train_data.corr(),annot =True,cmap="YlGnBu")

Out[20]: <Axes: >
```




```
In [21]: from sklearn.linear_model import LinearRegression

         from sklearn.preprocessing import StandardScaler
         scaler =StandardScaler()

         x_train,y_train =train_data.drop(['median_house_value'],axis = 1),train_data['median_house_value']
         x_train_s =scaler.fit_transform(x_train)

         reg = LinearRegression()

         reg.fit(x_train,y_train)

Out[21]: ▾ LinearRegression
         LinearRegression()
```

```
In [22]: test_data = x_test.join(y_test)
         test_data['total_rooms'] =np.log(test_data['total_rooms'] +1)
         test_data['total_bedrooms'] =np.log(test_data['total_bedrooms'] +1)
         test_data['population'] =np.log(test_data['population'] +1)
         test_data['householder'] =np.log(test_data['total_rooms'] +1)

         test_data=test_data.join(pd.get_dummies(test_data.ocean_proximity)).drop(['ocean_proximity'],axis=1)

         test_data['bedroom_ratio'] =test_data['total_bedrooms']/test_data['total_rooms']
         test_data['household_rooms'] =test_data['total_rooms']/test_data['householder']

         x_test,y_test =test_data.drop(['median_house_value'],axis = 1),test_data['median_house_value']
```

```
In [23]: # x_test_s = scaler.transform(x_test)
```

```
In [24]: reg.score(x_test,y_test)

Out[24]: 0.6456546365029245
```

```
In [25]: from sklearn.ensemble import RandomForestRegressor

         forest = RandomForestRegressor()

         forest.fit(x_train, y_train)

Out[25]: ▾ RandomForestRegressor
         RandomForestRegressor()
```

```
In [26]: forest.score(x_test,y_test)

Out[26]: 0.8009850525743809
```

```
In [27]: from sklearn.model_selection import GridSearchCV
         forest = RandomForestRegressor()

         param_grid= {
             "n_estimators":[3,10,30],
             "max_features":[2,4,6,8]
         }
         grid_search =GridSearchCV(forest, param_grid ,cv =5,
                                   scoring ="neg_mean_squared_error",
                                   return_train_score=True)

         grid_search.fit(x_train,y_train)

Out[27]: ▸ GridSearchCV
         ▸ estimator: RandomForestRegressor
           ▸ RandomForestRegressor
```

```
In [28]: grid_search.best_estimator_

Out[28]: ▾ RandomForestRegressor
         RandomForestRegressor(max_features=8, n_estimators=30)
```

```
In [29]: grid_search.best_estimator_.score(x_test,y_test)

Out[29]: 0.8027703607679195
```

```
In [ ]: 
```