In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
data = pd.read_csv("housing.csv")
```

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

Out[4]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	housel
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20640 rows × 10 columns

In [5]:
```
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 [6]:
```
data.dropna(inplace = True)
```

In [7]:
```
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 [12]:
```
from sklearn.model_selection import train_test_split

X = data.drop(['median_house_value'], axis = 1)
y = data['median_house_value']
```

In [9]:
```
X
```

Out[9]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
0	-122.23	37.88	41.0	880.0	129.0	322.0	
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1
2	-122.24	37.85	52.0	1467.0	190.0	496.0	
3	-122.25	37.85	52.0	1274.0	235.0	558.0	
4	-122.25	37.85	52.0	1627.0	280.0	565.0	
...	...	...	...	...	...	...	
20635	-121.09	39.48	25.0	1665.0	374.0	845.0	
20636	-121.21	39.49	18.0	697.0	150.0	356.0	
20637	-121.22	39.43	17.0	2254.0	485.0	1007.0	
20638	-121.32	39.43	18.0	1860.0	409.0	741.0	
20639	-121.24	39.37	16.0	2785.0	616.0	1387.0	

20433 rows × 9 columns

In [13]:
```
y
```

Out[13]:
```
0        452600.0
1        358500.0
2        352100.0
3        341300.0
4        342200.0
           ...
20635     78100.0
20636     77100.0
20637     92300.0
20638     84700.0
20639     89400.0
Name: median_house_value, Length: 20433, dtype: float64
```

In [14]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2)
```

In [15]:
```
train_data = X_train.join(y_train)
```

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

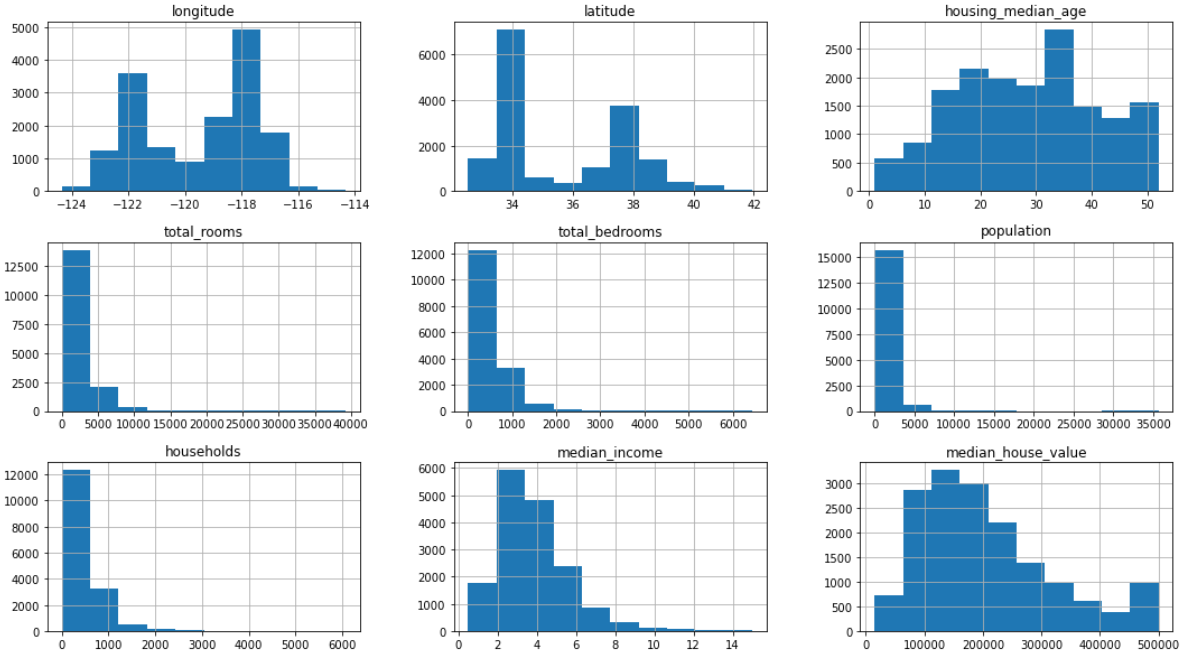
Out[16]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
5753	-118.27	34.18	52.0	3034.0	406.0	1158.0	
19035	-121.98	38.36	24.0	2434.0	630.0	1538.0	
4812	-118.37	34.02	44.0	1944.0	458.0	981.0	
4517	-118.20	34.04	44.0	1399.0	386.0	1419.0	
17676	-121.84	37.32	14.0	5762.0	1538.0	3979.0	1
...	...	...	...	...	...	...	
15225	-117.26	33.06	11.0	2660.0	352.0	1226.0	
2988	-119.02	35.33	35.0	2053.0	412.0	1193.0	
16652	-120.66	35.28	31.0	2773.0	844.0	1358.0	
11300	-117.92	33.77	28.0	3614.0	960.0	3282.0	
20071	-120.38	37.99	36.0	2864.0	603.0	1155.0	

16346 rows × 10 columns

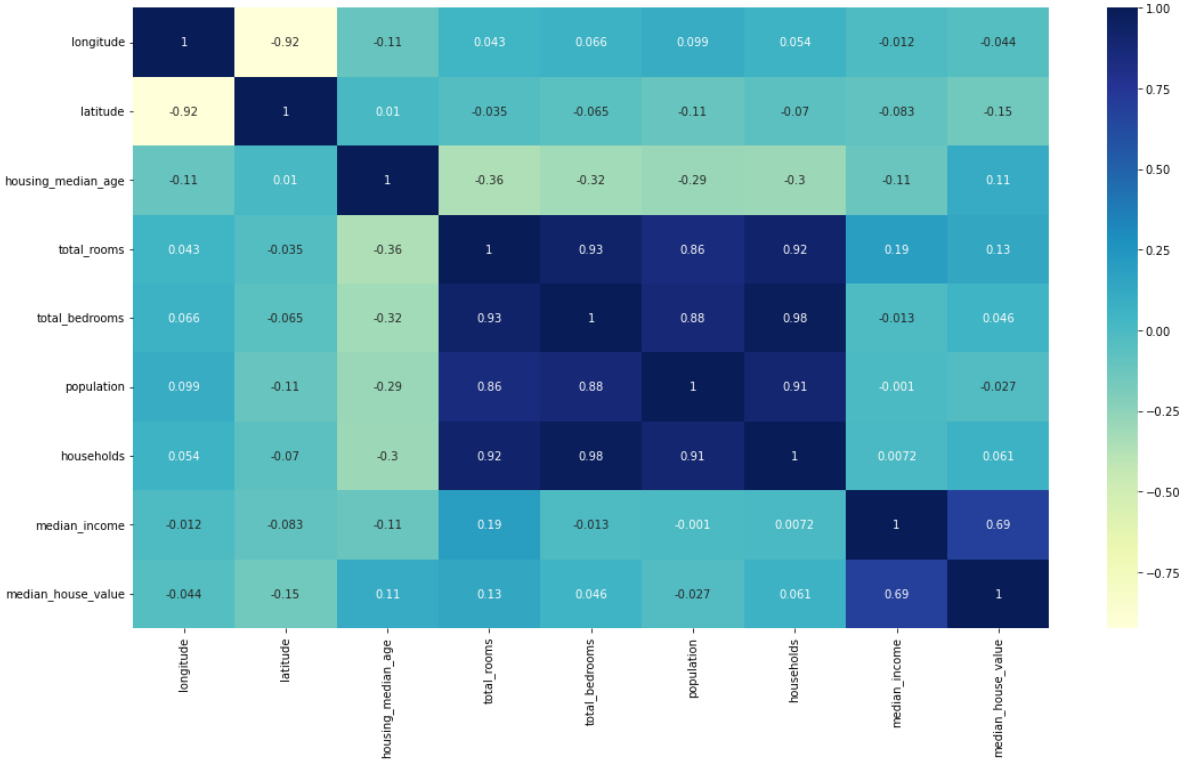
In [17]:
```
train_data.hist(figsize = (18,10))
```

Out[17]:
```
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)
```



In [18]:
```
plt.figure(figsize = (18,10))
sns.heatmap(train_data.corr(), annot = True, cmap = "YlGnBu")
```

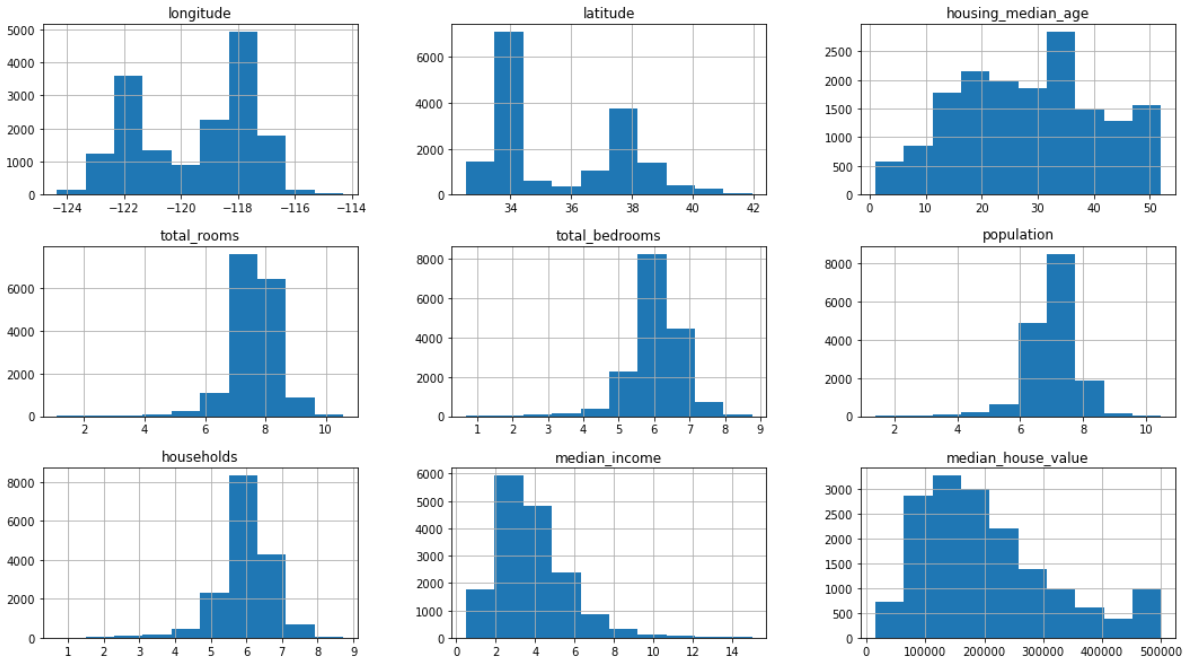
Out[18]: <AxesSubplot:>



In [19]:
```
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['households'] = np.log(train_data['households'] + 1)
```

In [20]:
```
train_data.hist(figsize=(18,10))
```

Out[20]:
```
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)
```



In [21]:
```
train_data.join(pd.get_dummies(train_data.ocean_proximity))
```

Out[21]:

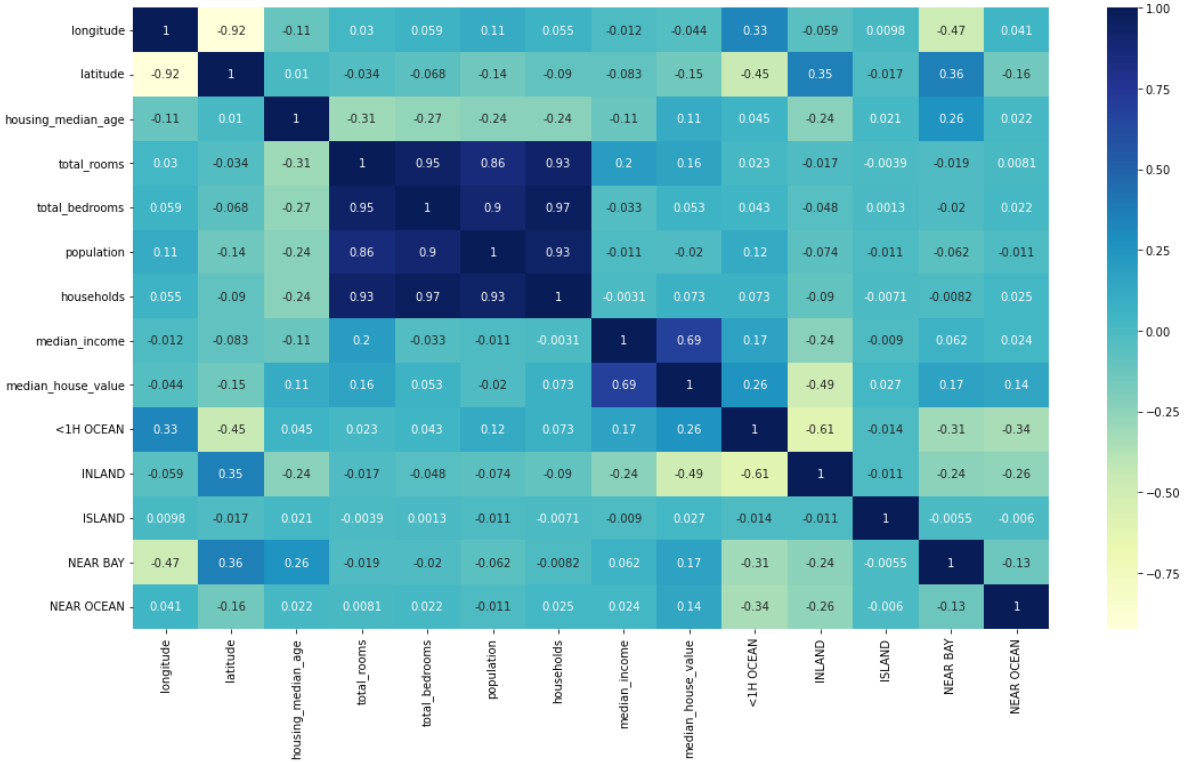
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	househ
5753	-118.27	34.18	52.0	8.017967	6.008813	7.055313	5.99
19035	-121.98	38.36	24.0	7.797702	6.447306	7.338888	6.35
4812	-118.37	34.02	44.0	7.573017	6.129050	6.889591	5.93
4517	-118.20	34.04	44.0	7.244228	5.958425	7.258412	5.92
17676	-121.84	37.32	14.0	8.659213	7.338888	8.289037	7.23
...	...	...	...	...	...	...	
15225	-117.26	33.06	11.0	7.886457	5.866468	7.112327	5.90
2988	-119.02	35.33	35.0	7.627544	6.023448	7.085064	5.96
16652	-120.66	35.28	31.0	7.928046	6.739337	7.214504	6.67
11300	-117.92	33.77	28.0	8.192847	6.867974	8.096513	6.79
20071	-120.38	37.99	36.0	7.960324	6.403574	7.052721	6.33

16346 rows × 15 columns

In [22]:
```
train_data = train_data.join(pd.get_dummies(train_data.ocean_proximity)).drop(['oc
```

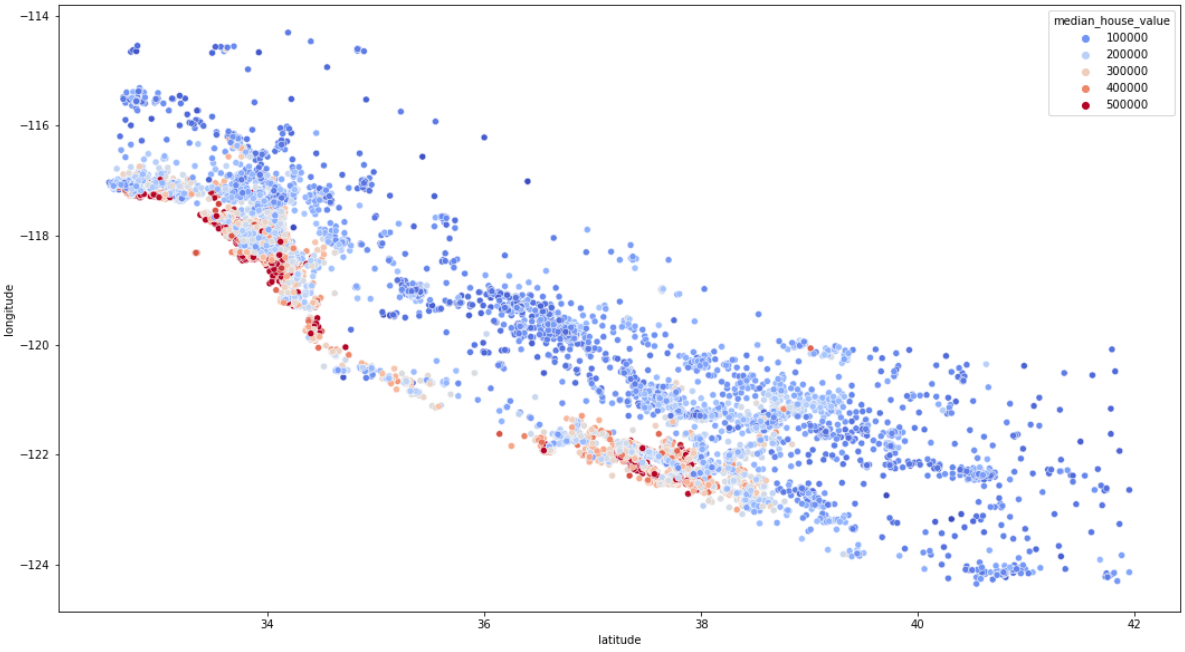
In [23]:
```
plt.figure(figsize = (18,10))
sns.heatmap(train_data.corr(), annot = True, cmap = "YlGnBu")
```

Out[23]:
```
<AxesSubplot:>
```



In [24]:
```
plt.figure(figsize=(18,10))
sns.scatterplot(x = "latitude", y = "longitude", data = train_data, hue = "median_
```

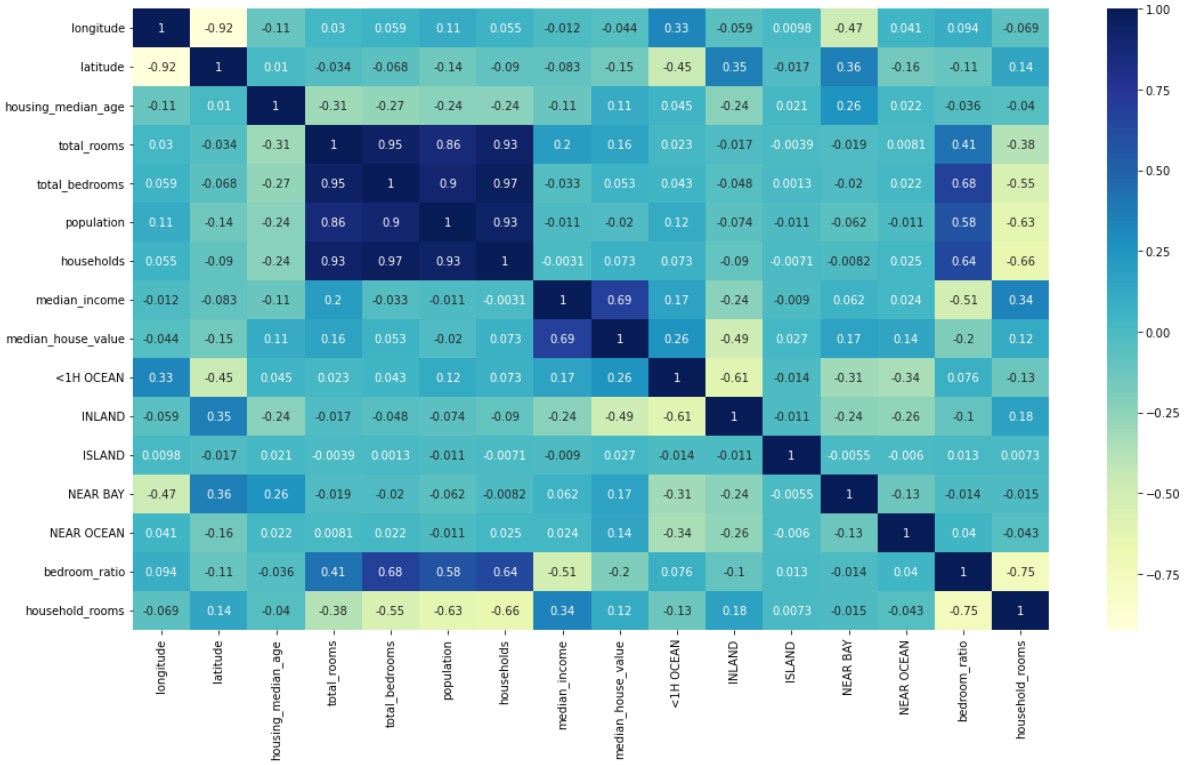
Out[24]:
```
<AxesSubplot:xlabel='latitude', ylabel='longitude'>
```



In [25]:
```
train_data['bedroom_ratio'] = train_data['total_bedrooms'] / train_data['total_roo
train_data['household_rooms'] = train_data['total_rooms'] / train_data['households
```

In [26]:
```
plt.figure(figsize = (18,10))
sns.heatmap(train_data.corr(), annot = True, cmap = "YlGnBu")
```

Out[26]:
```
<AxesSubplot:>
```



In [27]:
```
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['
X_train_s = scaler.fit_transform(X_train)

reg = LinearRegression()

reg.fit(X_train_s, y_train)
```

Out[27]:
```
LinearRegression()
```

In [28]:
```
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['households'] = np.log(test_data['households'] + 1)

test_data = test_data.join(pd.get_dummies(test_data.ocean_proximity)).drop(['ocean_

test_data['bedroom_ratio'] = test_data['total_bedrooms'] / test_data['total_rooms'
test_data['household_rooms'] = test_data['total_rooms'] / test_data['households']
```

In [29]:
```
X_test, y_test = test_data.drop(['median_house_value'], axis = 1), test_data['medi
```

In [30]:
```
X_test_s = scaler.transform(X_test)
```

In [31]:
```
X_train
```

Out[31]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	housel
5753	-118.27	34.18	52.0	8.017967	6.008813	7.055313	5.99
19035	-121.98	38.36	24.0	7.797702	6.447306	7.338888	6.35
4812	-118.37	34.02	44.0	7.573017	6.129050	6.889591	5.93
4517	-118.20	34.04	44.0	7.244228	5.958425	7.258412	5.92
17676	-121.84	37.32	14.0	8.659213	7.338888	8.289037	7.23
...	...	...	...	...	...	...	
15225	-117.26	33.06	11.0	7.886457	5.866468	7.112327	5.90
2988	-119.02	35.33	35.0	7.627544	6.023448	7.085064	5.96
16652	-120.66	35.28	31.0	7.928046	6.739337	7.214504	6.67
11300	-117.92	33.77	28.0	8.192847	6.867974	8.096513	6.79
20071	-120.38	37.99	36.0	7.960324	6.403574	7.052721	6.33

16346 rows × 15 columns

In [32]:
```
X_test
```

Out[32]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	housel
20013	-119.01	36.02	17.0	8.272826	6.610696	7.478170	6.53
3225	-119.66	36.30	18.0	7.045777	5.313206	6.576470	5.36
8520	-118.33	33.90	21.0	8.795431	7.593374	8.621013	7.46
17338	-120.45	34.87	4.0	7.335634	5.402677	6.302619	5.25
5800	-118.24	34.15	7.0	7.632401	6.508769	7.545918	6.46
...	...	...	...	...	...	...	
19508	-121.05	37.62	37.0	6.950815	5.283204	6.320768	5.28
4697	-118.37	34.07	52.0	6.989335	5.513429	6.150603	5.54
16824	-122.49	37.63	31.0	8.042378	6.432940	7.295056	6.42
8626	-118.39	33.88	34.0	7.587817	5.908083	6.738152	5.84
1963	-120.58	38.77	15.0	7.676010	5.978886	6.754604	5.87

4087 rows × 15 columns

In [34]:
```
reg.score(X_test_s, y_test)
```

Out[34]:
```
0.6620549129177645
```

In [35]:
```
from sklearn.ensemble import RandomForestRegressor

forest = RandomForestRegressor()
```

```
forest.fit(X_train, y_train)
```

Out[35]:
```
RandomForestRegressor()
```

In [36]:
```
forest.score(X_test, y_test)
```

Out[36]:
```
0.8029273594609136
```

In [43]:
```
from sklearn.model_selection import GridSearchCV

param_grid = {
    "n_estimators": [100, 200, 300],
    "min_samples_split": [2, 4],
    "max_depth": [None, 4, 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[43]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'max_depth': [None, 4, 8], 'min_samples_split': [2, 4],
                         'n_estimators': [100, 200, 300]},
             return_train_score=True, scoring='neg_mean_squared_error')
```

In [45]:
```
grid_search.best_estimator_
```

Out[45]:
```
RandomForestRegressor(n_estimators=300)
```

In [46]:
```
grid_search.best_estimator_.score(X_test,y_test)
```

Out[46]:
```
0.805869618797127
```