This Python notebook demonstrates the process of predicting median house price values using the California housing dataset.

It uses Linear Regression, Random Forest to build predictive models. Additionally, it also uses Scaling and Hyperparameter tuning using RandomizedSearchCV to achieve better results.

In [1]:
```
# Importing dependency libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Importing the data from the CSV file into the Pandas DataFrame data

data = pd.read_csv('housing.csv')
```

In [3]:
```
# Initial data exploration, finding null values
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 [4]:
```
# Dropping null value containing rows
data.dropna(inplace=True)
```

In [5]:
```
# Verifying that those 207 rows containing null values in the total_bedrooms were drop
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 [6]:
```
# Importing a method to split the d
from sklearn.model_selection import train_test_split

# Defining a Pandas DataFrame with just independent variables (X) and the other with j
X = data.drop(['median_house_value'], axis=1)
Y = data['median_house_value']
```

In [7]:
```
# Splitting the DataFrames into training and testing datasets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
```

In [8]:
```
# Joining the 'X_train' and 'Y_train' DataFrames, creating a new DataFrame named 'trai
train_data = X_train.join(Y_train)
train_data
```

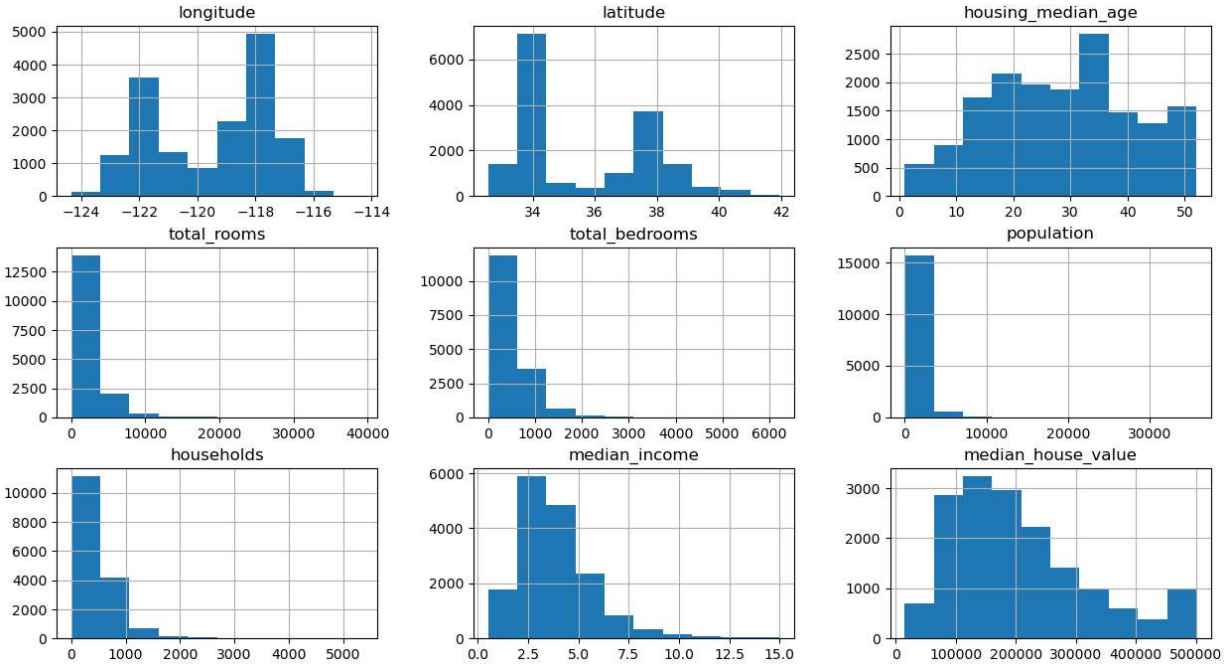
Out[8]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
9717	-121.79	36.85	28.0	1049.0	235.0	705.0	208.(
14544	-117.25	32.96	18.0	4773.0	743.0	1970.0	716.(
139	-122.20	37.82	39.0	3770.0	534.0	1265.0	500.(
17538	-121.89	37.35	43.0	1185.0	296.0	933.0	244.(
2189	-120.08	36.72	22.0	1339.0	251.0	820.0	276.(
...	...	...	...	...	...	...	.
17812	-121.86	37.39	17.0	1777.0	328.0	1235.0	329.(
1925	-120.50	38.87	10.0	81.0	41.0	55.0	16.(
1583	-121.97	37.87	4.0	1029.0	126.0	416.0	122.(
13681	-117.23	34.15	17.0	5036.0	817.0	2084.0	833.(
20013	-119.01	36.02	17.0	3915.0	742.0	1768.0	688.(

16346 rows × 10 columns

In [9]:
```
# Plotting the data distribution for every column in the training dataset
train_data.hist(figsize=(15,8))
```

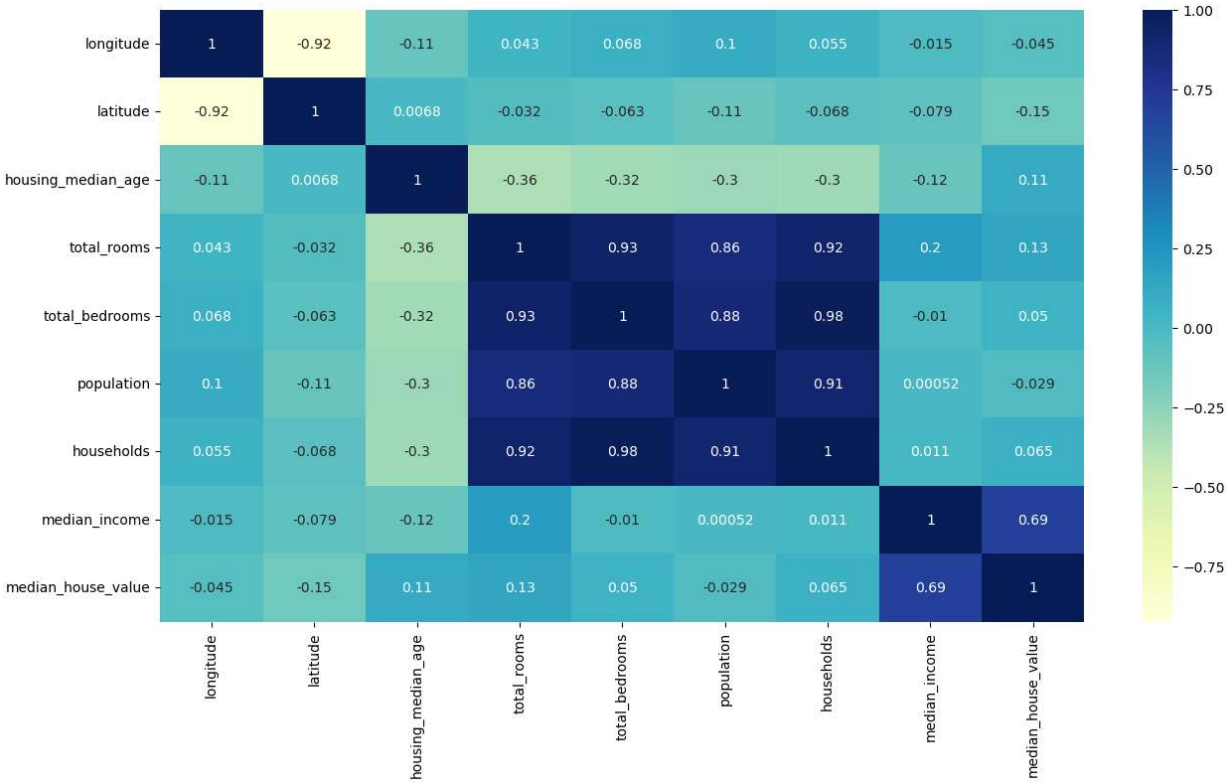
Out[9]:
```
array([[<Axes: title={'center': 'longitude'}>,
        <Axes: title={'center': 'latitude'}>,
        <Axes: title={'center': 'housing_median_age'}>],
       [<Axes: title={'center': 'total_rooms'}>,
        <Axes: title={'center': 'total_bedrooms'}>,
        <Axes: title={'center': 'population'}>],
       [<Axes: title={'center': 'households'}>,
        <Axes: title={'center': 'median_income'}>,
        <Axes: title={'center': 'median_house_value'}>]], dtype=object)
```



In [10]:
```
# Visualizing correlation matrix
plt.figure(figsize=(15,8))
sns.heatmap(train_data.corr(), annot=True, cmap="YlGnBu")
```

```
C:\Users\axays\AppData\Local\Temp\ipykernel_12072\3204835151.py:3: 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")
```

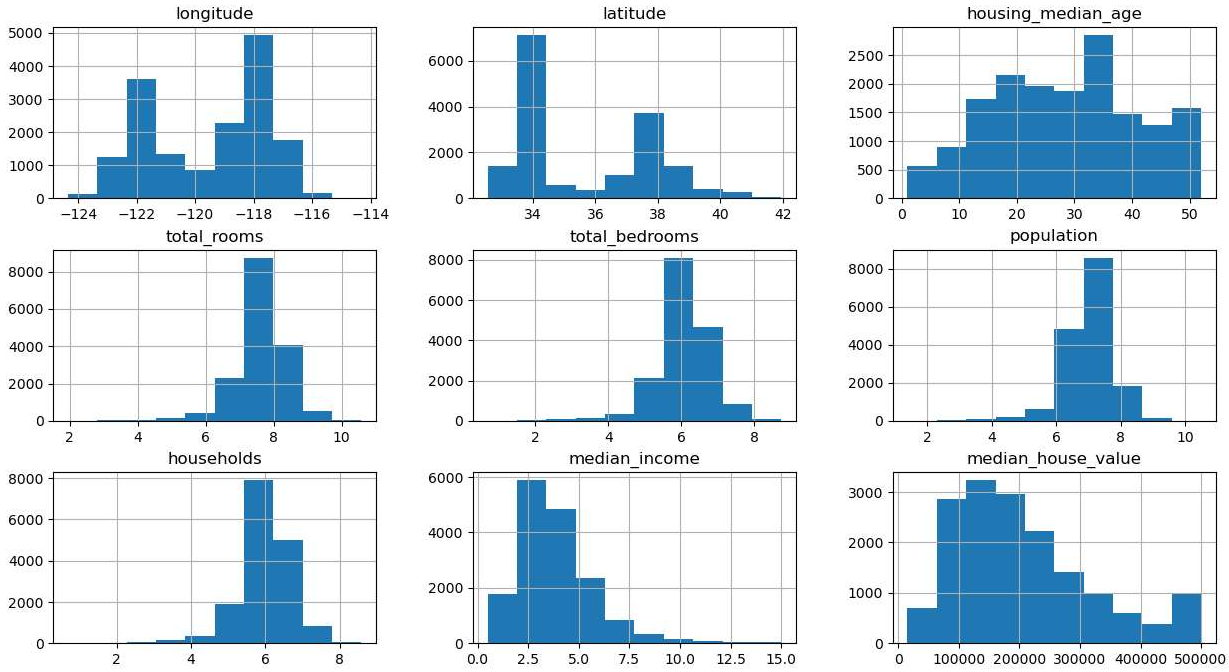
Out[10]:
```
<Axes: >
```



In [11]:
```
# As we see that the distribution of the values in these columns are skewed
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)

train_data.hist(figsize=(15,8))
```

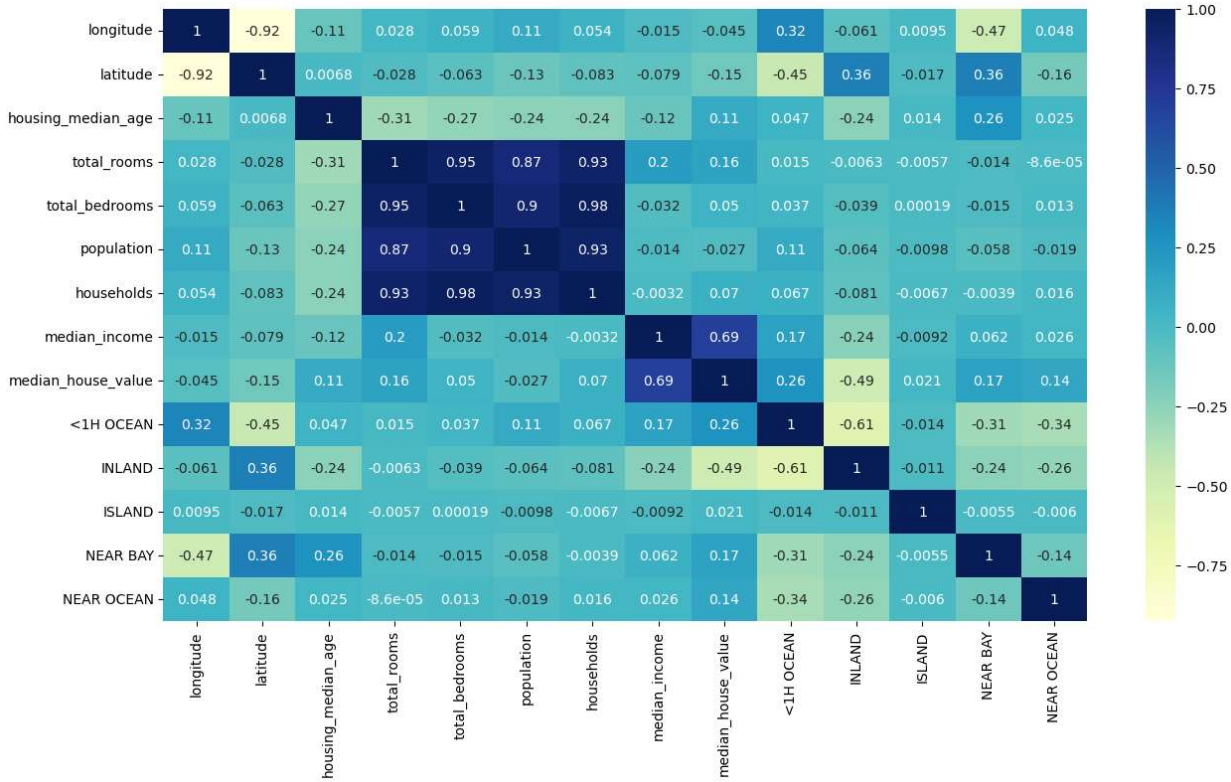
Out[11]:
```
array([[<Axes: title={'center': 'longitude'}>,
        <Axes: title={'center': 'latitude'}>,
        <Axes: title={'center': 'housing_median_age'}>],
       [<Axes: title={'center': 'total_rooms'}>,
        <Axes: title={'center': 'total_bedrooms'}>,
        <Axes: title={'center': 'population'}>],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



In [12]:
```
train_data = train_data.join(pd.get_dummies(train_data.ocean_proximity)).drop(['ocean_
```

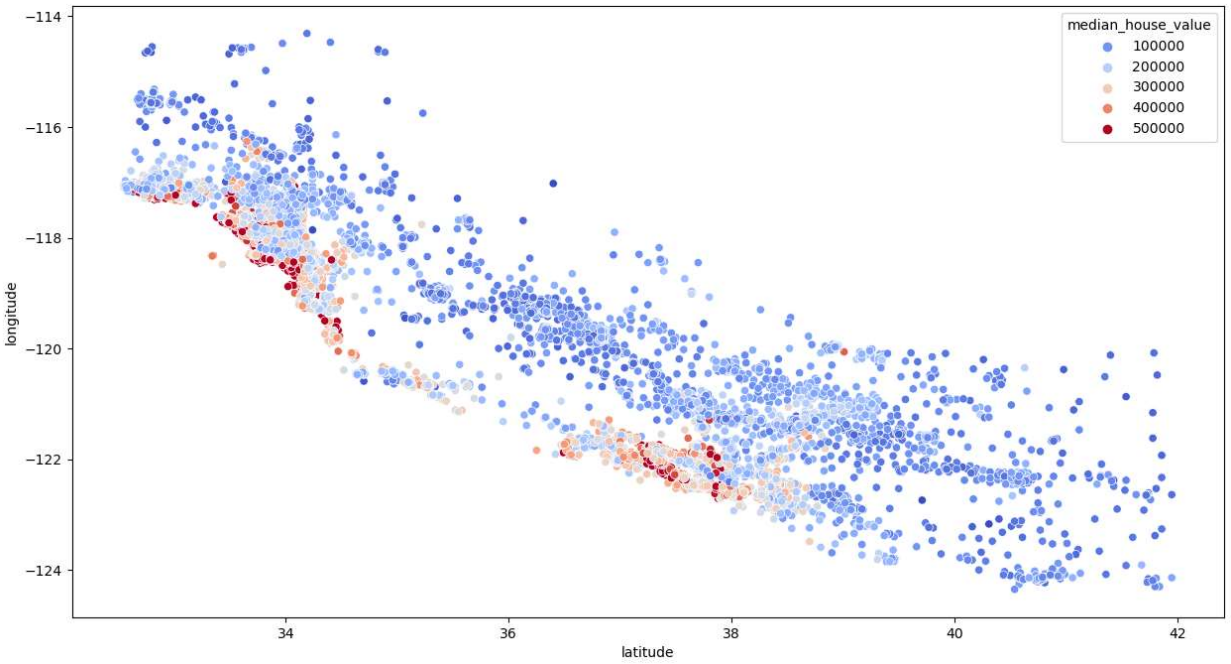
In [13]:
```
# Visualizing correlation matrix again
plt.figure(figsize=(15,8))
sns.heatmap(train_data.corr(), annot=True, cmap="YlGnBu")
```

Out[13]:
```
<Axes: >
```



In [14]:
```
# Plotting the scatterplot between latitude and longitude
plt.figure(figsize=(15,8))
sns.scatterplot(x="latitude", y="longitude", data=train_data, hue="median_house_value"
```

Out[14]:
```
<Axes: xlabel='latitude', ylabel='longitude'>
```

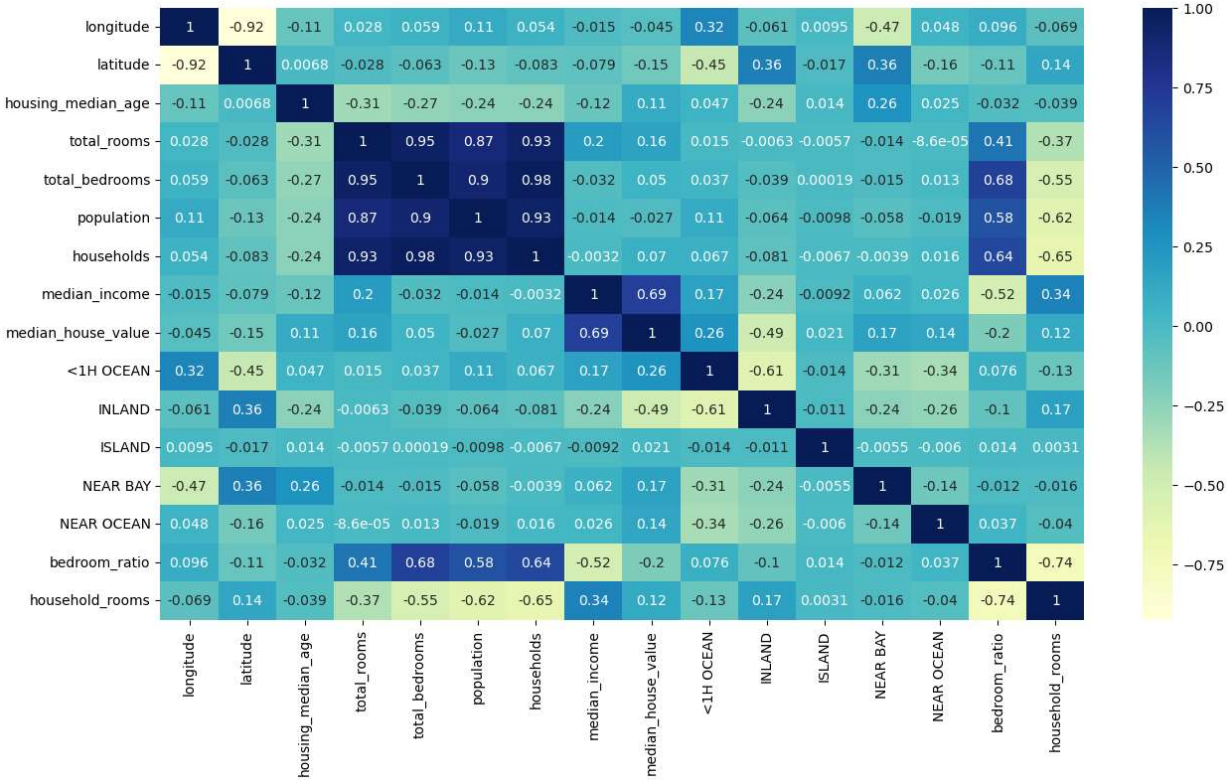


In [15]:
```
# Defining 2 new features called bedroom_ratio i.e. how many of the rooms in a household
train_data['bedroom_ratio'] = train_data['total_bedrooms'] / train_data['total_rooms']
train_data['household_rooms'] = train_data['total_rooms'] / train_data['households']
```

In [16]:
```
# Plotting a correlation heatmap with the considering the newly defined features

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

Out[16]:
```
<Axes: >
```



Building a Prediction model that determines the Median House Value using Linear Regression

In [17]:
```
# Importing LinearRegression method from the sklearn library to perform LR on our data

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# Redefining the training and test datasets with the feature additions/modifications w
X_train, Y_train = train_data.drop(['median_house_value'], axis=1), train_data['median

# below method converges the independent features in X_train
X_train_s = scaler.fit_transform(X_train)

reg = LinearRegression()
reg_s = LinearRegression()

reg.fit(X_train, Y_train)
reg_s.fit(X_train_s, Y_train)
```

Out[17]:
```
▾ LinearRegression

LinearRegression()
```

In [18]:
```
# Redefining Test dataset

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_pro

# Redefining 2 new features called bedroom_ratio i.e. how many of the rooms in a house
test_data['bedroom_ratio'] = test_data['total_bedrooms'] / test_data['total_rooms']
test_data['household_rooms'] = test_data['total_rooms'] / test_data['households']
```

In [19]:
```
test_data
```

Out[19]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	household
723	-122.10	37.67	34.0	8.205218	6.800170	7.816014	6.80682
8317	-118.32	33.34	52.0	6.904751	5.579730	5.834811	5.08140
4570	-118.27	34.06	30.0	7.479864	6.670766	7.691200	6.63987
11327	-117.95	33.76	29.0	7.512071	5.905362	7.440734	5.84064
7163	-118.17	34.04	46.0	6.559615	5.123964	6.486161	5.01063
...	...	...	...	...	...	...	...
13367	-117.58	34.00	2.0	8.928640	7.324490	7.938089	6.90975
9008	-118.60	34.08	40.0	6.765039	5.204007	5.991465	5.17615
3776	-118.42	34.16	25.0	7.926603	6.340359	7.091742	6.30261
17572	-121.92	37.32	39.0	6.729824	5.541264	6.558198	5.60947
14857	-117.08	32.64	43.0	6.913737	5.442418	6.308098	5.53338

4087 rows × 16 columns

In [20]:
```
X_test, Y_test = test_data.drop(['median_house_value'], axis=1), test_data['median_hou
```

In [21]:
```
X_test_s = scaler.fit_transform(X_test)
```

In [22]:
```
# Finding regression score before and after standardizing the features
print(reg.score(X_test, Y_test))
print(reg_s.score(X_test_s, Y_test))
```

```
0.6541077916167113
0.6543791383411723
```

Building a Prediction model that determines the Median House Value using Random Forest

In [23]:
```
# Training the data with RandomForestRegressor

from sklearn.ensemble import RandomForestRegressor

forest = RandomForestRegressor()

forest.fit(X_train_s, Y_train)
```

Out[23]:
```
RandomForestRegressor
RandomForestRegressor()
```

In [24]:
```
forest.score(X_test_s, Y_test)
```

Out[24]:
```
0.796702800304813
```

Identifying the best hyperparameters using RandomizedSearchCV for the RandomForest model

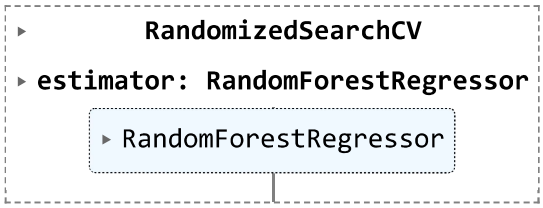
In [25]:
```
from sklearn.model_selection import RandomizedSearchCV

forest = RandomForestRegressor()

param_dist = {
    "n_estimators": [100, 200, 300],
    "min_samples_split": [2, 4],
    "max_depth": [None, 4, 8]
}

random_search = RandomizedSearchCV(forest, param_dist, n_iter=10, cv=5, scoring="neg_r
random_search.fit(X_train_s, Y_train)
```

Out[25]:



In [26]:
```
best_forest = random_search.best_estimator_
```

Finding the score for our RandomForest prediction model after hyperparameter tuning

In [27]:
```
best_forest.score(X_test_s, Y_test)
```

Out[27]:
```
0.7975522249255076
```