California housing prediction

Analysis work to be performed :-

1. Build a model of housing price to predict median housing values in california.
2. Train the model to learn from the data to predict the median housing price in any metrics.
3. Predict housing prices based on median_income and plot the regression chart for it.

In [1]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import normalize
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
df=pd.read_csv("housing_price.csv")
df.head()
#read the csv file
```

Out[2]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-114.31	34.19	15	5612	1283	1015	472	1.4936	66900
1	-114.47	34.40	19	7650	1901	1129	463	1.8200	80100
2	-114.56	33.69	17	720	174	333	117	1.6509	85700
3	-114.57	33.64	14	1501	337	515	226	3.1917	73400
4	-114.57	33.57	20	1454	326	624	262	1.9250	65500

In [3]:
```
df.describe
```

Out[3]:
```
<bound method NDFrame.describe of        longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0        -114.31     34.19                  15         5612            1283
1        -114.47     34.40                  19         7650            1901
2        -114.56     33.69                  17          720             174
3        -114.57     33.64                  14         1501             337
4        -114.57     33.57                  20         1454             326
...          ...       ...                 ...          ...             ...
16995    -124.26     40.58                  52         2217             394
16996    -124.27     40.69                  36         2349             528
16997    -124.30     41.84                  17         2677             531
16998    -124.30     41.80                  19         2672             552
16999    -124.35     40.54                  52         1820             300

       population  households  median_income  median_house_value
0            1015         472         1.4936             66900
1            1129         463         1.8200             80100
2             333         117         1.6509             85700
3             515         226         3.1917             73400
4             624         262         1.9250             65500
...           ...         ...            ...               ...
16995         907         369         2.3571            111400
16996        1194         465         2.5179             79000
16997        1244         456         3.0313            103600
16998        1298         478         1.9797             85800
16999         806         270         3.0147             94600

[17000 rows x 9 columns]>
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17000 entries, 0 to 16999
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           17000 non-null  float64
 1   latitude            17000 non-null  float64
 2   housing_median_age  17000 non-null  int64  
 3   total_rooms         17000 non-null  int64  
 4   total_bedrooms      17000 non-null  int64  
 5   population          17000 non-null  int64  
 6   households          17000 non-null  int64  
 7   median_income       17000 non-null  float64
 8   median_house_value  17000 non-null  int64  
dtypes: float64(3), int64(6)
memory usage: 1.2 MB
```

In [5]:

```
df.isnull().sum()
#checking the null values
```

Out[5]:

```
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
dtype: int64
```

In [6]:

```
from sklearn.model_selection import train_test_split
x=df.drop(['median_house_value'],axis=1)
y=df['median_house_value']
#splitting the arrays to train and test
```

In [8]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
train_data=x_train.join(y_train)
train_data
#split the dataset
```

Out[8]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
4622	-118.06	33.78	22	4048	562	1637	541	7.3463	355600
12119	-121.44	38.63	33	1077	271	753	236	1.3462	55900
3481	-117.90	34.09	34	1562	272	825	266	4.1250	220800
3152	-117.83	33.68	4	3226	838	1666	800	4.1652	184500
6895	-118.30	33.75	42	967	175	481	163	5.6611	265600
...	...	...	...	...	...	...	...	...	...
7813	-118.39	34.04	44	1873	286	635	283	5.5951	461300
10955	-120.89	37.59	33	1016	206	617	209	2.1510	195800
5192	-118.13	33.86	37	2259	425	1183	413	5.1805	201600
12172	-121.45	38.52	37	1477	321	888	312	2.5592	70300
235	-116.49	33.80	13	8789	1875	1274	688	3.7396	148900

13600 rows × 9 columns

In [9]:

```
from sklearn.preprocessing import StandardScaler
scale= StandardScaler()
scaled_data = scale.fit_transform(x_train)
print(scaled_data)
scaled_data_test = scale.fit_transform(x_test)
print(scaled_data_test)
#standardize the data
```

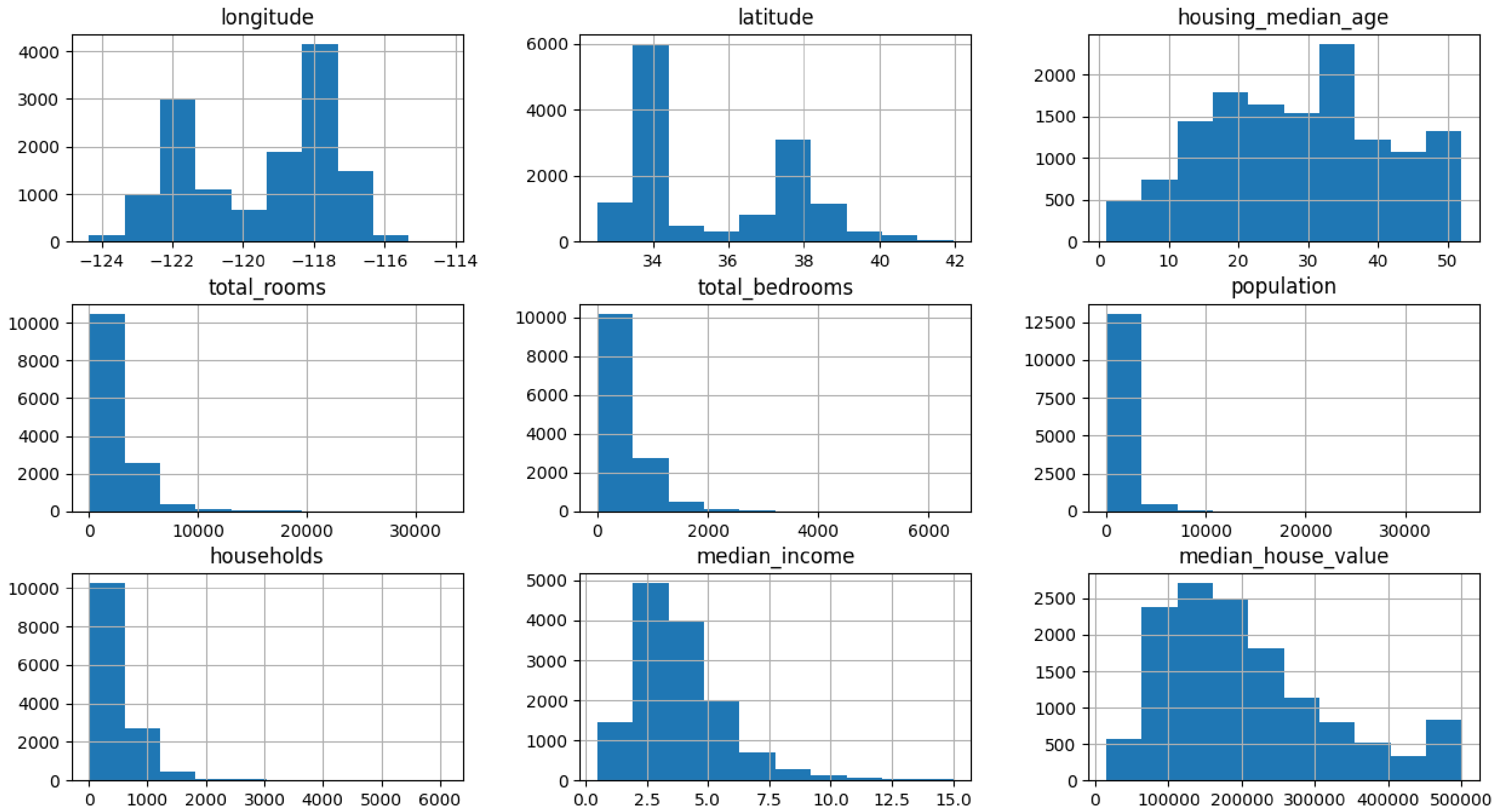
```
[[ 0.74578857 -0.85965162 -0.52696659 ...  0.17292805  0.09709981
   1.80019585]
 [-0.93873315  1.40658945  0.3440237  ... -0.5865208  -0.68892632
  -1.32111009]
 [ 0.82552925 -0.71479909  0.42320463 ... -0.52466524 -0.61161228
   0.12444664]
 ...
 [ 0.71090203 -0.82227032  0.66074743 ... -0.21710563 -0.23277345
   0.67352723]
 [-0.94371694  1.35519017  0.66074743 ... -0.47054162 -0.49306407
  -0.69009659]
 [ 1.52824393 -0.85030629 -1.239595   ... -0.13892708  0.47593863
  -0.0760419 ]]
[[ 0.75265968 -0.84540243 -0.75144152 ...  1.12557149  0.45456319
   1.2798361 ]
 [-1.17296708  0.97981431 -0.42913785 ... -0.33510864 -0.38331347
   1.25935147]
 [-1.18297033  0.78694347  0.86007681 ... -0.72758543 -0.70765282
   0.40840188]
 ...
 [ 0.74265643 -0.84540243  0.61834906 ... -0.64057407 -0.6373793
   1.12125593]
 [-1.14795894  1.92535185 -1.31547293 ...  5.65756766  7.48461869
  -1.18707455]
 [-1.34302243  1.42671016 -0.83201743 ... -0.98306561 -1.01036955
  -0.38833643]]
```

In [10]:

```
train_data.hist(figsize=(15,8))
#fetching the dataset in hist plot
```

Out[10]:

```
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 [11]:
```
train_data.corr()
#checking the correlation of data
```

Out[11]:

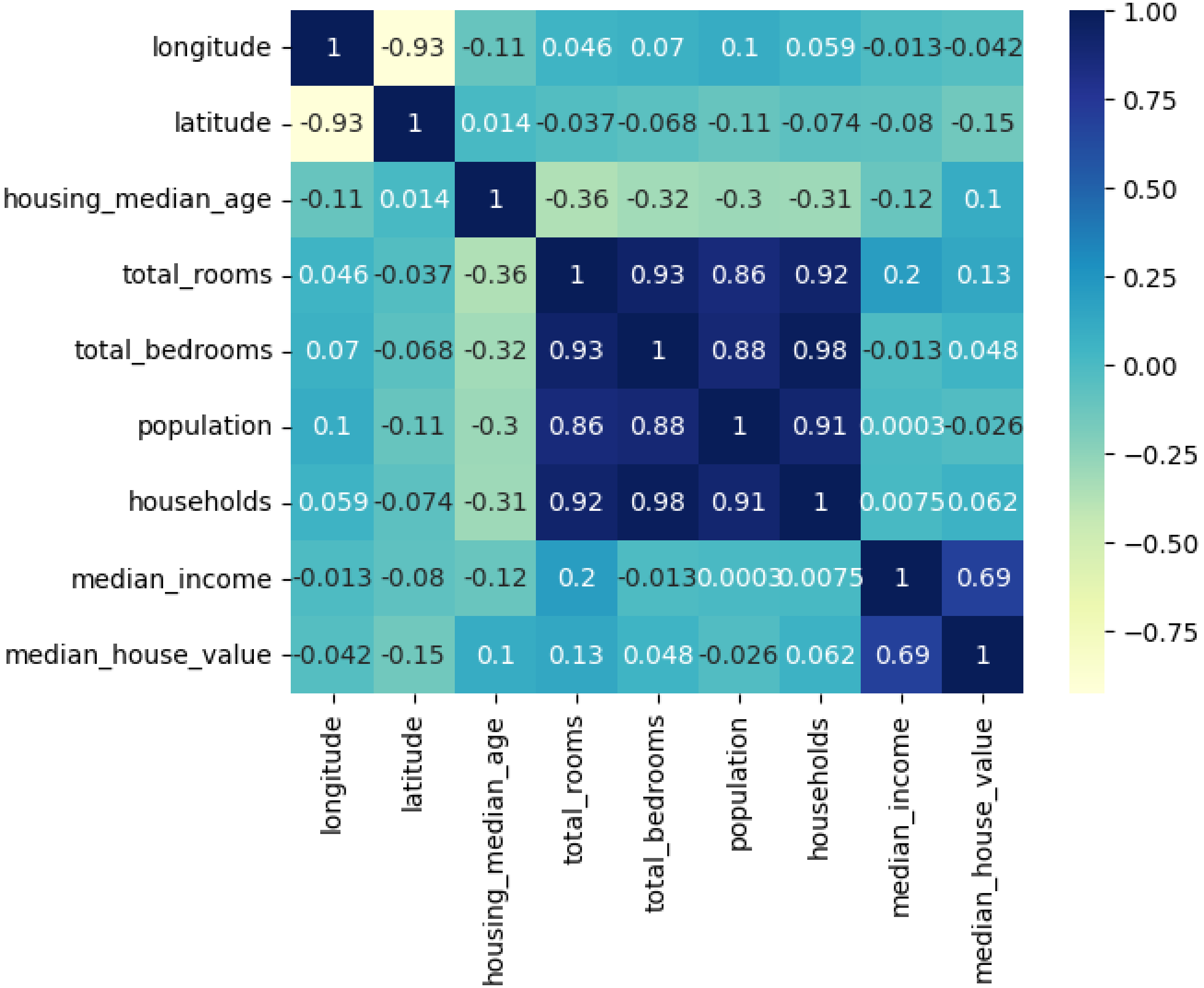
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_va
longitude	1.000000	-0.925686	-0.112210	0.046172	0.070360	0.101173	0.059395	-0.012813	-0.042
latitude	-0.925686	1.000000	0.014379	-0.037409	-0.067922	-0.110094	-0.074029	-0.080379	-0.145
housing_median_age	-0.112210	0.014379	1.000000	-0.363661	-0.322275	-0.299098	-0.305473	-0.118993	0.100
total_rooms	0.046172	-0.037409	-0.363661	1.000000	0.927664	0.861651	0.919057	0.197031	0.133
total_bedrooms	0.070360	-0.067922	-0.322275	0.927664	1.000000	0.879947	0.981719	-0.012878	0.047
population	0.101173	-0.110094	-0.299098	0.861651	0.879947	1.000000	0.907168	0.000300	-0.025
households	0.059395	-0.074029	-0.305473	0.919057	0.981719	0.907168	1.000000	0.007467	0.062
median_income	-0.012813	-0.080379	-0.118993	0.197031	-0.012878	0.000300	0.007467	1.000000	0.689
median_house_value	-0.042425	-0.145316	0.100135	0.133320	0.047825	-0.025898	0.062132	0.689975	1.000

In [12]:

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

Out[12]:

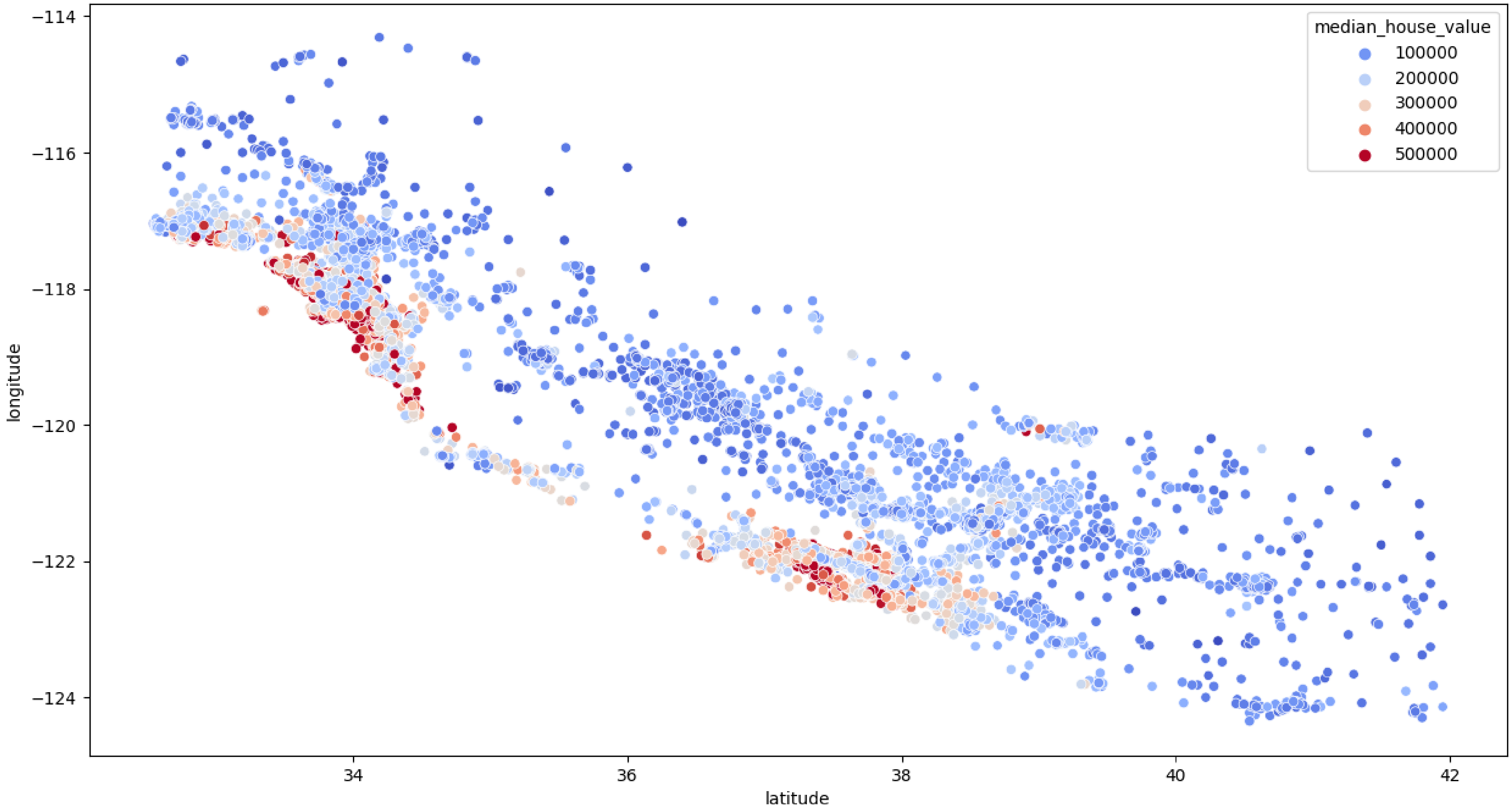
```
<Figure size 1500x800 with 0 Axes>
```




```
<Figure size 1500x800 with 0 Axes>
```

In [13]:
```python
plt.figure(figsize=(15,8))
sns.scatterplot(x='latitude',y='longitude',data=train_data,hue='median_house_value',palette='coolwarm')
```

Out[13]:
```
<AxesSubplot: xlabel='latitude', ylabel='longitude'>
```



In [14]:
```
from sklearn.linear_model import LinearRegression
x_train,y_train=train_data.drop(['median_house_value'],axis=1),train_data['median_house_value']
reg=LinearRegression()
reg.fit(x_train,y_train)
# performing linear regression , predict output for the test dataset using fitted model.
```

Out[14]:
```
▾ LinearRegression
LinearRegression()
```

In [15]:
```
test_data=x_test.join(y_test)
x_test,y_test=test_data.drop(['median_house_value'],axis=1),test_data['median_house_value']
reg.score(x_test,y_test)
#drop the median house value and check the accuracy
```

Out[15]:
```
0.6603046069190839
```

In [16]:
```
from sklearn.ensemble import RandomForestRegressor
forest=RandomForestRegressor()
forest.fit(x_train,y_train)
forest.score(x_test,y_test)
```

Out[16]:
```
0.8320425124162404
```

In [17]:
```
y_predict=reg.predict(x_test)
from sklearn.metrics import mean_squared_error
print('mean squared error is:',mean_squared_error(y_test,y_predict))
# define the mean squared error value
```

```
mean squared error is: 4475811800.058847
```

In [ ]: