In [39]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [130]:
```
from sklearn import datasets
dir(datasets)
```

In [29]:
```
from sklearn.datasets import fetch_california_housing
housing=fetch_california_housing()
housing
```

In [12]:
```
housing.keys()
#housing['data']
```

Out[12]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'feature_names', 'DESC
R'])
```

NOW CREATE DATAFRAME

In [22]:
```
import pandas as pd
housing_df=pd.DataFrame(housing['data'],
            columns=housing['feature_names'])
housing_df['MedHouseVal']=housing['target']
housing_df.head(3)
```

Out[22]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedH
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	

In [23]:
```
housing_df.isnull().sum()
```

Out[23]:
```
MedInc          0
HouseAge        0
AveRooms        0
AveBedrms       0
Population      0
AveOccup        0
Latitude        0
Longitude       0
MedHouseVal     0
dtype: int64
```

In [24]:
```
housing_df.dtypes
```

Out[24]:
```
MedInc          float64
HouseAge        float64
AveRooms        float64
AveBedrms       float64
Population      float64
AveOccup        float64
Latitude        float64
Longitude       float64
MedHouseVal     float64
dtype: object
```

STEP-1 CREATE THE LINEARREGRESSION MODEL ON THIS

- DIVIDE THE DATA IN X AND Y

In [33]:
```
X=housing_df.drop('MedHouseVal',axis=1)
Y=housing_df['MedHouseVal']
```

- NOW DIVIDE THE TRAIN & TEST DATA

In [37]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,Y
                                               ,random_state=7)
```

- CHECK THE DATASHAPE AND DATA INDEX

In [41]:
```
print(x_train.shape,y_train.shape)
```

```
(15480, 8) (15480,)
```

In [42]:
```
print(x_test.shape,y_test.shape)
```

```
(5160, 8) (5160,)
```

- NOW LETS CHECK DOS REALLY DIVIDE INT 25% AND 75%

In [43]:
```
25*len(housing_df)/100
```

Out[43]:
```
5160.0
```

In [44]:
```
75*len(housing_df)/100
```

Out[44]:
```
15480.0
```

- NOW WE WILL TRAIN THE MODEL

In [46]:
```
from sklearn.linear_model import LinearRegression
LR=LinearRegression()
LR.fit(x_train,y_train)
```

Out[46]:
```
LinearRegression()
```

- NOW PREDICTIONS

In [48]:
```
Y_per=LR.predict(x_test)
```

In [49]:
```
Y_per
```

Out[49]:
```
array([1.64674141, 2.47861172, 2.42657918, ..., 1.9109084 , 2.03570794,
       2.79079465])
```

- NOW LETS CHECK WEATHER OUR Y_TEST & Y_PER ARE SAME OR NOT

In [54]:
```
print(y_test.shape,Y_per.shape)
```

```
(5160,) (5160,)
```

In [55]:
```
df=pd.DataFrame()
df['y_test']=y_test
df['y_per']=Y_per
```

In [56]:
```
df
```

Out[56]:

	y_test	y_per
4648	3.60000	1.646741
8740	3.36000	2.478612
162	2.69900	2.426579
15735	2.87500	2.014797
18380	5.00001	4.322450
...	...	...
16012	5.00001	4.202564
7002	1.58200	1.887380
6899	2.20500	1.910908
18576	1.21400	2.035708
6740	3.59900	2.790795

5160 rows × 2 columns

- NOW LETS DO EVALUATION METRCS

In [59]:
```
import numpy as np
```

In [61]:
```
from sklearn.metrics import r2_score,mean_squared_error
rsqr=r2_score(y_test,Y_per)
mse=mean_squared_error(y_test,Y_per)
RMSE=np.sqrt(mse)
print('the rsquare is ',rsqr)
print('the mse is ',mse)
print('the rmse is ',RMSE)
```

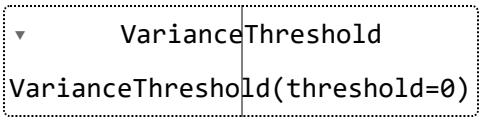
```
the rsquare is  0.6070242241118249
the mse is  0.523752986617112
the rmse is  0.7237078047230885
```

- NOW LETS DO FEATURE SELECTION

- VARIANCE TRESHOLD

In [64]:
```
from sklearn.feature_selection import VarianceThreshold
VC=VarianceThreshold(threshold=0)
VC.fit(X)
```

Out[64]:



In [65]:
```
VC.variances_
```

Out[65]:
```
array([3.60914769e+00, 5.10000000e+01, 6.12123614e+00, 2.24580619e-01,
       3.56790000e+04, 1.07864799e+02, 4.56207160e+00, 4.01394488e+00])
```

In [68]:
```
VC.get_support()
```

Out[68]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True])
```

In [69]:
```
VC.get_feature_names_out()
```

Out[69]:
```
array(['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population',
       'AveOccup', 'Latitude', 'Longitude'], dtype=object)
```

- MUTUAL INFO CLASSIFER

In [123]:
```
from sklearn.feature_selection import mutual_info_regression
MIC=mutual_info_regression(X,Y)
MIC
df=pd.DataFrame()
df['columns']=X.columns
df['INFORMATION_GAIN']=MIC
df
```

Out[123]:

	columns	INFORMATION_GAIN
0	MedInc	0.387632
1	HouseAge	0.032595
2	AveRooms	0.103479
3	AveBedrms	0.023618
4	Population	0.021496
5	AveOccup	0.072457
6	Latitude	0.370496
7	Longitude	0.401510

- P-VALUE

In [115]:
```
from statsmodels.api import OLS
OLS(Y,X).fit().summary()
```

Out[115]:

OLS Regression Results

Dep. Variable:	MedHouseVal	**R-squared (uncentered):**	0.892
Model:	OLS	**Adj. R-squared (uncentered):**	0.892
Method:	Least Squares	**F-statistic:**	2.137e+04
Date:	Wed, 17 Apr 2024	**Prob (F-statistic):**	0.00
Time:	12:56:25	**Log-Likelihood:**	-24087.
No. Observations:	20640	**AIC:**	4.819e+04
Df Residuals:	20632	**BIC:**	4.825e+04
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
MedInc	0.5135	0.004	120.594	0.000	0.505	0.522
HouseAge	0.0157	0.000	33.727	0.000	0.015	0.017
AveRooms	-0.1825	0.006	-29.673	0.000	-0.195	-0.170
AveBedrms	0.8651	0.030	28.927	0.000	0.806	0.924
Population	7.792e-06	5.09e-06	1.530	0.126	-2.19e-06	1.78e-05
AveOccup	-0.0047	0.001	-8.987	0.000	-0.006	-0.004
Latitude	-0.0639	0.004	-17.826	0.000	-0.071	-0.057
Longitude	-0.0164	0.001	-14.381	0.000	-0.019	-0.014

Omnibus:	4353.392	**Durbin-Watson:**	0.909
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	14087.489
Skew:	1.069	**Prob(JB):**	0.00
Kurtosis:	6.436	**Cond. No.**	1.03e+04

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 1.03e+04. This might indicate that there are strong multicollinearity or other numerical problems.

In [116]:
```
#population is having more p-value so we can drop this column
```

- VIF

In [129]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
df=pd.DataFrame()
df['columns']=X.columns
df['VIF']=[variance_inflation_factor(X,i) for i in range(len(X.columns))]
df
```

Out[129]:

	columns	VIF
0	MedInc	11.511140
1	HouseAge	7.195917
2	AveRooms	45.993601
3	AveBedrms	43.590314
4	Population	2.935745
5	AveOccup	1.095243
6	Latitude	559.874071
7	Longitude	633.711654

In [ ]:
```
# we can drop the columns which are having the vif above 10
```

- NOW LETS SAVE THE MODEL

In [83]:
```
import pickle as pkl
```

In [84]:
```
pkl.dump(LR,
         open('LR_HOSUNG.PKL','wb'))
```

- LETS OPEN THE MODEL

In [86]:
```
MODEL=pkl.load(open('LR_HOSUNG.PKL','rb'))
```

In [87]:
```
MODEL
```

Out[87]:
```
LinearRegression
LinearRegression()
```

- NOW LETS TEST THE MODEL

In [89]:
```
y_per=MODEL.predict(x_test)
```

In [107]:
```
df=pd.DataFrame()
df['Y_PER BY 1ST']=Y_per
df['y_per by Load_pickel']=y_per
df['y_per_postman']=y_per_postman
```

In [110]:
```
df.head(5)# all are same
```

Out[110]:

	Y_PER BY 1ST	y_per by Load_pickel	y_per_postman
0	1.646741	1.646741	1.646741
1	2.478612	2.478612	2.478612
2	2.426579	2.426579	2.426579
3	2.014797	2.014797	2.014797
4	4.322450	4.322450	4.322450

- DONE

In [98]:
```
y_per_postman=[1.6467414120592707,2.478611717350006,2.4265791769745277,2.01479
```

In [96]:
```
x_test.to_csv('test_data1.csv',index=False)
```

In [106]:
```
test_df=pd.read_csv('test_data1.csv')
test_df.head(2)
```

Out[106]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	2.0278	31.0	2.846928	1.091641	3107.0	3.128902	34.06	-118.31
1	4.3056	30.0	5.036932	1.011364	905.0	2.571023	33.81	-118.31

In [ ]:

In [ ]:

In [ ]: