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

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

```
df = pd.read_csv('/content/BostonHousing.csv')
df
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9

506 rows × 14 columns

```
df.isnull().sum()
```

	0
crim	0
zn	0
indus	0
chas	0
nox	0
rm	5
age	0
dis	0
rad	0
tax	0
ptratio	0
b	0
lstat	0
medv	0

dtype: int64

```
df['rm'] = df['rm'].fillna(df['rm'].mean())
```

```
df.columns
```

```
Index(['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'b', 'lstat', 'medv'],
      dtype='object')
```

```
x = df[['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'b', 'lstat']]
y = df[  'medv']


x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)


model = LinearRegression()
model.fit(x_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred=model.predict(x_test)


y_pred
```

```
array([28.82239584, 35.99603416, 15.09228245, 25.22034225, 18.87953301,
       23.21045939, 17.58938357, 14.29516645, 23.05962597, 20.60597442,
       24.79156482, 18.68781897, -6.97029537, 21.83401717, 19.20507393,
       26.27813307, 20.55086169,  5.66348432, 40.41183713, 17.64392229,
       27.30510418, 30.03661805, 11.14086477, 24.0914498 , 17.89366267,
       15.80205231, 22.94370814, 14.25673851, 22.26791641, 19.23579151,
       22.25904121, 25.22873441, 25.67890247, 18.00008838, 16.70518236,
       17.13274168, 31.18319423, 20.16619825, 23.71425877, 24.7786056 ,
       13.93555271, 31.98051639, 42.52615489, 17.44537599, 27.1307741 ,
       17.07962694, 13.88267891, 26.04977923, 20.37219815, 29.96892294,
       21.37151548, 34.31366662, 15.87099228, 26.15194641, 39.49293053,
       22.84450351, 18.95220195, 32.67601472, 25.00352327, 12.92044639,
       20.67837288, 30.54139508, 31.58535337, 15.90452705, 20.52706895,
       16.51387993, 20.4986864 , 25.9949182 , 30.62708393, 11.43313558,
       20.52847936, 27.56285765, 10.85306835, 15.98341973, 23.86842182,
        5.66419446, 21.45790602, 41.27110222, 18.55870022,  9.10009182,
       20.97669092, 13.06145524, 21.01248621,  9.34763448, 23.12906582,
       31.78982219, 19.10102537, 25.57623879, 29.13970987, 20.16918415,
       25.58097968,  5.20522737, 20.16633787, 15.09180847, 12.88579183,
       20.80741473, 24.68376669, -0.76799732, 13.33690195, 15.61891927,
       22.20011624, 24.57604373, 10.77905183, 19.4893846 , 23.23800179,
       11.77033442, 18.35489911, 25.42193401, 20.87981383, 24.10064283,
        7.36466825, 19.15421904, 21.92792631, 27.38632316, 32.49027919,
       14.87174688, 35.02399177, 12.85456759, 20.8142438 , 28.41670133,
       15.67730363, 24.66814714,  3.28649267, 23.79235367, 25.72187428,
       23.03753525, 24.74374103])
```

```
model.score(x_train,y_train)
# accuracy
```

```
0.7475676233088484
```

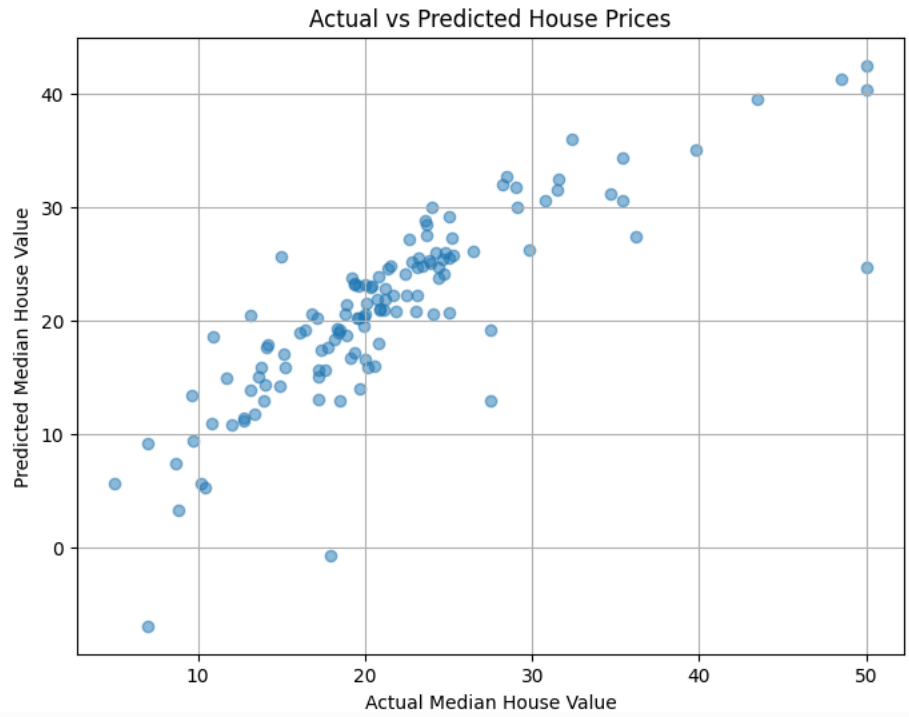
```
model.score(x_test,y_test)
```

```
0.6831144311098885
```

```
#  mean error
np.sqrt(mean_squared_error(y_test, y_pred))
```

```
np.float64(4.710689042447866)
```

```
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel("Actual Median House Value")
plt.ylabel("Predicted Median House Value")
plt.title("Actual vs Predicted House Prices")
plt.grid(True)
plt.show()
```


Actual vs Predicted House Prices
Predicted Median House Value
Actual Median House Value
0
10
20
30
40
10
20
30
40
50