In [31]:

```
import pandas as pd
import numpy as np
```

In [32]:

```
from sklearn.datasets import fetch_california_housing
boston = fetch_california_housing()
```

In [33]:

```
data = pd.DataFrame(boston.data)
```

In [34]:

```
data.head()
```

Out[34]:

	0	1	2	3	4	5	6	7
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
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

In [36]:

```
data.columns = boston.feature_names
```

In [37]:

```
data['PRICE'] = boston.target
```

In [39]:

```
data.head()
```

Out[39]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	PRI
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.5
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.5
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.5
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.4
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.4

In [40]:

```
data.isnull().sum()
```

Out[40]:

```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
Latitude      0
Longitude     0
PRICE         0
dtype: int64
```

In [45]:

```
data.describe()
```

Out[45]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	

In [44]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   MedInc      20640 non-null  float64
 1   HouseAge    20640 non-null  float64
 2   AveRooms    20640 non-null  float64
 3   AveBedrms   20640 non-null  float64
 4   Population  20640 non-null  float64
 5   AveOccup    20640 non-null  float64
 6   Latitude    20640 non-null  float64
 7   Longitude   20640 non-null  float64
 8   PRICE       20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

In [47]:

```
x = data.iloc[:, :-1]
y = data.PRICE
```

In [50]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 4
```

In [51]:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
```

In [52]:

```
regressor.fit(x_train, y_train)
```

Out[52]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [53]:

```
y_pred = regressor.predict(x_test)
```

In [54]:

```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(rmse)
```

```
0.7245753833536993
```

In [57]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [58]:

```
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [60]:

```
import keras
from keras.layers import Activation, Dense, Dropout
from keras.models import Sequential
```

In [61]:

```
model = Sequential()
model.add(Dense(128, activation='relu', input_dim = 8))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(1))
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

In [62]:

```
model.fit(x_train, y_train, epochs = 100)
```

```
Epoch 1/100
516/516 [==============================] - 4s 4ms/step - loss: 0.6903
Epoch 2/100
516/516 [==============================] - 1s 2ms/step - loss: 0.3691
Epoch 3/100
516/516 [==============================] - 1s 2ms/step - loss: 0.3379
Epoch 4/100
516/516 [==============================] - 2s 3ms/step - loss: 0.3585
Epoch 5/100
516/516 [==============================] - 2s 4ms/step - loss: 0.3109
Epoch 6/100
516/516 [==============================] - 2s 4ms/step - loss: 0.2993
Epoch 7/100
516/516 [==============================] - 1s 2ms/step - loss: 0.2931
Epoch 8/100
516/516 [==============================] - 1s 2ms/step - loss: 0.2914
Epoch 9/100
516/516 [==============================] - 1s 2ms/step - loss: 0.2927
Epoch 10/100
```

In [63]:

```
y_pred = model.predict(x_test)
```

```
129/129 [==============================] - 1s 3ms/step
```

In [64]:

```
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
```

In [65]:

```
rmse
```

Out[65]:

```
0.5211309857354227
```

In [ ]: