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

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

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

```
data.head()
```

	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

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

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

```
data.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	PRICE
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

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

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

```
data.describe()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	P
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.00
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35.631861	-119.569704	2.06
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2.135952	2.003532	1.15
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32.540000	-124.350000	0.14
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33.930000	-121.800000	1.19
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34.260000	-118.490000	1.79
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37.710000	-118.010000	2.64
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	41.950000	-114.310000	5.00

```
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
```

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

```
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)
```

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

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

```
▾ LinearRegression
LinearRegression()
```

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

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

```
0.7245753833536993
```

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

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

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

```
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')
```

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

```
Epoch 84/100
516/516 [==============================] - 1s 2ms/step - loss: 0.1675
Epoch 85/100
516/516 [==============================] - 1s 2ms/step - loss: 0.1671
Epoch 86/100
516/516 [==============================] - 1s 3ms/step - loss: 0.1676
Epoch 87/100
516/516 [==============================] - 1s 2ms/step - loss: 0.1652
Epoch 88/100
516/516 [==============================] - 1s 2ms/step - loss: 0.1655
Epoch 89/100
516/516 [==============================] - 3s 5ms/step - loss: 0.1632
Epoch 90/100
516/516 [==============================] - 2s 4ms/step - loss: 0.1654
Epoch 91/100
516/516 [==============================] - 2s 3ms/step - loss: 0.1648
Epoch 92/100
516/516 [==============================] - 1s 2ms/step - loss: 0.1619
Epoch 93/100
516/516 [==============================] - 1s 3ms/step - loss: 0.1601
Epoch 94/100
516/516 [==============================] - 1s 3ms/step - loss: 0.1593
Epoch 95/100
516/516 [==============================] - 1s 3ms/step - loss: 0.1629
Epoch 96/100
516/516 [==============================] - 1s 2ms/step - loss: 0.1589
Epoch 97/100
516/516 [==============================] - 1s 2ms/step - loss: 0.1594
Epoch 98/100
516/516 [==============================] - 1s 2ms/step - loss: 0.1579
Epoch 99/100
516/516 [==============================] - 2s 3ms/step - loss: 0.1548
Epoch 100/100
516/516 [==============================] - 2s 4ms/step - loss: 0.1558
<keras.callbacks.History at 0x7f935c149f60>
```

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

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

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

```
rmse
```

```
0.5211309857354227
```

✓ 0s completed at 5:43 PM