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

```
data = pd.read_csv('/content/HousingData.csv')
```

```
data.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LST
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.9
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.(
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.9
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	Na

Next steps: Generate code with `data` View recommended plots

```
data.columns
```

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MEDV'],
      dtype='object')
```

```
data.shape
```

```
(506, 14)
```

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

```
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```

```
data.dropna(inplace=True)
```

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

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

```
data.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AG
count	394.000000	394.000000	394.000000	394.000000	394.000000	394.000000	394.00000
mean	3.690136	11.460660	11.000863	0.068528	0.553215	6.280015	68.93274
std	9.202423	23.954082	6.908364	0.252971	0.113112	0.697985	27.88870
min	0.006320	0.000000	0.460000	0.000000	0.389000	3.561000	2.90000
25%	0.081955	0.000000	5.130000	0.000000	0.453000	5.879250	45.47500
50%	0.268880	0.000000	8.560000	0.000000	0.538000	6.201500	77.70000
75%	3.435973	12.500000	18.100000	0.000000	0.624000	6.605500	94.25000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.00000

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 394 entries, 0 to 504
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     394 non-null    float64
 1   ZN       394 non-null    float64
 2   INDUS    394 non-null    float64
 3   CHAS     394 non-null    float64
 4   NOX      394 non-null    float64
 5   RM       394 non-null    float64
 6   AGE      394 non-null    float64
 7   DIS      394 non-null    float64
 8   RAD      394 non-null    int64  
 9   TAX      394 non-null    int64  
 10  PTRATIO  394 non-null    float64
```

```
 11  B        394 non-null    float64
 12  LSTAT    394 non-null    float64
 13  MEDV     394 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 46.2 KB
```

```
from sklearn.preprocessing import StandardScaler
# Split the data into input and output variables
X = data.drop('MEDV', axis=1)
y =data['MEDV']
# Scale the input features
scaler =StandardScaler()
X =scaler.fit_transform(X)
```

```
from sklearn.model_selection import train_test_split
# Splitthe data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
print('Training set shape:', X_train.shape, y_train.shape)
print('Testing set shape:', X_test.shape, y_test.shape)
```

```
Training set shape: (275, 13) (275,)
Testing set shape: (119, 13) (119,)
```

```
import tensorflow.keras as keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

```
# Define the model architecture
model = Sequential()

# Input layer
model.add(Dense(1, input_dim=13, activation='linear'))
```

```
# Display the model summary
print(model.summary())
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 1)                 14

=================================================================
Total params: 14 (56.00 Byte)
Trainable params: 14 (56.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

```
# Compile the model
model.compile(optimizer = 'adam',loss ='mean_squared_error',metrics=['mae'])

x_val=X_train
y_val=y_train


history = model.fit(X_train,y_train,epochs=20,batch_size=512,validation_data=(x_val
```

```
Epoch 1/20
1/1 [==============================] - 0s 382ms/step - loss: 541.2143 - mae: 21.8711
Epoch 2/20
1/1 [==============================] - 0s 25ms/step - loss: 541.0932 - mae: 21.8704 -
Epoch 3/20
1/1 [==============================] - 0s 24ms/step - loss: 540.9720 - mae: 21.8696 -
Epoch 4/20
1/1 [==============================] - 0s 25ms/step - loss: 540.8513 - mae: 21.8689 -
Epoch 5/20
1/1 [==============================] - 0s 25ms/step - loss: 540.7305 - mae: 21.8681 -
Epoch 6/20
1/1 [==============================] - 0s 25ms/step - loss: 540.6099 - mae: 21.8674 -
Epoch 7/20
1/1 [==============================] - 0s 24ms/step - loss: 540.4896 - mae: 21.8666 -
Epoch 8/20
1/1 [==============================] - 0s 25ms/step - loss: 540.3694 - mae: 21.8659 -
Epoch 9/20
1/1 [==============================] - 0s 25ms/step - loss: 540.2493 - mae: 21.8651 -
Epoch 10/20
1/1 [==============================] - 0s 24ms/step - loss: 540.1295 - mae: 21.8644 -
Epoch 11/20
1/1 [==============================] - 0s 25ms/step - loss: 540.0098 - mae: 21.8636 -
Epoch 12/20
1/1 [==============================] - 0s 43ms/step - loss: 539.8902 - mae: 21.8629 -
Epoch 13/20
1/1 [==============================] - 0s 27ms/step - loss: 539.7709 - mae: 21.8621 -
Epoch 14/20
1/1 [==============================] - 0s 27ms/step - loss: 539.6517 - mae: 21.8613 -
Epoch 15/20
1/1 [==============================] - 0s 30ms/step - loss: 539.5328 - mae: 21.8606 -
Epoch 16/20
1/1 [==============================] - 0s 29ms/step - loss: 539.4140 - mae: 21.8598 -
Epoch 17/20
1/1 [==============================] - 0s 28ms/step - loss: 539.2955 - mae: 21.8591 -
Epoch 18/20
1/1 [==============================] - 0s 24ms/step - loss: 539.1771 - mae: 21.8583 -
Epoch 19/20
1/1 [==============================] - 0s 28ms/step - loss: 539.0588 - mae: 21.8576 -
Epoch 20/20
1/1 [==============================] - 0s 27ms/step - loss: 538.9408 - mae: 21.8568 -
```

```
results = model.evaluate(X_test, y_test)
```

```
4/4 [==============================] - 0s 2ms/step - loss: 628.3030 - mae: 23.4571
```

```
Start coding or generate with AI.
```