In [1]:

```
import tensorflow as tf
from tensorflow.keras.datasets import boston_housing
from sklearn import preprocessing
```

In [2]:

```
(train_x,train_y),(test_x,test_y)=boston_housing.load_data()
```

In [3]:

```
print('Train Shape :',train_x.shape)
print('Test Shape :',test_x.shape)
print('Actual Train Shape :',train_y.shape)
print('Actual Test Shape :',test_y.shape)
```

```
Train Shape : (404, 13)
Test Shape : (102, 13)
Actual Train Shape : (404,)
Actual Test Shape : (102,)
```

In [4]:

```
train_x[0]
```

Out[4]:

```
array([  1.23247,   0.     ,   8.14   ,   0.     ,   0.538  ,   6.142  ,
        91.7    ,   3.9769 ,   4.     , 307.     ,  21.     , 396.9    ,
        18.72   ])
```

In [5]:

```
train_y[0]
```

Out[5]:

```
15.2
```

In [6]:

```
train_x=preprocessing.normalize(train_x)
test_x=preprocessing.normalize(test_x)
```

In [7]:

```
train_x[0]
```

Out[7]:

```
array([0.0024119 , 0.        , 0.01592969, 0.        , 0.00105285,
       0.01201967, 0.17945359, 0.00778265, 0.00782786, 0.6007879 ,
       0.04109624, 0.77671895, 0.03663436])
```

In [8]:

```
train_y[0]
```

Out[8]:

```
15.2
```

In [9]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import *
```

In [10]:

```
def housepricepredmodel():
    model=Sequential()
    model.add(Dense(128,activation='relu',input_shape=(train_x[0].shape)))
    model.add(Dense(64,activation='relu'))
    model.add(Dense(32,activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='rmsprop',loss='mse',metrics=['mae'])
    return model
```

In [11]:

```
import numpy as np
k=4
num_val_samples = len(train_x)
num_epochs=100
all_scores=[]
```

In [12]:

```
model = housepricepredmodel()
history=model.fit(x=train_x,y=train_y,epochs=num_epochs,batch_size=1,verbose=1,validatio
```

```
- mae: 3.9735 - val_loss: 28.2779 - val_mae: 3.8152
Epoch 27/100
404/404 [==============================] - 1s 3ms/step - loss: 31.7178
- mae: 3.9010 - val_loss: 28.7385 - val_mae: 4.0316
Epoch 28/100
404/404 [==============================] - 1s 3ms/step - loss: 29.3280
- mae: 3.8723 - val_loss: 33.0439 - val_mae: 3.9401
Epoch 29/100
404/404 [==============================] - 1s 4ms/step - loss: 29.5122
- mae: 3.8025 - val_loss: 33.8570 - val_mae: 4.7513
Epoch 30/100
404/404 [==============================] - 2s 5ms/step - loss: 26.9461
- mae: 3.7111 - val_loss: 26.5773 - val_mae: 3.8765
Epoch 31/100
404/404 [==============================] - 2s 5ms/step - loss: 26.4445
- mae: 3.6411 - val_loss: 29.4456 - val_mae: 4.3102
Epoch 32/100
404/404 [==============================] - 2s 4ms/step - loss: 26.5311
- mae: 3.6764 - val_loss: 33.6430 - val_mae: 4.0089
Epoch 33/100
```

In [15]:

```
mse_nn, mae_nn = model.evaluate(test_x, test_y)

print('Mean squared error on test data: ', mse_nn)
print('Mean absolute error on test data: ', mae_nn)
```

```
4/4 [==============================] - 0s 5ms/step - loss: 28.0184 - mae:
3.5639
Mean squared error on test data:  28.018367767333984
Mean absolute error on test data:  3.5639488697052
```

In [ ]:

In [ ]: