dl1

April 21, 2024

Boston House Price Prediction Name : Nidhi Kamath

[26]:
```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
from sklearn import preprocessing
```

[38]:
```
(X_train, Y_train), (X_test, Y_test) = keras.datasets.boston_housing.load_data()
```

[28]:
```
print("Training data shape:", X_train.shape)
print("Test data shape:", X_test.shape)
print("Train output data shape:", Y_train.shape)
print("Actual Test output data shape:", Y_test.shape)
```

```
Training data shape: (404, 13)
Test data shape: (102, 13)
Train output data shape: (404,)
Actual Test output data shape: (102,)
```

[29]:
```
X_train = preprocessing.normalize(X_train)
X_test = preprocessing.normalize(X_test)
```

[30]:
```
model = Sequential([
    Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)
])
```

```
/home/nidhi/.local/lib/python3.10/site-
packages/keras/src/layers/core/dense.py:86: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

[31]:
```
model.summary()
```

```
Model: "sequential_11"
```

Layer (type)	Output Shape	Param #
dense_44 (Dense)	(None, 128)	1,792
dense_45 (Dense)	(None, 64)	8,256
dense_46 (Dense)	(None, 32)	2,080
dense_47 (Dense)	(None, 1)	33

```
Total params: 12,161 (47.50 KB)

Trainable params: 12,161 (47.50 KB)

Non-trainable params: 0 (0.00 B)
```

[32]:
```
model.compile(loss='mse', optimizer='rmsprop', metrics=['mae'])
```

[33]:
```
history = model.fit(X_train, Y_train, epochs=20, batch_size=1, verbose=1,␣
↪validation_data=(X_test, Y_test))
```

```
Epoch 1/20
404/404             1s 2ms/step -
loss: 257.4678 - mae: 12.2965 - val_loss: 79.8688 - val_mae: 6.3300
Epoch 2/20
404/404             1s 1ms/step -
loss: 81.2376 - mae: 6.1475 - val_loss: 60.4605 - val_mae: 5.6386
Epoch 3/20
404/404             1s 1ms/step -
loss: 58.2065 - mae: 5.2785 - val_loss: 56.4742 - val_mae: 5.5420
Epoch 4/20
404/404             1s 1ms/step -
loss: 63.3948 - mae: 5.5304 - val_loss: 55.2899 - val_mae: 5.4546
Epoch 5/20
404/404             1s 2ms/step -
loss: 63.0134 - mae: 5.4990 - val_loss: 61.6034 - val_mae: 5.5417
Epoch 6/20
404/404             1s 2ms/step -
loss: 56.0785 - mae: 4.9579 - val_loss: 55.9747 - val_mae: 5.3536
Epoch 7/20
404/404             1s 2ms/step -
```

```
loss: 64.0252 - mae: 5.4795 - val_loss: 61.3780 - val_mae: 5.5002
Epoch 8/20
404/404             1s 1ms/step -
loss: 48.8235 - mae: 4.5386 - val_loss: 52.6091 - val_mae: 5.1388
Epoch 9/20
404/404             1s 1ms/step -
loss: 48.1088 - mae: 4.7712 - val_loss: 59.7066 - val_mae: 5.3833
Epoch 10/20
404/404             1s 2ms/step -
loss: 56.3099 - mae: 5.0725 - val_loss: 48.1332 - val_mae: 4.9502
Epoch 11/20
404/404             1s 2ms/step -
loss: 47.2650 - mae: 4.6020 - val_loss: 47.3174 - val_mae: 4.8223
Epoch 12/20
404/404             1s 2ms/step -
loss: 36.3078 - mae: 4.0888 - val_loss: 77.9395 - val_mae: 7.7030
Epoch 13/20
404/404             1s 2ms/step -
loss: 53.5331 - mae: 5.2450 - val_loss: 44.5192 - val_mae: 4.6327
Epoch 14/20
404/404             1s 2ms/step -
loss: 50.3486 - mae: 4.8726 - val_loss: 43.0127 - val_mae: 4.5491
Epoch 15/20
404/404             1s 2ms/step -
loss: 52.6901 - mae: 5.0294 - val_loss: 50.7954 - val_mae: 4.9646
Epoch 16/20
404/404             1s 2ms/step -
loss: 38.6620 - mae: 4.1478 - val_loss: 45.2976 - val_mae: 4.6517
Epoch 17/20
404/404             1s 2ms/step -
loss: 45.5645 - mae: 4.5826 - val_loss: 37.2901 - val_mae: 4.2180
Epoch 18/20
404/404             1s 2ms/step -
loss: 32.5251 - mae: 4.0921 - val_loss: 37.0078 - val_mae: 4.1849
Epoch 19/20
404/404             1s 2ms/step -
loss: 37.1011 - mae: 4.0821 - val_loss: 56.6494 - val_mae: 6.6335
Epoch 20/20
404/404             1s 2ms/step -
loss: 30.4977 - mae: 3.9257 - val_loss: 43.6990 - val_mae: 5.5365
```

[35]:
```
test_input1 = np.array([(0.0024119, 0.0, 0.01592969, 0.0, 0.00105285, 0.
↪01201967, 0.17945359, 0.00778265, 0.00782786, 0.6007879, 0.04109624, 0.
↪77671895, 0.03663436)])
```

```
test_input2 = np.array([(4.07923050e-05, 1.54587284e-01, 3.80378407e-03, 0.0, 7.
↪77620881e-04, 1.42595058e-02, 2.94184285e-02, 1.17486336e-02, 3.
↪74757051e-03, 6.52077269e-01, 2.75446433e-02, 7.40857215e-01, 5.
↪82747215e-03)])
```

[36]:
```
print("Actual output: 15.2")
print("Predicted Output:", model.predict(test_input1))

print("Actual output: 42.3")
print("Predicted Output:", model.predict(test_input2))
```

```
Actual output: 15.2
1/1             0s 63ms/step
Predicted Output: [[22.11968]]
Actual output: 42.3
1/1             0s 19ms/step
Predicted Output: [[35.09147]]
```

[ ]: