Boston_Housing - Regression Analysis

Import TensorFlow & Keras

```
import warnings
warnings.filterwarnings('ignore')
```

- import TensorFlow

```
import tensorflow as tf

tf.__version__
```

```
'2.4.1'
```

- GPU 설정 Off

```
tf.test.gpu_device_name()
```

```
''
```

- import Keras

```
import keras

keras.__version__
```

```
'2.4.3'
```

I. Boston_Housing Data_Set Load & Review

1) Load Boston_Housing Data_Set

```
from keras.datasets import boston_housing

(train_data, train_targets), (X_test, y_test) =  boston_housing.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/boston_hous
57344/57026 [==============================] - 0s 0us/step
```

2) Data_Set Information

```
print(train_data.shape)
print(X_test.shape)

print(train_targets[:10])
print(y_test[:10])
```

```
(404, 13)
(102, 13)
[15.2 42.3 50.  21.1 17.7 18.5 11.3 15.6 15.6 14.4]
[ 7.2 18.8 19.  27.  22.2 24.5 31.2 22.9 20.5 23.2]
```

II. Data Preprocessing

1) Standardization

- train_data & test_data

```
mean = train_data.mean(axis = 0)
std = train_data.std(axis = 0)

train_data = train_data - mean
train_data = train_data / std

X_test = X_test - mean
X_test = X_test / std
```

2) Train & Validation Split

```
from sklearn.model_selection import train_test_split

X_train, X_valid, y_train, y_valid = train_test_split(train_data, train_targets,
                                                      test_size = 0.2,
                                                      random_state = 2045)

X_train.shape, X_valid.shape, y_train.shape, y_valid.shape
```

```
((323, 13), (81, 13), (323,), (81,))
```

III. Boston_Housing Keras Modeling

1) Model Define

```
from keras import models
from keras import layers

boston = models.Sequential(name = 'Regression')
boston.add(layers.Dense(64, activation = 'relu', input_shape = (13,)))
boston.add(layers.Dense(64, activation = 'relu'))
boston.add(layers.Dense(1))
```

```
boston.summary()
```

```
Model: "Regression"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                896
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 5,121
Trainable params: 5,121
Non-trainable params: 0
_________________________________________________________________
```

2) Model Compile

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

3) Model Fit

- 약 4분

```
%%time

Hist_boston = boston.fit(X_train, y_train,
                         epochs = 500,
                         batch_size = 1,
                         validation_data = (X_valid, y_valid))
```

```
Epoch 1/500
323/323 [==============================] - 2s 3ms/step - loss: 408.0891 - mae: 17.3323 - v
Epoch 2/500
323/323 [==============================] - 0s 1ms/step - loss: 35.8454 - mae: 4.2989 - val
```

```
Epoch 3/500
323/323 [==============================] - 0s 1ms/step - loss: 25.5345 - mae: 3.2788 - val
Epoch 4/500
323/323 [==============================] - 0s 1ms/step - loss: 15.0729 - mae: 2.6414 - val
Epoch 5/500
323/323 [==============================] - 0s 1ms/step - loss: 11.4394 - mae: 2.4952 - val
Epoch 6/500
323/323 [==============================] - 1s 2ms/step - loss: 9.9953 - mae: 2.2301 - val_
Epoch 7/500
323/323 [==============================] - 0s 1ms/step - loss: 10.1875 - mae: 2.3232 - val
Epoch 8/500
323/323 [==============================] - 0s 1ms/step - loss: 13.7809 - mae: 2.5423 - val
Epoch 9/500
323/323 [==============================] - 0s 1ms/step - loss: 10.1890 - mae: 2.2230 - val
Epoch 10/500
323/323 [==============================] - 0s 1ms/step - loss: 9.3636 - mae: 2.1011 - val_
Epoch 11/500
323/323 [==============================] - 0s 1ms/step - loss: 14.1553 - mae: 2.4500 - val
Epoch 12/500
323/323 [==============================] - 0s 2ms/step - loss: 10.3239 - mae: 2.2110 - val
Epoch 13/500
323/323 [==============================] - 0s 1ms/step - loss: 11.5812 - mae: 2.1928 - val
Epoch 14/500
323/323 [==============================] - 0s 1ms/step - loss: 9.4291 - mae: 2.2010 - val_
Epoch 15/500
323/323 [==============================] - 1s 2ms/step - loss: 9.7199 - mae: 2.0890 - val_
Epoch 16/500
323/323 [==============================] - 0s 1ms/step - loss: 9.3342 - mae: 2.0159 - val_
Epoch 17/500
323/323 [==============================] - 0s 1ms/step - loss: 9.2149 - mae: 2.0570 - val_
Epoch 18/500
323/323 [==============================] - 0s 1ms/step - loss: 6.9626 - mae: 1.7980 - val_
Epoch 19/500
323/323 [==============================] - 0s 2ms/step - loss: 8.3578 - mae: 2.0070 - val_
Epoch 20/500
323/323 [==============================] - 0s 1ms/step - loss: 9.9050 - mae: 1.9170 - val_
Epoch 21/500
323/323 [==============================] - 0s 2ms/step - loss: 11.9065 - mae: 2.1559 - val
Epoch 22/500
323/323 [==============================] - 0s 1ms/step - loss: 8.5232 - mae: 2.0219 - val_
Epoch 23/500
323/323 [==============================] - 0s 2ms/step - loss: 7.3518 - mae: 1.8154 - val_
Epoch 24/500
323/323 [==============================] - 0s 1ms/step - loss: 8.6240 - mae: 2.1033 - val_
Epoch 25/500
323/323 [==============================] - 0s 1ms/step - loss: 7.4729 - mae: 1.8477 - val_
Epoch 26/500
323/323 [==============================] - 0s 1ms/step - loss: 7.3017 - mae: 1.8235 - val_
Epoch 27/500
323/323 [==============================] - 0s 1ms/step - loss: 9.1445 - mae: 1.9920 - val_
Epoch 28/500
323/323 [==============================] - 0s 2ms/step - loss: 9.7588 - mae: 1.9800 - val_
Epoch 29/500
323/323 [==============================] - 1s 4ms/step - loss: 7.8428 - mae: 1.8383 - val_
```

▾ 4) Model Evaluate

```
test_mse_score, test_mae_score = boston.evaluate(X_test, y_test)

print('MAE is :',test_mae_score)
```

```
4/4 [==============================] - 0s 3ms/step - loss: 15.3165 - mae: 2.7720
MAE is : 2.7720444202423096
```

5) Visualization

- 전체 시각화

```
import matplotlib.pyplot as plt

epochs = range(1, len(Hist_boston.history['val_mae']) + 1)

plt.figure(figsize = (9, 6))
plt.plot(epochs, Hist_boston.history['val_mae'])
plt.title('Validation MAE')
plt.xlabel('Epochs')
plt.ylabel('Mean Absolute Error')
plt.grid()
plt.show()
```

- 5번째 이후 MAE 확인

```
def smooth_curve(points, factor=0.9):
  smoothed_points = []
  for point in points:
    if smoothed_points:
      previous = smoothed_points[-1]
      smoothed_points.append(previous * factor + point * (1 - factor))
    else:
      smoothed_points.append(point)
  return smoothed_points

mae_history = Hist_boston.history['val_mae']

mae_history = smooth_curve(mae_history[5:])

plt.figure(figsize = (9, 6))
plt.plot(range(1, len(mae_history) + 1), mae_history)
plt.title('Validation MAE')
plt.xlabel('Epochs')
plt.ylabel('Mean Absolute Error')
plt.grid()
plt.show()
```

6) Keras Session Clear

```
from keras import backend as K

K.clear_session()
```

IV. Early Stopping

1) Model Define & Compile

```
from keras import models
from keras import layers

boston = models.Sequential(name = 'EarlyStopping')
boston.add(layers.Dense(64, activation = 'relu', input_shape = (13,)))
boston.add(layers.Dense(64, activation = 'relu'))
boston.add(layers.Dense(1))

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

2) EarlyStopping( )

- monitor : 모니터링 대상 성능
- mode : 모니터링 대상을 최소화(min) 또는 최대화(max)
- patience : 성능이 개선되지 않는 epoch 횟수

```
from keras.callbacks import EarlyStopping

es = EarlyStopping(monitor = 'val_mae',
                   mode = 'min',
                   patience = 50,
                   verbose = 1)
```

3) ModelCheckpoint( )

- 'best_boston.h5' : 최적모델이 저장될 경로
- save_best_only : 최적모델만 저장할지 지정

```
from keras.callbacks import ModelCheckpoint

mc = ModelCheckpoint('best_boston.h5',
                     monitor = 'val_mae',
                     mode = 'min',
                     save_best_only = True,
                     verbose = 1)
```

4) Model Fit with callbacks

- callbacks : Earlystopping( ) 과 ModelCheckpoint( ) 객체 지정

```
%%time

Hist_boston = boston.fit(X_train, y_train,
                         epochs = 500,
                         batch_size = 1,
                         validation_data = (X_valid, y_valid),
                         callbacks = [es, mc],
                         verbose = 1)
```

```
Epoch 1/500
323/323 [==============================] - 1s 2ms/step - loss: 375.5133 - mae: 16.7263 - v

Epoch 00001: val_mae improved from inf to 4.19915, saving model to best_boston.h5
Epoch 2/500
323/323 [==============================] - 0s 1ms/step - loss: 31.0737 - mae: 4.0960 - val

Epoch 00002: val_mae improved from 4.19915 to 3.22446, saving model to best_boston.h5
Epoch 3/500
323/323 [==============================] - 0s 1ms/step - loss: 20.8790 - mae: 3.2386 - val

Epoch 00003: val_mae did not improve from 3.22446
Epoch 4/500
323/323 [==============================] - 0s 1ms/step - loss: 13.8037 - mae: 2.6871 - val

Epoch 00004: val_mae improved from 3.22446 to 2.80099, saving model to best_boston.h5
Epoch 5/500
323/323 [==============================] - 0s 1ms/step - loss: 22.3158 - mae: 3.0859 - val

Epoch 00005: val_mae improved from 2.80099 to 2.50964, saving model to best_boston.h5
Epoch 6/500
323/323 [==============================] - 0s 1ms/step - loss: 11.5438 - mae: 2.3494 - val

Epoch 00006: val_mae did not improve from 2.50964
Epoch 7/500
323/323 [==============================] - 0s 1ms/step - loss: 11.8471 - mae: 2.4197 - val

Epoch 00007: val_mae did not improve from 2.50964
Epoch 8/500
323/323 [==============================] - 0s 1ms/step - loss: 14.1425 - mae: 2.4430 - val

Epoch 00008: val_mae improved from 2.50964 to 2.41554, saving model to best_boston.h5
Epoch 9/500
323/323 [==============================] - 0s 1ms/step - loss: 8.0779 - mae: 2.0357 - val_

Epoch 00009: val_mae improved from 2.41554 to 2.27911, saving model to best_boston.h5
Epoch 10/500
323/323 [==============================] - 0s 1ms/step - loss: 9.3922 - mae: 2.1384 - val_
```

```
Epoch 00010: val_mae did not improve from 2.27911
Epoch 11/500
323/323 [==============================] - 0s 1ms/step - loss: 8.3917 - mae: 1.9231 - val_

Epoch 00011: val_mae did not improve from 2.27911
Epoch 12/500
323/323 [==============================] - 0s 1ms/step - loss: 16.4750 - mae: 2.4027 - val

Epoch 00012: val_mae improved from 2.27911 to 2.20807, saving model to best_boston.h5
Epoch 13/500
323/323 [==============================] - 0s 1ms/step - loss: 14.1810 - mae: 2.2891 - val

Epoch 00013: val_mae did not improve from 2.20807
Epoch 14/500
323/323 [==============================] - 0s 1ms/step - loss: 12.8692 - mae: 2.2880 - val

Epoch 00014: val_mae did not improve from 2.20807
Epoch 15/500
323/323 [==============================] - 0s 1ms/step - loss: 12.8152 - mae: 2.1686 - val
```

5) Best Model

```
!ls -l
```

```
total 76
-rw-r--r-- 1 root root 70296 Mar 19 05:31 best_boston.h5
drwxr-xr-x 1 root root  4096 Mar  5 14:37 sample_data
```

6) Model Evaluate

```
test_mse_score, test_mae_score = boston.evaluate(X_test, y_test)

print('MAE is :',test_mae_score)
```

```
4/4 [==============================] - 0s 2ms/step - loss: 16.0990 - mae: 2.5802
MAE is : 2.5802247524261475
```

#

#

#

The End

#

#

#