Boston_Housing - Regression Analysis

Import TensorFlow

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

- import TensorFlow

```
import tensorflow as tf

tf.__version__
```

```
'2.5.0'
```

- GPU 설정 Off

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

```
''
```

I. Boston_Housing Data_Set Load & Review

1) Load Boston_Housing Data_Set

```
from tensorflow.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_housing.npz
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 tensorflow.keras import models
from tensorflow.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 [==============================] - 1s 2ms/step - loss: 206.3569 - mae: 10.5232 - val_loss: 55.0467 - val_mae: 4.6740
Epoch 2/500
323/323 [==============================] - 0s 1ms/step - loss: 31.7885 - mae: 4.0144 - val_loss: 29.7434 - val_mae: 3.2614
Epoch 3/500
323/323 [==============================] - 0s 1ms/step - loss: 21.0897 - mae: 3.2275 - val_loss: 23.4591 - val_mae: 2.9676
Epoch 4/500
323/323 [==============================] - 0s 1ms/step - loss: 17.4675 - mae: 2.9070 - val_loss: 23.9720 - val_mae: 2.6921
Epoch 5/500
323/323 [==============================] - 0s 1ms/step - loss: 14.9011 - mae: 2.5350 - val_loss: 18.9767 - val_mae: 2.6660
Epoch 6/500
323/323 [==============================] - 0s 1ms/step - loss: 14.0015 - mae: 2.4218 - val_loss: 20.7499 - val_mae: 2.6891
Epoch 7/500
323/323 [==============================] - 0s 1ms/step - loss: 12.6803 - mae: 2.3742 - val_loss: 20.7302 - val_mae: 2.8262
Epoch 8/500
323/323 [==============================] - 0s 1ms/step - loss: 12.2892 - mae: 2.3268 - val_loss: 16.7822 - val_mae: 2.6706
Epoch 9/500
323/323 [==============================] - 0s 1ms/step - loss: 11.7068 - mae: 2.2676 - val_loss: 18.5376 - val_mae: 2.4919
Epoch 10/500
323/323 [==============================] - 0s 1ms/step - loss: 11.7264 - mae: 2.3026 - val_loss: 16.9605 - val_mae: 2.5034
Epoch 11/500
323/323 [==============================] - 0s 1ms/step - loss: 10.6383 - mae: 2.1892 - val_loss: 14.1962 - val_mae: 2.3968
Epoch 12/500
323/323 [==============================] - 0s 1ms/step - loss: 10.5795 - mae: 2.1634 - val_loss: 15.3658 - val_mae: 2.6810
Epoch 13/500
323/323 [==============================] - 0s 1ms/step - loss: 10.4747 - mae: 2.1286 - val_loss: 15.1358 - val_mae: 2.3433
Epoch 14/500
323/323 [==============================] - 0s 1ms/step - loss: 10.0656 - mae: 2.1155 - val_loss: 20.1966 - val_mae: 2.6726
Epoch 15/500
323/323 [==============================] - 0s 1ms/step - loss: 10.1617 - mae: 2.0973 - val_loss: 14.2747 - val_mae: 2.4898
Epoch 16/500
323/323 [==============================] - 1s 2ms/step - loss: 9.7994 - mae: 2.1224 - val_loss: 16.1126 - val_mae: 2.4896
Epoch 17/500
323/323 [==============================] - 1s 2ms/step - loss: 9.3140 - mae: 2.0364 - val_loss: 16.1220 - val_mae: 2.4216
Epoch 18/500
323/323 [==============================] - 0s 1ms/step - loss: 9.1157 - mae: 2.0464 - val_loss: 18.5794 - val_mae: 2.4914
Epoch 19/500
323/323 [==============================] - 0s 1ms/step - loss: 9.0841 - mae: 2.0059 - val_loss: 13.3270 - val_mae: 2.3076
Epoch 20/500
323/323 [==============================] - 0s 1ms/step - loss: 9.3135 - mae: 2.0385 - val_loss: 13.5742 - val_mae: 2.2280
Epoch 21/500
323/323 [==============================] - 1s 2ms/step - loss: 9.0800 - mae: 1.9463 - val_loss: 17.8132 - val_mae: 2.7220
Epoch 22/500
323/323 [==============================] - 0s 1ms/step - loss: 8.9187 - mae: 1.9428 - val_loss: 13.2486 - val_mae: 2.3096
Epoch 23/500
323/323 [==============================] - 0s 1ms/step - loss: 8.4834 - mae: 1.9038 - val_loss: 13.0267 - val_mae: 2.2537
Epoch 24/500
323/323 [==============================] - 0s 1ms/step - loss: 8.4881 - mae: 1.9092 - val_loss: 12.1227 - val_mae: 2.4194
Epoch 25/500
323/323 [==============================] - 0s 1ms/step - loss: 8.6540 - mae: 1.8688 - val_loss: 14.9188 - val_mae: 2.3019
Epoch 26/500
323/323 [==============================] - 1s 2ms/step - loss: 8.2998 - mae: 1.9225 - val_loss: 12.3937 - val_mae: 2.2767
Epoch 27/500
323/323 [==============================] - 0s 1ms/step - loss: 7.9866 - mae: 1.9043 - val_loss: 13.1696 - val_mae: 2.3215
Epoch 28/500
323/323 [==============================] - 0s 1ms/step - loss: 7.9872 - mae: 1.8588 - val_loss: 12.9914 - val_mae: 2.3923
Epoch 29/500
323/323 [==============================] - 0s 1ms/step - loss: 7.5418 - mae: 1.8566 - val_loss: 12.3483 - val_mae: 2.3998
Epoch 30/500
323/323 [==============================] - 0s 1ms/step - loss: 7.4755 - mae: 1.8694 - val_loss: 15.5297 - val_mae: 2.4485
```

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: 13.1888 - mae: 2.6355
MAE is : 2.6355042457580566
```

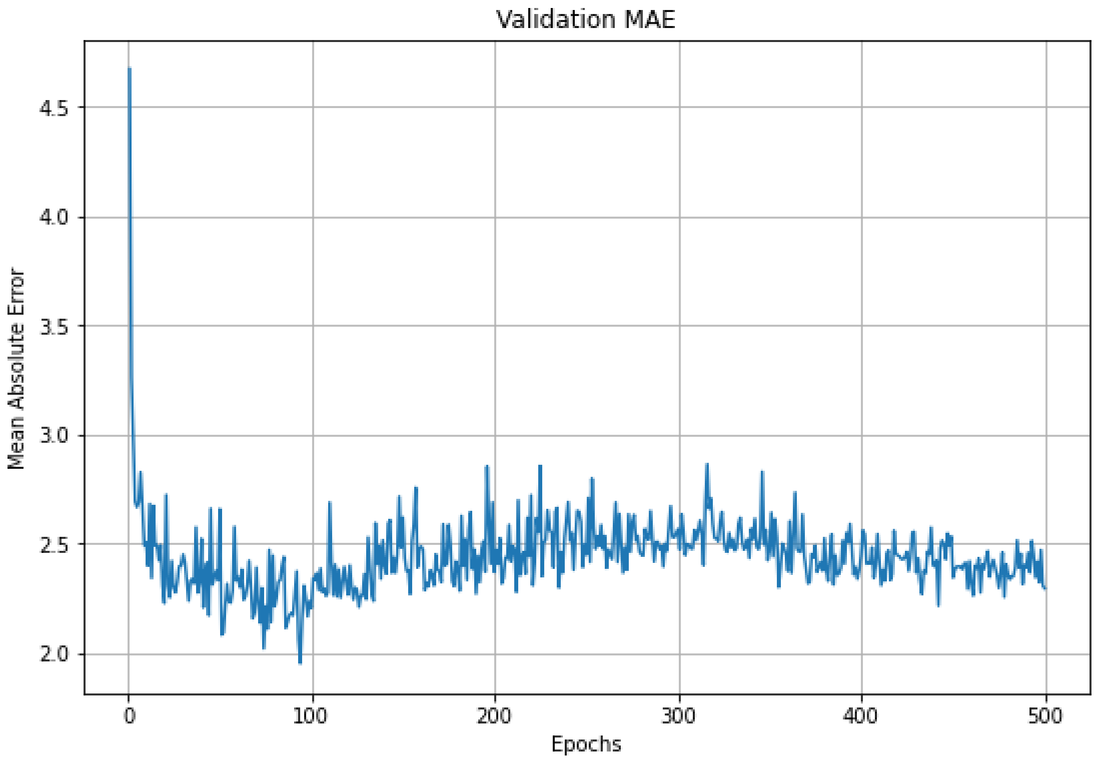
5) Visualization

- 전체 시각화

```
import matplotlib.pyplot as plt

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

plt.figure(figsize = (9, 6))
plt.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 tensorflow.keras import backend as K

K.clear_session()
```

IV. Early Stopping

1) Model Define & Compile

```
from tensorflow.keras import models
from tensorflow.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 tensorflow.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 tensorflow.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,
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: 174.6352 - mae: 9.5986 - val_loss: 41.4591 - val_mae: 3.9775

Epoch 00001: val_mae improved from inf to 3.97753, saving model to best_boston.h5
Epoch 2/500
323/323 [==============================] - 0s 1ms/step - loss: 26.0134 - mae: 3.5166 - val_loss: 34.0707 - val_mae: 3.4632

Epoch 00002: val_mae improved from 3.97753 to 3.46316, saving model to best_boston.h5
Epoch 3/500
323/323 [==============================] - 0s 1ms/step - loss: 20.9524 - mae: 3.1276 - val_loss: 24.6288 - val_mae: 2.9403

Epoch 00003: val_mae improved from 3.46316 to 2.94029, saving model to best_boston.h5
Epoch 4/500
323/323 [==============================] - 0s 1ms/step - loss: 16.0361 - mae: 2.8049 - val_loss: 20.7016 - val_mae: 2.5512

Epoch 00004: val_mae improved from 2.94029 to 2.55120, saving model to best_boston.h5
Epoch 5/500
323/323 [==============================] - 0s 1ms/step - loss: 14.6006 - mae: 2.5936 - val_loss: 18.9381 - val_mae: 2.4038

Epoch 00005: val_mae improved from 2.55120 to 2.40377, saving model to best_boston.h5
Epoch 6/500
323/323 [==============================] - 0s 1ms/step - loss: 13.4802 - mae: 2.3926 - val_loss: 18.4286 - val_mae: 2.4781

Epoch 00006: val_mae did not improve from 2.40377
Epoch 7/500
323/323 [==============================] - 0s 1ms/step - loss: 12.2325 - mae: 2.3327 - val_loss: 17.5996 - val_mae: 2.7025

Epoch 00007: val_mae did not improve from 2.40377
Epoch 8/500
323/323 [==============================] - 0s 1ms/step - loss: 11.8126 - mae: 2.1980 - val_loss: 17.8016 - val_mae: 2.3399

Epoch 00008: val_mae improved from 2.40377 to 2.33989, saving model to best_boston.h5
Epoch 9/500
323/323 [==============================] - 0s 1ms/step - loss: 11.5974 - mae: 2.1922 - val_loss: 19.6174 - val_mae: 2.5759

Epoch 00009: val_mae did not improve from 2.33989
Epoch 10/500
323/323 [==============================] - 0s 1ms/step - loss: 11.0843 - mae: 2.2556 - val_loss: 19.9826 - val_mae: 2.5556

Epoch 00010: val_mae did not improve from 2.33989
Epoch 11/500
323/323 [==============================] - 0s 1ms/step - loss: 10.9461 - mae: 2.1911 - val_loss: 16.9740 - val_mae: 2.4868

Epoch 00011: val_mae did not improve from 2.33989
Epoch 12/500
323/323 [==============================] - 0s 1ms/step - loss: 10.4827 - mae: 2.1062 - val_loss: 13.7914 - val_mae: 2.2074

Epoch 00012: val_mae improved from 2.33989 to 2.20738, saving model to best_boston.h5
Epoch 13/500
323/323 [==============================] - 0s 1ms/step - loss: 10.1270 - mae: 2.1304 - val_loss: 15.4937 - val_mae: 2.4149

Epoch 00013: val_mae did not improve from 2.20738
Epoch 14/500
323/323 [==============================] - 0s 1ms/step - loss: 9.7104 - mae: 2.0834 - val_loss: 14.5782 - val_mae: 2.2716

Epoch 00014: val_mae did not improve from 2.20738
Epoch 15/500
323/323 [==============================] - 0s 1ms/step - loss: 9.5792 - mae: 2.0759 - val_loss: 13.1434 - val_mae: 2.2645

Epoch 00015: val_mae did not improve from 2.20738
```

5) Best Model

```
!ls -l
```

```
total 76
-rw-r--r-- 1 root root 70280 Jul 20 07:49 best_boston.h5
drwxr-xr-x 1 root root  4096 Jul 16 13:20 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 4ms/step - loss: 15.3469 - mae: 2.4857
MAE is : 2.4856765270233154
```

#

#

#

The End

#

#

#