Boston_Housing - Regression Analysis

Import TensorFlow

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

- import TensorFlow

```
import tensorflow as tf

tf.__version__
```

```
'2.6.0'
```

- 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
65536/57026 [==================================] - 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))
```

```
                  validation_data = (X_valid, y_valid))
```

```
Epoch 1/500
323/323 [==============================] - 1s 2ms/step - loss: 169.1796 - mae: 9.2839 - val_loss: 49.8417 - val_mae: 4.7335
Epoch 2/500
323/323 [==============================] - 0s 1ms/step - loss: 27.6938 - mae: 3.7014 - val_loss: 32.0510 - val_mae: 3.2380
Epoch 3/500
323/323 [==============================] - 0s 1ms/step - loss: 20.5296 - mae: 3.1082 - val_loss: 26.7319 - val_mae: 2.8775
Epoch 4/500
323/323 [==============================] - 0s 1ms/step - loss: 17.1432 - mae: 2.8202 - val_loss: 23.8753 - val_mae: 2.6984
Epoch 5/500
323/323 [==============================] - 0s 1ms/step - loss: 15.2803 - mae: 2.6192 - val_loss: 23.9531 - val_mae: 2.7549
Epoch 6/500
323/323 [==============================] - 0s 1ms/step - loss: 13.9756 - mae: 2.5121 - val_loss: 21.6628 - val_mae: 2.5448
Epoch 7/500
323/323 [==============================] - 0s 1ms/step - loss: 13.1017 - mae: 2.4134 - val_loss: 21.2226 - val_mae: 2.6390
Epoch 8/500
323/323 [==============================] - 0s 1ms/step - loss: 12.0634 - mae: 2.3589 - val_loss: 18.0113 - val_mae: 2.4361
Epoch 9/500
323/323 [==============================] - 0s 1ms/step - loss: 11.5901 - mae: 2.2416 - val_loss: 15.0613 - val_mae: 2.3809
Epoch 10/500
323/323 [==============================] - 0s 1ms/step - loss: 11.3197 - mae: 2.2767 - val_loss: 16.7939 - val_mae: 2.4738
Epoch 11/500
323/323 [==============================] - 0s 1ms/step - loss: 10.8267 - mae: 2.2766 - val_loss: 15.9401 - val_mae: 2.4892
Epoch 12/500
323/323 [==============================] - 0s 1ms/step - loss: 10.6656 - mae: 2.1654 - val_loss: 15.2449 - val_mae: 2.5322
Epoch 13/500
323/323 [==============================] - 0s 1ms/step - loss: 10.4131 - mae: 2.1367 - val_loss: 16.2353 - val_mae: 2.3145
Epoch 14/500
323/323 [==============================] - 0s 1ms/step - loss: 9.8730 - mae: 2.0853 - val_loss: 15.9195 - val_mae: 2.3135
Epoch 15/500
323/323 [==============================] - 0s 1ms/step - loss: 9.9676 - mae: 2.1116 - val_loss: 14.3031 - val_mae: 2.0870
Epoch 16/500
323/323 [==============================] - 0s 1ms/step - loss: 10.2477 - mae: 2.0648 - val_loss: 14.6655 - val_mae: 2.2939
Epoch 17/500
323/323 [==============================] - 0s 1ms/step - loss: 9.7677 - mae: 2.0351 - val_loss: 13.4949 - val_mae: 2.1981
Epoch 18/500
323/323 [==============================] - 1s 2ms/step - loss: 9.3972 - mae: 1.9759 - val_loss: 16.9912 - val_mae: 2.5318
Epoch 19/500
323/323 [==============================] - 0s 1ms/step - loss: 9.5498 - mae: 2.0177 - val_loss: 13.4918 - val_mae: 2.4475
Epoch 20/500
323/323 [==============================] - 0s 1ms/step - loss: 9.1466 - mae: 2.0152 - val_loss: 14.6800 - val_mae: 2.3117
Epoch 21/500
323/323 [==============================] - 0s 1ms/step - loss: 9.1832 - mae: 1.9576 - val_loss: 12.5864 - val_mae: 2.1020
Epoch 22/500
323/323 [==============================] - 0s 1ms/step - loss: 8.4909 - mae: 1.9983 - val_loss: 14.4070 - val_mae: 2.3417
Epoch 23/500
323/323 [==============================] - 0s 1ms/step - loss: 8.8307 - mae: 2.0059 - val_loss: 12.8591 - val_mae: 2.3332
Epoch 24/500
323/323 [==============================] - 0s 1ms/step - loss: 8.5528 - mae: 1.9633 - val_loss: 11.7008 - val_mae: 2.2749
Epoch 25/500
323/323 [==============================] - 0s 1ms/step - loss: 8.3164 - mae: 1.9505 - val_loss: 13.1953 - val_mae: 2.2372
Epoch 26/500
323/323 [==============================] - 0s 1ms/step - loss: 8.3302 - mae: 1.8971 - val_loss: 12.5720 - val_mae: 2.1830
Epoch 27/500
323/323 [==============================] - 0s 1ms/step - loss: 8.3059 - mae: 1.8726 - val_loss: 12.5079 - val_mae: 2.0922
Epoch 28/500
323/323 [==============================] - 0s 1ms/step - loss: 8.0627 - mae: 1.8912 - val_loss: 13.4896 - val_mae: 2.2201
Epoch 29/500
323/323 [==============================] - 0s 1ms/step - loss: 8.1637 - mae: 1.9172 - val_loss: 14.0658 - val_mae: 2.2782
Epoch 30/500
```

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: 16.1713 - mae: 2.8066
MAE is : 2.806593894958496
```

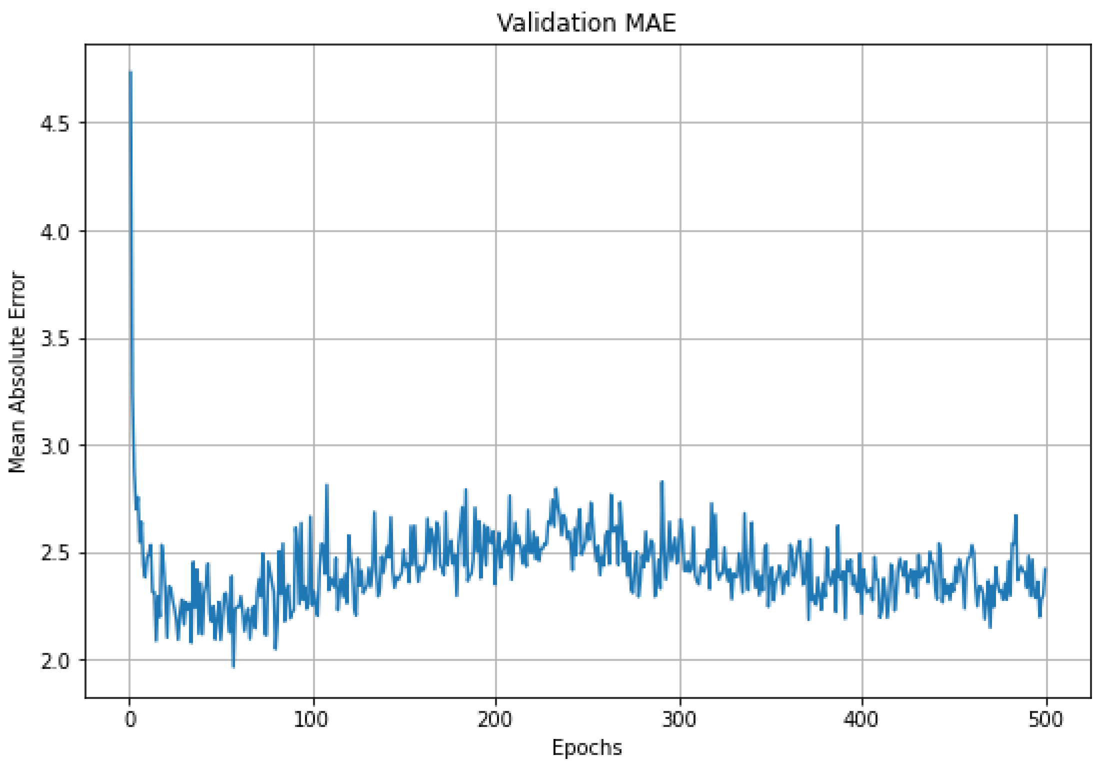
5) Visualization

- 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()
```



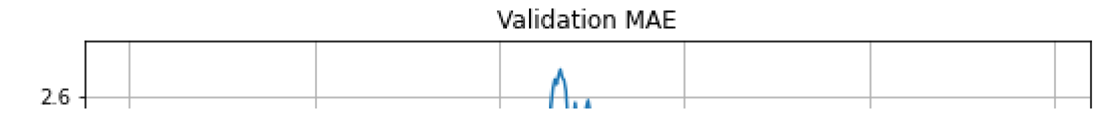
- 5번째 이후 MAE 확인

```
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 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,
                         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: 186.2309 - mae: 9.8153 - val_loss: 48.5474 - val_mae: 4.0331

Epoch 00001: val_mae improved from inf to 4.03314, saving model to best_boston.h5
Epoch 2/500
323/323 [==============================] - 0s 1ms/step - loss: 23.4233 - mae: 3.3636 - val_loss: 29.9810 - val_mae: 2.9452

Epoch 00002: val_mae improved from 4.03314 to 2.94522, saving model to best_boston.h5
Epoch 3/500
323/323 [==============================] - 0s 1ms/step - loss: 18.1602 - mae: 2.9084 - val_loss: 26.2302 - val_mae: 2.7227

Epoch 00003: val_mae improved from 2.94522 to 2.72271, saving model to best_boston.h5
Epoch 4/500
323/323 [==============================] - 0s 1ms/step - loss: 16.1800 - mae: 2.6808 - val_loss: 25.3993 - val_mae: 2.7262

Epoch 00004: val_mae did not improve from 2.72271
Epoch 5/500
323/323 [==============================] - 0s 1ms/step - loss: 14.3402 - mae: 2.6092 - val_loss: 21.7670 - val_mae: 2.5597

Epoch 00005: val_mae improved from 2.72271 to 2.55968, saving model to best_boston.h5
Epoch 6/500
323/323 [==============================] - 0s 1ms/step - loss: 13.0374 - mae: 2.4645 - val_loss: 21.1780 - val_mae: 2.6298

Epoch 00006: val_mae did not improve from 2.55968
Epoch 7/500
323/323 [==============================] - 0s 1ms/step - loss: 12.6326 - mae: 2.4106 - val_loss: 22.2451 - val_mae: 2.6349

Epoch 00007: val_mae did not improve from 2.55968
Epoch 8/500
323/323 [==============================] - 0s 1ms/step - loss: 12.2240 - mae: 2.3938 - val_loss: 18.7530 - val_mae: 2.6626

Epoch 00008: val_mae did not improve from 2.55968
Epoch 9/500
323/323 [==============================] - 0s 1ms/step - loss: 12.0775 - mae: 2.3363 - val_loss: 18.7194 - val_mae: 2.5845

Epoch 00009: val_mae did not improve from 2.55968
Epoch 10/500
323/323 [==============================] - 0s 1ms/step - loss: 11.1523 - mae: 2.2747 - val_loss: 16.9566 - val_mae: 2.7784

Epoch 00010: val_mae did not improve from 2.55968
Epoch 11/500
323/323 [==============================] - 0s 1ms/step - loss: 10.4463 - mae: 2.2197 - val_loss: 19.2765 - val_mae: 2.6450

Epoch 00011: val_mae did not improve from 2.55968
Epoch 12/500
323/323 [==============================] - 0s 1ms/step - loss: 10.7900 - mae: 2.2469 - val_loss: 18.1644 - val_mae: 2.4539

Epoch 00012: val_mae improved from 2.55968 to 2.45392, saving model to best_boston.h5
Epoch 13/500
323/323 [==============================] - 0s 1ms/step - loss: 11.2632 - mae: 2.1598 - val_loss: 16.3796 - val_mae: 2.3680

Epoch 00013: val_mae improved from 2.45392 to 2.36799, saving model to best_boston.h5
Epoch 14/500
323/323 [==============================] - 0s 1ms/step - loss: 10.5911 - mae: 2.1895 - val_loss: 19.4203 - val_mae: 2.4792

Epoch 00014: val_mae did not improve from 2.36799
Epoch 15/500
323/323 [==============================] - 0s 1ms/step - loss: 10.3874 - mae: 2.1136 - val_loss: 17.7036 - val_mae: 2.4127

Epoch 00015: val_mae did not improve from 2.36799
```

5) Best Model

```
!ls -l
```

```
total 76
-rw-r--r-- 1 root root 70280 Sep  1 00:50 best_boston.h5
drwxr-xr-x 1 root root  4096 Aug 25 13:35 sample_data
```

6) Model Evaluate

```
from tensorflow.keras.models import load_model
```

```
best_boston = load_model('best_boston.h5')
```

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

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

```
4/4 [==============================] - 0s 3ms/step - loss: 25.0139 - mae: 2.6835
MAE is : 2.683537483215332
```

#

#

#

The End

#

#

#