Listing 3.24. Loading the Boston housing dataset

```
[ ] from keras.datasets import boston_housing

    (train_data, train_targets), (test_data, test_targets) = boston_housing.load_data()
```

```
train_data.shape#變數有13個
```

```
(404, 13)
```

```
[ ] test_data.shape
```

```
(102, 13)
```

```
[ ] train_data[0]
```

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

```
[ ] train_targets[0]
```

```
15.2
```

Listing 3.25. Normalizing the data

```
[ ] mean = train_data.mean(axis=0)
    train_data -= mean
    std = train_data.std(axis=0)
    train_data /= std

    test_data -= mean#用train data 的mean
    test_data /= std
```

Listing 3.26. Model definition

```
[ ] from keras import models
    from keras import layers
    def build_model():
        model = models.Sequential()
        model.add(layers.Dense(64, activation='relu',input_shape=(train_data.shape[1],)))
        model.add(layers.Dense(64, activation='relu'))
        model.add(layers.Dense(1))#後面無須做其他的處理activation
        model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
        return model
```

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

+ 程式碼 + 文字

```
[ ] for i in range(k):
        print('processing fold #', i)
        val_data = train_data[i * num_val_samples: (i + 1) * num_val_samples]
        val_targets = train_targets[i * num_val_samples: (i + 1) * num_val_samples]
        partial_train_data = np.concatenate(
        [train_data[:i * num_val_samples],
        train_data[(i + 1) * num_val_samples:]],axis=0)
        partial_train_targets = np.concatenate([train_targets[:i * num_val_samples],
                                                train_targets[(i + 1) * num_val_samples:]],
                                               axis=0)

        model = build_model()
        model.fit(partial_train_data, partial_train_targets,
        epochs=num_epochs, batch_size=1, verbose=0)
        val_mse, val_mae = model.evaluate(val_data, val_targets, verbose=0)
        all_scores.append(val_mae)
```

```
processing fold # 0
processing fold # 1
processing fold # 2
processing fold # 3
```

```
[ ] all_scores
```

```
[2.4615018367767334, 2.361207962036133, 2.9153897762298584, 2.443928241729736]
```

```
[ ] np.mean(all_scores)#平均是三個單位
```

```
2.5455069541931152
```

Listing 3.28. Saving the validation logs at each fold

```
[ ] num_epochs = 500
    all_mae_histories = []
    for i in range(k):
        print('processing fold #', i)
        val_data = train_data[i * num_val_samples: (i + 1) * num_val_samples]
        val_targets = train_targets[i * num_val_samples: (i + 1) * num_val_samples]
        partial_train_data = np.concatenate(
            [train_data[:i * num_val_samples],
             train_data[(i + 1) * num_val_samples:]],
            axis=0)
        partial_train_targets = np.concatenate(
            [train_targets[:i * num_val_samples],
             train_targets[(i + 1) * num_val_samples:]],
            axis=0)

        model = build_model()
        history = model.fit(partial_train_data, partial_train_targets,
                            validation_data=(val_data, val_targets),
                            epochs=num_epochs, batch_size=1, verbose=0)
        mae_history = history.history['val_mae']#有誤
        all_mae_histories.append(mae_history)
```

```
processing fold # 0
processing fold # 1
processing fold # 2
processing fold # 3
```

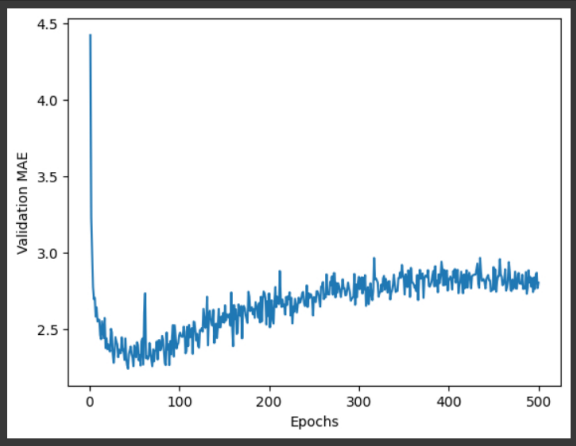
Listing 3.29. Building the history of successive mean K-fold validation scores

```
[ ] average_mae_history = [
        np.mean([x[i] for x in all_mae_histories]) for i in range(num_epochs)]
```

Listing 3.30. Plotting validation scores

```
[ ] import matplotlib.pyplot as plt

    plt.plot(range(1, len(average_mae_history) + 1), average_mae_history)
    plt.xlabel('Epochs')
    plt.ylabel('Validation MAE')
    plt.show()
    #忽略前10筆
```

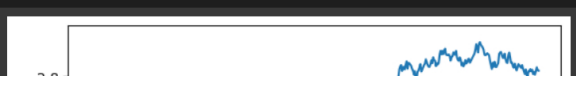


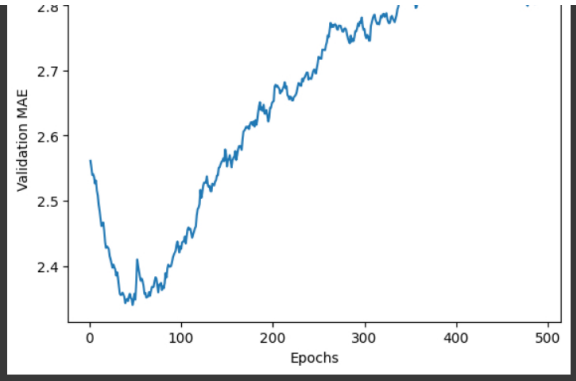
Listing 3.31. Plotting validation scores, excluding the first 10 data points

```
[ ] def smooth_curve(points, factor=0.9):#賦予權重->前一筆0.9 類似平滑化功能
        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

    smooth_mae_history = smooth_curve(average_mae_history[10:])#從第11筆開始

    plt.plot(range(1, len(smooth_mae_history) + 1), smooth_mae_history)
    plt.xlabel('Epochs')
    plt.ylabel('Validation MAE')
    plt.show()
```





Listing 3.32. Training the final model

```
model = build_model()
model.fit(train_data, train_targets,
                    epochs=80, batch_size=16, verbose=0)
test_mse_score, test_mae_score = model.evaluate(test_data, test_targets)
```

```
4/4 [==============================] - 0s 4ms/step - loss: 17.5403 - mae: 2.5419
```

```
test_mae_score#沒有比較大低於3
```

```
2.541926622390747
```