In [1]:
```
from keras.datasets import boston_housing

(train_data, train_targets), (test_data, test_targets) = boston_housing.load_d
ata()
```

In [2]:
```
mean = train_data.mean(axis=0)
train_data -= mean
std = train_data.std(axis=0)
train_data /=std

test_data -= mean
test_data /= std
```

In [3]:
```
from keras import models, layers

def build_model():
    model = models.Sequential()
    model.add(layers.Dense(64, activation= 'relu', input_shape= (train_data.sh
ape[1],)))
    model.add(layers.Dense(64, activation= 'relu'))
    model.add(layers.Dense(1))
    model.compile(optimizer= 'rmsprop', loss= 'mse', metrics= ['mae'])
    return model
```

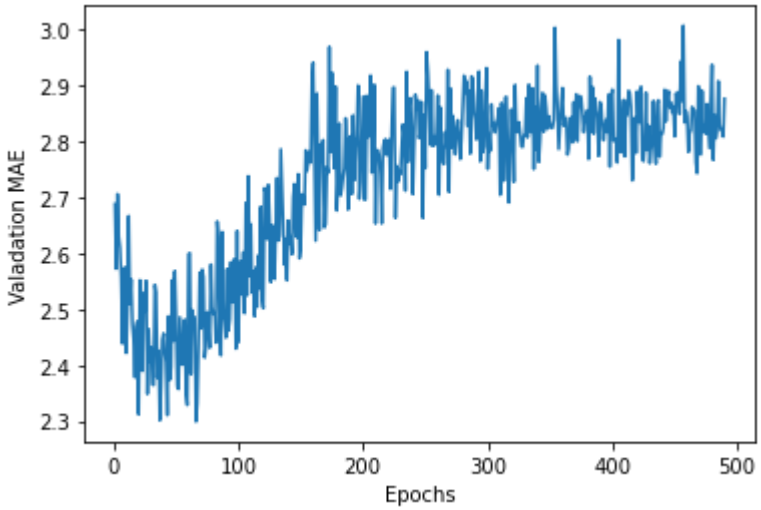
In [4]:
```
import numpy as np
k = 4
num_val_samples = len(train_data) // k
num_epochs = 500
all_mae_histories = []
all_scores = []
for i in range(k):
    print('processing fold #', i)
    val_data = train_data[i * num_val_samples: (i + 1) * num_val_samples]
    val_targets = train_targets[i * num_val_samples: (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)
    history.history.keys()
    val_mse, val_mae = model.evaluate(val_data, val_targets, verbose= 0)
    all_scores.append(val_mae)
    mae_history = history.history['val_mae']
    all_mae_histories.append(mae_history)
```

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

In [5]:
```
average_mae_history = [np.mean([x[i] for x in all_mae_histories]) for i in ran
ge(num_epochs)]
```

In [9]:
```
import matplotlib.pyplot as plt
plt.plot(range(1, len(average_mae_history[10:])+1), average_mae_history[10:])
plt.xlabel('Epochs')
plt.ylabel('Valadation MAE');
```



In [17]:
```
model=build_model()
model.fit(train_data, train_targets, epochs= 80, batch_size= 15, verbose=0)
test_mse_score, test_mae_score = model.evaluate(test_data, test_targets)
```

```
4/4 [==============================] - 0s 1ms/step - loss: 17.4750 - mae: 2.6
343
```

In [18]:
```
test_mae_score
```

Out[18]:
```
2.6343274116516113
```

In [ ]:
```

```