```
sgd_reg = SGDRegressor(n_iter=1, warm_start=True, penalty=None,
                       learning_rate="constant", eta0=0.0005)

minimum_val_error = float("inf")
best_epoch = None
best_model = None
for epoch in range(1000):
    sgd_reg.fit(X_train_poly_scaled, y_train)  # continues where it left off
    y_val_predict = sgd_reg.predict(X_val_poly_scaled)
    val_error = mean_squared_error(y_val_predict, y_val)
    if val_error < minimum_val_error:
        minimum_val_error = val_error
        best_epoch = epoch
        best_model = clone(sgd_reg)
```

Note that with `warm_start=True`, when the `fit()` method is called, it just continues training where it left off instead of restarting from scratch.

Logistic Regression

As we discussed in Chapter 1, some regression algorithms can be used for classification as well (and vice versa). *Logistic Regression* (also called *Logit Regression*) is commonly used to estimate the probability that an instance belongs to a particular class (e.g., what is the probability that this email is spam?). If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class (called the positive class, labeled "1"), or else it predicts that it does not (i.e., it belongs to the negative class, labeled "0"). This makes it a binary classifier.

Estimating Probabilities

So how does it work? Just like a Linear Regression model, a Logistic Regression model computes a weighted sum of the input features (plus a bias term), but instead of outputting the result directly like the Linear Regression model does, it outputs the *logistic* of this result (see Equation 4-13).

Equation 4-13. Logistic Regression model estimated probability (vectorized form)

$$\hat{p} = h_{\theta}(\mathbf{x}) = \sigma\left(\theta^T \cdot \mathbf{x}\right)$$

The logistic—also called the *logit*, noted $\sigma(\cdot)$—is a *sigmoid function* (i.e., *S*-shaped) that outputs a number between 0 and 1. It is defined as shown in Equation 4-14 and Figure 4-21.