suited for imbalanced classification. The coefficients of the logistic regression algorithm are fit using an optimization algorithm that minimizes the negative log likelihood (loss) for the model on the training dataset.

$$\min \sum_{i=1}^{n} -(\log(yhat_i) \times y_i + \log(1 - yhat_i) \times (1 - y_i)) \tag{16.1}$$

This involves the repeated use of the model to make predictions followed by an adaptation of the coefficients in a direction that reduces the loss of the model. The calculation of the loss for a given set of coefficients can be modified to take the class balance into account. By default, the errors for each class may be considered to have the same weighting, say 1.0. These weightings can be adjusted based on the importance of each class.

$$\min \sum_{i=1}^{n} -(w0 \times \log(yhat_i) \times y_i + w1 \times \log(1 - yhat_i) \times (1 - y_i)) \tag{16.2}$$

The weighting is applied to the loss so that smaller weight values result in a smaller error value, and in turn, less update to the model coefficients. A larger weight value results in a larger error calculation, and in turn, more update to the model coefficients.

- **Small Weight**: Less importance, less update to the model coefficients.
- **Large Weight**: More importance, more update to the model coefficients.

As such, the modified version of logistic regression is referred to as Weighted Logistic Regression, Class-Weighted Logistic Regression or Cost-Sensitive Logistic Regression. The weightings are sometimes referred to as importance weightings. Although straightforward to implement, the challenge of weighted logistic regression is the choice of the weighting to use for each class.

16.4 Weighted Logistic Regression with Scikit-Learn

The scikit-learn Python machine learning library provides an implementation of logistic regression that supports class weighting. The `LogisticRegression` class provides the `class_weight` argument that can be specified as a model hyperparameter. The `class_weight` is a dictionary that defines each class label (e.g. 0 and 1) and the weighting to apply in the calculation of the negative log likelihood when fitting the model. For example, a 1 to 1 weighting for each class 0 and 1 can be defined as follows:

```
...
# define model
weights = {0:1.0, 1:1.0}
model = LogisticRegression(solver='lbfgs', class_weight=weights)
```

Listing 16.9: Example of defining the default class weighting for logistic regression.

The class weighing can be defined multiple ways; for example:

- **Domain expertise**, determined by talking to subject matter experts.