The result shows that the model does a good job in capturing the variability in the target feature. ***Of course, we want to have such good performances on the test dataset***.

Resampling Techniques

This section describes another vital concept for evaluating the performance of supervised learning methods. Resampling methods are a set of techniques that involve selecting a subset of the available dataset, training on that data subset, and then using the remainder of the data to evaluate the trained model.

This process involves creating subsets of the available data into a training set and a validation set. The training set is used to train the model, while the validation set will evaluate the performance of the learned model on unseen data. Typically, this process will be carried out repeatedly to get an approximate measure of the training and test errors.

We will examine three techniques for data resampling and also give some examples of when to use a particular technique. The techniques we'll examine are

- The validation set technique (or train-test split)
- The leave-one-out cross-validation (LOOCV) technique
- The *k*-fold cross-validation technique

The Validation Set

The validation set is the simplest approach for data resampling, and it involves randomly dividing the dataset into two parts; these are the training set and the validation set. The division can be into two equal sets if you have a big enough dataset, or it could be a 60/40 or 70/30 split.

After splitting, the model is trained on the training set, and its performance is evaluated on the validation set. This process is summarized in the list as follows:

1. Randomly split the dataset into
   - Training set
   - Validation set
2. Train the model on the training set.

3. Evaluate the model performance on the validation set using the appropriate error metric for a regression or classification problem.
4. No. 1 to No. 3 can be repeated.
5. Report the error metric to get the ensemble training and validation set error distribution.

A sample validation set is shown in Figure 14-13.

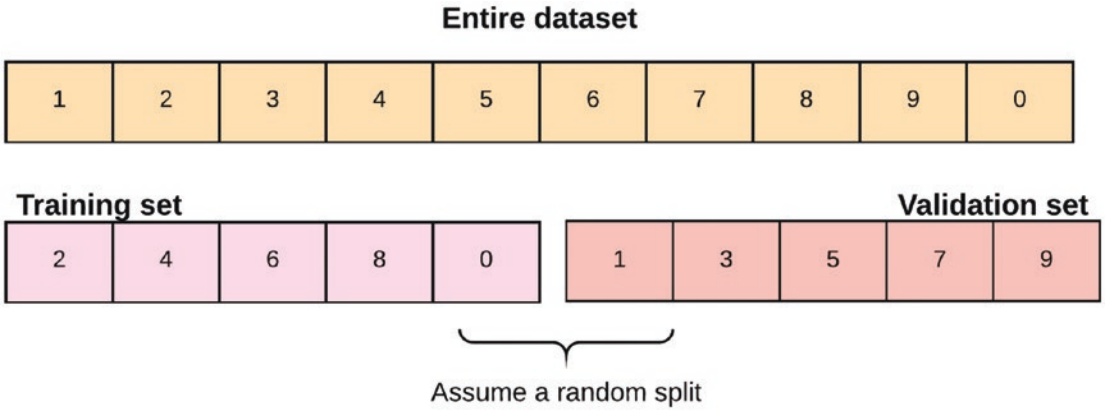

Figure 14-13. *Validation set*

Leave-One-Out Cross-Validation (LOOCV)

The leave-one-out cross-validation approach (commonly referred to as LOOCV) involves dividing the dataset into a training set and a test set. But unlike the validation approach, LOOCV assigns just one example to the test set, and trains the model on the remainder of the dataset. This process is repeated until all the examples in the dataset have been used for evaluating the model.

Assuming we have ten examples in a dataset (let n be used to denote the size of the dataset) to build a learning model. We will train the model using $n - 1$ examples and evaluate the model on just the single remaining example, hence the name leave-one-out. This process is repeated n times for all the examples in the dataset. At the end of the n iterations, we will report the average error estimate.

A sample LOOCV is shown in Figure 14-14.

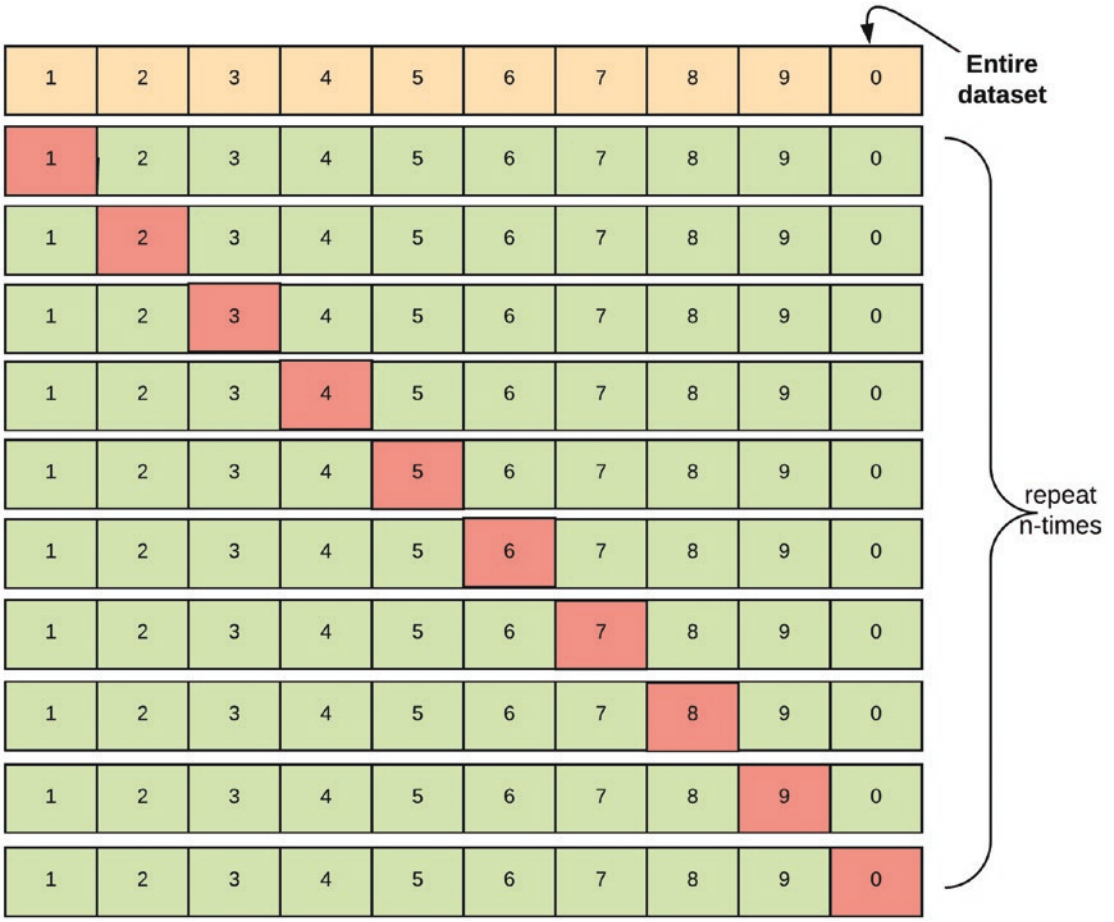

Figure 14-14. *LOOCV*

The main drawback to using LOOCV is that it is *computationally expensive*. The word *computationally expensive* is when a process takes a lot of computing time and memory to complete its execution.

k-Fold Cross-Validation

k-Fold cross-validation mitigates the computational cost of LOOCV while maintaining its benefits in terms of giving an unbiased estimate of the performance of the learned model when evaluated on validation data.

Let's use the following recipe to explain the idea behind *k*-fold CV:

- Divide the dataset into *k* parts or folds. Assume we have a dataset with 20 records; we'll divide the dataset into four parts. See Figure 14-15.

Figure 14-15. *Divide your dataset into k parts or folds*

- Hold out one of the four splits as a test set, and train the model on the remaining splits. Repeat this until all the splits have been held out for testing. See Figure 14-16.

Figure 14-16. *Train the model using k − 1 example sets or splits*

- Report the ensemble error metric.

Note From this explanation, we can see that LOOCV is a special case where $k = n$.

Improving Model Performance

To improve the performance of the model, a few of the techniques to consider are

1. Systematic feature engineering
2. Using ensemble learning methods (we'll discuss more on this in a later chapter)
3. Hyper-parameter tuning of the algorithm

Feature Engineering

In model building, a significant portion of time is spent on feature engineering. Feature engineering is the practice of systematically going through each feature in the dataset and investigating its relevance to the targets.

Through feature engineering, we can cleverly introduce new features by combining one or more existing features, and this can impact the prediction accuracy of the model. Feature engineering can sometimes be the difference between a decent learning model and a competition-winning model.

Ensemble Methods

Ensemble methods combine the output of weaker models to produce a better performing model. Two major classes of ensemble learning algorithms are

- Boosting
- Bagging

In practice, ensemble methods such as Random forests are known to do very well in various machine learning problems and are the algorithms of choice for machine learning competitions.

Hyper-parameter Tuning

When modeling with a learning algorithm, we can adjust certain configurations of the algorithm. These configurations are called hyper-parameters. Hyper-parameters are tuned to get the best settings of the algorithms that will optimize the performance of the model. One strategy is to use a grid search to adjust the hyper-parameters when fine-tuning the model.