Classifier performance -

It is important to know how well our classifier performs. The performance of a classifier is a compound characteristic, whose most important component is the classification accuracy. If we were able to try the classifier on all possible input objects, we would know exactly how accurate it is. Unfortunately, this is hardly a possible scenario, so an estimate of the accuracy is used instead.

Assume that a labeled data set Z of size $N \times n$ is available for testing the accuracy of our classifier D. The most natural way to calculate an estimate of the error, is to run D on all the objects in Z and find the proportion of misclassified objects. To look at the error from a probabilistic point of view, we can adopt the following model. The classifier commits an error with probability P_D on any object $x \in \mathbb{R}^n$, which is a useful assumption. Then the estimate of P_D is

$$\hat{P}_D = \frac{N_{error}}{N}$$

For the given data set Z of size $N \times n$, containing n-dimensional feature vectors describing N objects. We would like to use as much as possible of the data to build the classifier (training), and also as much as possible unseen data to test it's performance more thoroughly (testing). However, if we use all data for training and the same data for testing, we might overtrain the classifier so that it perfectly learns the available data and fails on unseen data. That is why it is important to have a separate data set on which to examine the final product. The main alternatives for making the best use of Z can be summarized as follows-

1) Resubstitution (R- method) - Design classifier D on Z and test it on Z. $\hat{P}_D$ is optimistically biased.

2) Holdout (H-method) - Traditionally split Z into halves, use one half for training, and the other half for calculating $\hat{P}_D$. Splits in other proportions are also used in practice. We can swap the two subsets, get another estimate $\hat{P}_D$ and average the two. A version of this method is the data shuffle, where we do L random splits of Z into training and testing parts and average all L estimates of P_D, as calculated on the respective testing parts.

3) Cross-validation (Rotation method or π-method) - We choose an integer K (most preferably a factor of N) and randomly divide Z into K subsets of size N/K. Then we use one subset to test the performance of D trained on the union of the remaining K-1 subsets. This procedure is repeated K times, choosing a different part for testing each time. To get the final value of $\hat{P}_D$, we average the K estimates. When K=N, the method is called the 'Leave-one-out' method.

4) Bootstrap (B-method) - This is done by randomly generating L sets of cardinality N from the original set with replacement. Then we assess and average the error rate of the classifiers built on these sets.