Andhra Pradesh State Skill Development Corporation

Skill AP
APSSDC

eSkill AP
Learn Anytime Anywhere
Andhra Pradesh State Skill Development Corporation

Machine Learning

Modelling and Evaluation

CHAPTER 4
Modelling and Evaluation

Selecting a Model :

Now that you are familiar with the basic learning process and have understood model abstraction and generalization in that context, let's try to formalize it in context of a motivating example. Continuing the thread of the potential attack during the election campaign, the New City Police department has succeeded in foiling the bid to attack the electoral candidate. However, this was a wake-up call for them and they want to take a proactive action to eliminate all criminal activities in the region. They want to find the pattern of criminal activities in the recent past, i.e. they want to see whether the number of criminal incidents per month has any relation with the average income of the local population, weapon sales, the inflow of immigrants, and other such factors. Therefore, an association between potential causes of disturbance and criminal incidents has to be determined. In other words, the goal or target is to develop a model to infer how the criminal incidents change based on the potential influencing factors mentioned above.

In the machine learning paradigm, the potential causes of disturbance, e.g. average income of the local population, weapon sales, the inflow of immigrants, etc. are input variables. They are also called predictors, attributes, features, independent variables, or simply variables. The number of criminal incidents is an output variable (also called response or dependent variable). Input variables can be denoted by X, while individual input variables are represented as $X1$, $X2$, $X3$, …, Xn and output variables by symbol Y. The relationship between X and Y is represented in the general form: $Y = f(X) + e$, where 'f' is the target function and 'e' is a random error term.

But the problem that we just talked about is one specific type of problem in machine learning. We have seen in Chapter 1 that there are three broad categories of machine learning approaches used for resolving different types of problems. Quickly recapitulating, they are

1. Supervised Learning
    1. Classification
    2. Regression
2. Unsupervised Learning
    1. Clustering
    2. Association analysis
3. Reinforcement Learning

For each of the cases, the model that has to be created/trained is different. Multiple factors play a role when we try to select the model for solving a machine learning problem. The most important factors are

(i) The kind of problem we want to solve using machine learning
(ii) The nature of the underlying data.

The problem may be related to the prediction of a class value like whether a tumour is malignant or benign, whether the next day will be snowy or rainy, etc. It may be related to prediction – but of some numerical value like what the price of a house should be in the next quarter, what is the expected growth of a certain IT stock in the next 7 days, etc. Certain problems are related to grouping of data like finding customer segments that are using a certain product, movie genres which have got more box office success in the last one year, etc. So, it is very difficult to give a generic guidance related to which machine learning has to be selected. In other words, there is no one model that works best for every machine learning problem. This is what '**No Free Lunch**' theorem also states.

Any learning model tries to simulate some real-world aspect. However, it is simplified to a large extent removing all intricate details. These simplifications are based on certain assumptions –

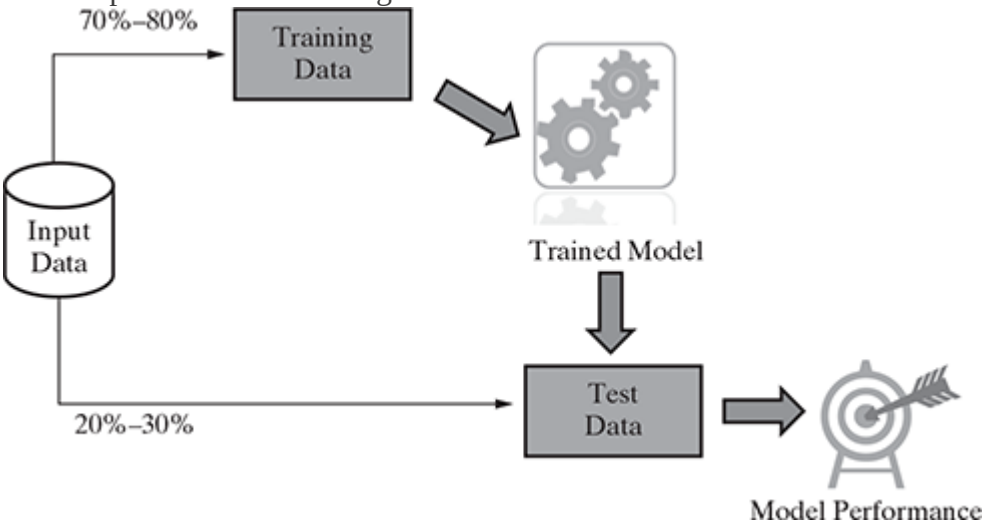
which are quite dependent on situations. Based on the exact situation, i.e. the problem in hand and the data characteristics, assumptions may or may not hold. So the same model may yield remarkable results in a certain situation while it may completely fail in a different situation. That's why, while doing the data exploration, which we covered in the previous chapter, we need to understand the data characteristics, combine this understanding with the problem we are trying to solve and then decide which model to be selected for solving the problem.

Let's try to understand the philosophy of model selection in a structured way. Machine learning algorithms are broadly of two types: models for supervised learning, which primarily focus on solving predictive problems and models for unsupervised learning, which solve descriptive problems

Training a Model

Holdout method:

In case of supervised learning, a model is trained using the labelled input data. However, how can we understand the performance of the model? The test data may not be available immediately. Also, the label value of the test data is not known. That is the reason why a part of the input data is held back (that is how the name holdout originates) for evaluation of the model. This subset of the input data is used as the test data for evaluating the performance of a trained model. In general 70%–80% of the input data (which is obviously labelled) is used for model training. The remaining 20%–30% is used as test data for validation of the performance of the model. However, a different proportion of dividing the input data into training and test data is also acceptable. To make sure that the data in both the buckets are similar in nature, the division is done randomly. Random numbers are used to assign data items to the partitions. This method of partitioning the input data into two parts – training and test data which is by holding back a part of the input data for validating the trained model is known as holdout method.



Holdout method

Once the model is trained using the training data, the labels of the test data are predicted using the model's target function. Then the predicted value is compared with the actual value of the label. This is possible because the test data is a part of the input data with known labels. The performance of the model is in general measured by the accuracy of prediction of the label value.

In certain cases, the input data is partitioned into three portions – a training and a test data, and a third validation data. The validation data is used in place of test data, for measuring the model

performance. It is used in iterations and to refine the model in each iteration. The test data is used only for once, after the model is refined and finalized, to measure and report the final performance of the model as a reference for future learning efforts.

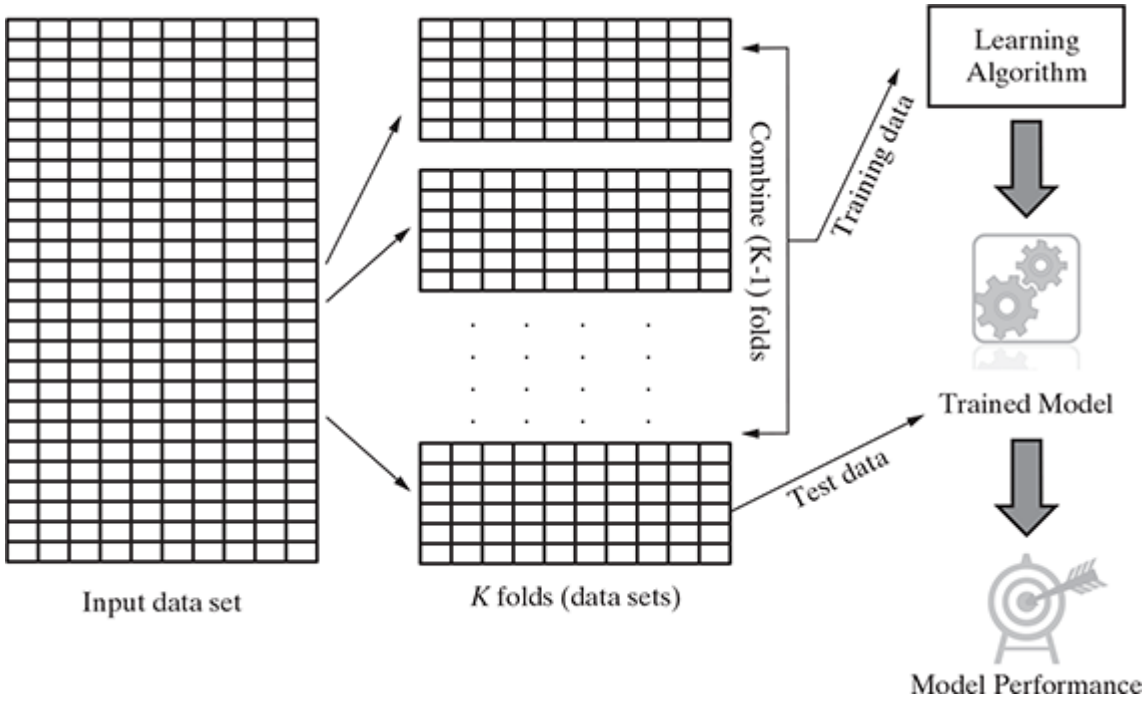
An obvious problem in this method is that the division of data of different classes into the training and test data may not be proportionate. This situation is worse if the overall percentage of data related to certain classes is much less compared to other classes. This may happen despite the fact that random sampling is employed for test data selection. This problem can be addressed to some extent by applying stratified random sampling in place of sampling. In case of stratified random sampling, the whole data is broken into several homogenous groups or strata and a random sample is selected from each such stratum. This ensures that the generated random partitions have equal proportions of each class.

K-fold Cross-validation method:

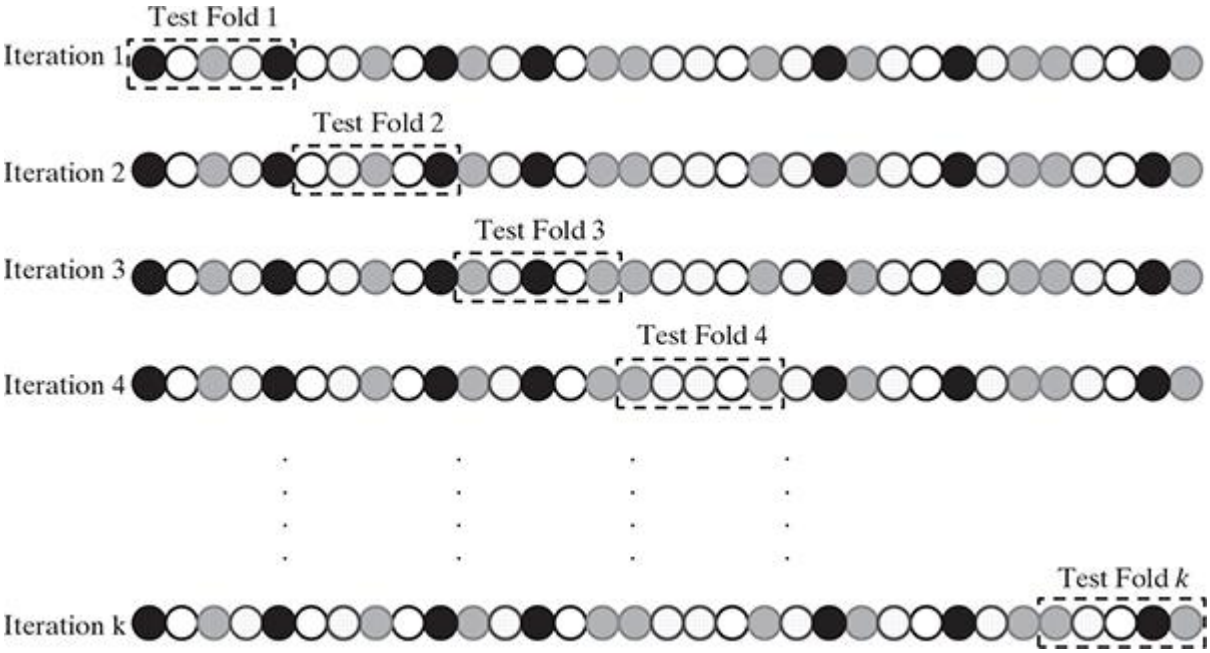
Holdout method employing stratified random sampling approach still heads into issues in certain specific situations. Especially, the smaller datasets may have the challenge to divide the data of some of the classes proportionally amongst training and test data sets. A special variant of holdout method, called repeated holdout, is sometimes employed to ensure the randomness of the composed data sets. In repeated holdout, several random holdouts are used to measure the model performance. In the end, the average of all performances is taken. As multiple holdouts have been drawn, the training and test data (and also validation data, in case it is drawn) are more likely to contain representative data from all classes and resemble the original input data closely. This process of repeated holdout is the basis of k-fold cross-validation technique. In k-fold cross-validation, the data set is divided into k-completely distinct or non-overlapping random partitions called folds. depicts an overall approach for k-fold cross-validation.

The value of ‘k’ in k-fold cross-validation can be set to any number. However, there are two approaches which are extremely popular:

10-fold cross-validation (10-fold CV) and Leave-one-out cross-validation (LOOCV),10-fold cross-validation is by far the most popular approach. In this approach, for each of the 10-folds, each comprising approximately 10% of the data, one of the folds is used as the test data for validating model performance trained based on the remaining 9 folds (or 90% of the data). This is repeated 10 times, once for each of the 10 folds being used as the test data and the remaining folds as the training data. The average performance across all folds is being reported. depicts the detailed approach of selecting the ‘k’ folds in k-fold cross-validation. As can be observed in the figure, each of the circles resembles a record in the input data set whereas the different colors indicate the different classes that the records belong to. The entire data set is broken into ‘k’ folds – out of which one fold is selected in each iteration as the test data set. The fold selected as a test data set in each of the ‘k’ iterations is different. Also, note that though the circles resemble the records in the input data set, the contiguous circles represented as folds do not mean that they are subsequent records in the data set. This is more a virtual representation and not a physical representation. As already mentioned, the records in a fold are drawn by using random sampling technique.



Overall approach for K-fold cross-validation



Detailed approach for fold selection

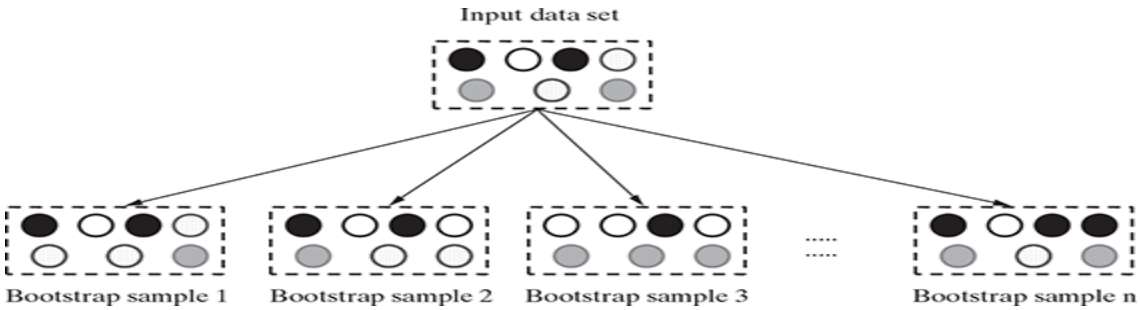
Leave-one-out cross-validation (LOOCV) is an extreme case of k-fold cross-validation using one record or data instance at a time as a test data. This is done to maximize the count of data used to train the model. It is obvious that the number of iterations for which it has to be run is equal to the total number of data in the input data set. Hence, obviously, it is computationally very expensive and not used much in practice.

Bootstrap sampling

Bootstrap sampling or simply bootstrapping is a popular way to identify training and test data sets from the input data set. It uses the technique of Simple Random Sampling with Replacement

(SRSWR), which is a well-known technique in sampling theory for drawing random samples. We have seen earlier that k-fold cross-validation divides the data into separate partitions – say 10 partitions in case of 10-fold cross-validation. Then it uses data instances from partition as test data and the remaining partitions as training data. Unlike this approach adopted in case of k-fold cross- validation, bootstrapping randomly picks data instances from the input data set, with the possibility of the same data instance to be picked multiple times. This essentially means that from the input data set having 'n' data instances, bootstrapping can create one or more training data sets having 'n' data instances, some of the data instances being repeated multiple times. briefly presents the approach followed in bootstrap sampling.

This technique is particularly useful in case of input data sets of small size, i.e. having very less number of data instances.



Bootstrap sampling

CROSS-VALIDATION	BOOTSTRAPPING
It is a special variant of holdout method, called repeated holdout. Hence uses stratified random sampling approach (without replacement). Data set is divided into 'k' random partitions, with each partition containing approximately $\frac{n}{k}$ number of unique data elements, where 'n' is the total number of data elements and 'k' is the total number of folds.	It uses the technique of Simple Random Sampling with Replacement (SRSWR). So the same data instance may be picked up multiple times in a sample.
The number of possible training/test data samples that can be drawn using this technique is finite.	In this technique, since elements can be repeated in the sample, possible number of training/test data samples is unlimited.

Lazy vs. Eager learner

Eager learning follows the general principles of machine learning – it tries to construct a generalized, input-independent target function during the model training phase. It follows the typical steps of machine learning, i.e. abstraction and generalization and comes up with a trained model at the end of the learning phase. Hence, when the test data comes in for classification, the eager learner is ready with the model and doesn't need to refer back to the training data. Eager learners take more time in the learning phase than the lazy learners. Some of the algorithms which adopt an eager learning approach include Decision Tree, Support Vector Machine, Neural Network, etc.

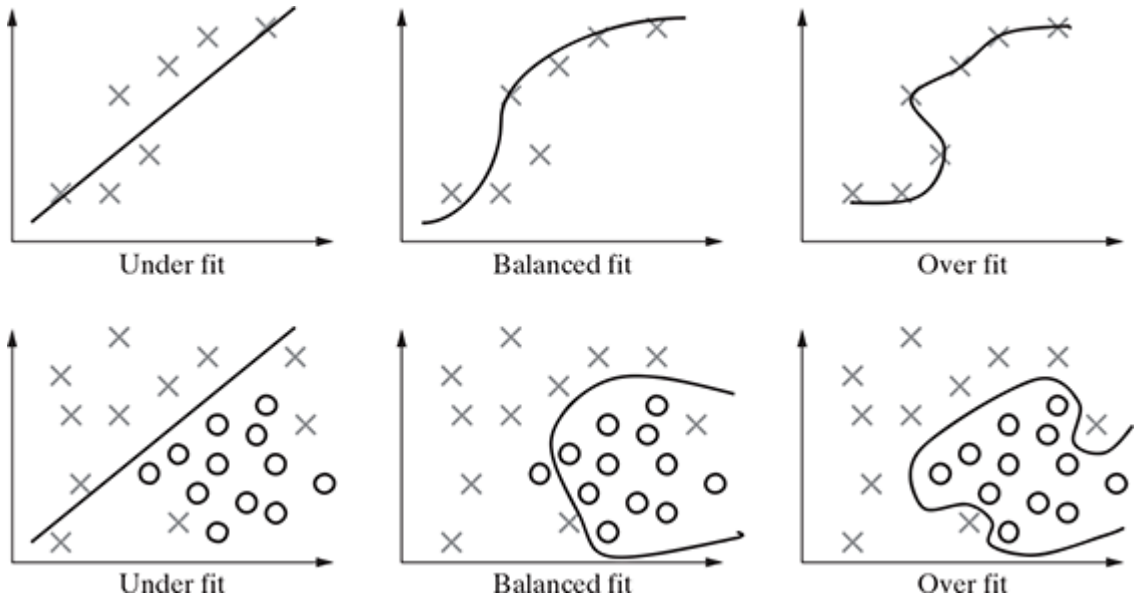
Lazy learning, on the other hand, completely skips the abstraction and generalization processes, as explained in context of a typical machine learning process. In that respect, strictly speaking, a lazy learner doesn't 'learn' anything. It uses the training data in exact, and uses the knowledge to classify the unlabelled test data. Since lazy learning uses training data as-is, it is also known as rote learning (i.e. memorization technique based on repetition). Due to its heavy dependency on the given training data instance, it is also known as instance learning. They are also called non-parametric learning. Lazy learners take very little time in training because not much of training actually happens. However, it takes quite some time in classification as for each tuple of test data, a comparison-based assignment of label happens. One of the most popular algorithms for lazy learning is k-nearest neighbour.

Model Representation and Interpretability

We have already seen that the goal of supervised machine learning is to learn or derive a target function which can best determine the target variable from the set of input variables. A key consideration in learning the target function from the training data is the extent of generalization. This is because the input data is just a limited, specific view and the new, unknown data in the test data set may be differing quite a bit from the training data. Fitness of a target function approximated by a learning algorithm determines how correctly it is able to classify a set of data it has never seen.

Under fitting:

If the target function is kept too simple, it may not be able to capture the essential nuances and represent the underlying data well. A typical case of under fitting may occur when trying to represent a non-linear data with a linear model as demonstrated by both cases of under fitting. Many times under fitting happens due to unavailability of sufficient training data. Under fitting results in both poor performance with training data as well as poor generalization to test data. Under fitting can be avoided by
using more training data reducing features by effective feature selection



Under fitting and Over fitting of models

Over fitting

Over fitting refers to a situation where the model has been designed in such a way that it emulates the training data too closely. In such a case, any specific deviation in the training data, like noise or outliers, gets embedded in the model. It adversely impacts the performance of the model on the test data. Over fitting, in many cases, occur as a result of trying to fit an excessively complex model to closely match the training data. This is represented with a sample data set . The target function, in these cases, tries to make sure all training data points are correctly partitioned by the decision boundary. However, more often than not, this exact nature is not replicated in the unknown test data set. Hence, the target function results in wrong classification in the test data set. Overfitting results in good performance with training data set, but poor generalization and hence poor performance with test data set. Overfitting can be avoided by

1. using re-sampling techniques like *k*-fold cross validation
2. hold back of a validation data set
3. remove the nodes which have little or no predictive power for the given machine learning problem.

Both underfitting and overfitting result in poor classification quality which is reflected by low classification accuracy.