Chapter 3

Modelling and Evaluation

OBJECTIVE OF THE CHAPTER:

The previous chapter gives a comprehensive understanding of the basic data types in the context of machine learning. It also enables a beginner in the field of machine learning to acquire an understanding about the nature and quality of the data by effective exploration of the data set. In this chapter, the objective is to introduce the basic concepts of learning. In this regard, the information shared concerns the aspects of model selection and application. It also imparts knowledge regarding how to judge the effectiveness of the model in doing a specific learning task, supervised or unsupervised, and how to boost the model performance using different tuning parameters.

3.1 INTRODUCTION

The learning process of machines may seem quite magical to somebody who is new to machine learning. The thought that a machine is able to think and take intelligent action may be mesmerizing – much like a science fiction or a fantasy story. However, delving a bit deeper helps them realize that it is not as magical as it may seem to be. In fact, it tries to emulate human learning by applying mathematical and statistical formulations. In that sense, both human and machine learning strives to build formulations or mapping based on a limited number of observations. As introduced

in Chapter 1, the basic learning process, irrespective of the fact that the learner is a human or a machine, can be divided into three parts:

1. Data Input
2. Abstraction
3. Generalization

Though in Chapter 1 we have understood these aspects in details, let's quickly refresh our memory with an example. It's a fictitious situation. The detective department of New City Police has got a tip that in a campaign gathering for the upcoming election, a criminal is going to launch an attack on the main candidate. However, it is not known who the person is and quite obviously the person might use some disguise. The only thing that is for sure is the person is a history-sheeter or a criminal having a long record of serious crime. From the criminal database, a list of such criminals along with their photographs has been collected. Also, the photos taken by security cameras positioned at different places near the gathering are available with the detective department. They have to match the photos from the criminal database with the faces in the gathering to spot the potential attacker. So the main problem here is to spot the face of the criminal based on the match with the photos in the criminal database.

This can be done using human learning where a person from the detective department can scan through each shortlisted photo and try to match that photo with the faces in the gathering. A person having a strong memory can take a glance at the photos of all criminals in one shot and then try to find a face in the gathering which closely resembles one of the criminal photos that she has viewed. Easy, isn't it? But that is not possible in reality. The number of criminals in the database and hence the count of photos runs in hundreds, if not thousands. So taking a look at all the photos and memorizing them is not possible. Also, an exact match is out of the question as the criminal, in most probability, will come in disguise. The strategy to be taken here is to match the photos in smaller counts and also based on certain salient physical features like the shape of the jaw, the slope of the forehead, the size of the eyes, the structure of the ear, etc. So, the photos from the criminal database form the input data. Based on it, key features can be abstracted. Since human matching for each and every photo may soon lead to a visual as well as mental fatigue, a generalization of abstracted feature-based data is a good way to detect potential criminal faces in the gathering. For example, from the abstracted feature-based data, say it is observed that most of the criminals have a shorter distance between the inner corners of the eyes, a smaller angle between the nose and the corners of the mouth, a higher curvature to the upper lip, etc. Hence, a face in the gathering may be classified as 'potentially criminal' based on whether they match with these generalized observations. Thus, using the input data, feature-based abstraction could be built and by applying generalization of the abstracted data, human learning could classify the faces as potentially criminal ultimately leading to spotting of the criminal.

The same thing can be done using machine learning too. Unlike human detection, a machine has no subjective baggage, no emotion, no bias due to past experience, and above all no mental fatigue. The machine can also use the same input data, i.e. criminal database photos, apply computational techniques to abstract feature-based concept map from the input data and generalize the same in the form of a classification algorithm to decide whether a face in the gathering is potentially criminal or not.

When we talk about the learning process, abstraction is a significant step as it represents raw input data in a summarized and structured format, such that a meaningful insight is obtained from the data. This structured representation of raw input data to the meaningful pattern is called a **model**. The model might have different forms. It might be a mathematical equation, it might be a graph or tree structure, it might be a computational block, etc. The decision regarding which model is to be selected for a specific data set is taken by the learning task, based on the problem to be solved and the type of data. For example, when the problem is related to prediction and the target field is numeric and continuous, the regression model is assigned. The process of assigning a model, and fitting a specific model to a data set is called model **training**. Once the model is trained, the raw input data is summarized into an abstracted form.

However, with abstraction, the learner is able to only summarize the knowledge. This knowledge might be still very broad-based – consisting of a huge number of feature-based data and inter-relations. To generate actionable insight from such broad-based knowledge is very difficult. This is where generalization comes into play. Generalization searches through the huge set of abstracted knowledge to come up with a small and manageable set of key findings. It is not possible to do an exhaustive search by reviewing each of the abstracted findings one-by-one. A heuristic search is employed, an approach which is also used for human learning (often termed as 'gut-feel'). It is quite obvious that the heuristics sometimes result in erroneous result. If the outcome is systematically incorrect, the learning is said to have a **bias**.

Points to Ponder:

- A machine learning algorithm creates its cognitive capability by building a mathematical formulation or function, known as target function, based on the features in the input data set.
- Just like a child learning things for the first time needs her parents guidance to decide whether she is right or wrong, in machine learning someone has to provide some non-learnable parameters, also called hyper-parameters. Without these human inputs, machine learning algorithms cannot be successful.

■

3.2 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, 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 an 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 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 $X_1, X_2, X_3, \ldots, X_n$ and output variable 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.

Note:

Just like a target function with respect to a machine learning model, some other functions which are frequently tracked are

- A **cost function** (also called **error function**) helps to measure the extent to which the model is going wrong in estimating the relationship between X and Y. In that sense, cost function can tell how bad the model is performing. For example, R-squared (to be discussed later in this chapter) is a cost function of regression model.
- **Loss function** is almost synonymous to cost function – only difference being loss function is usually a function defined on a data point, while cost function is for the entire training data set.
- Machine learning is an optimization problem. We try to define a model and tune the parameters to find the most suitable solution to a problem. However, we need to have a way to evaluate the quality or optimality of a solution. This is done using **objective function**. Objective means goal.
- Objective function takes in data and model (along with parameters) as input and returns a value. Target is to find values of model parameter to maximize or minimize the return value. When the objective is to minimize the value, it becomes synonymous to cost function. Examples: maximize the reward function in reinforcement learning, maximize the posterior probability in Naive Bayes, minimize squared error in regression.

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
   (a) Classification
   (b) Regression

2. Unsupervised
   (a) Clustering
   (b) Association analysis
3. Reinforcement

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 and (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.

3.2.1 Predictive models

Models for supervised learning or predictive models, as is understandable from the name itself, try to predict certain value using the values in an input data set. The learning model attempts to establish a relation between the target feature, i.e. the feature being predicted, and the predictor features. The predictive models have a clear focus on what they want to learn and how they want to learn.

Predictive models, in turn, may need to predict the value of a category or class to which a data instance belongs to. Below are some examples:

(i) Predicting win/loss in a cricket match
(ii) Predicting whether a transaction is fraud
(iii) Predicting whether a customer may move to another product

The models which are used for prediction of target features of categorical value are known as classification models. The target feature is known as a class and the categories to which classes are divided into are called levels. Some of the popular classification models include *k*-Nearest Neighbor (kNN), Naïve Bayes, and Decision Tree.

Predictive models may also be used to predict numerical values of the target feature based on the predictor features. Below are some examples:

(i) Prediction of revenue growth in the succeeding year
(ii) Prediction of rainfall amount in the coming monsoon
(iii) Prediction of potential flu patients and demand for flu shots next winter

The models which are used for prediction of the numerical value of the target feature of a data instance are known as regression models. Linear Regression and Logistic Regression models are popular regression models.

Points to Ponder:

- Categorical values can be converted to numerical values and vice versa. For example, for stock price growth prediction, any growth percentage lying between certain ranges may be represented by a categorical value, e.g. 0%–5% as 'low', 5%–10% as 'moderate', 10%–20% as 'high' and > 20% as 'booming'. In a similar way, a categorical value can be converted to numerical value, e.g. in the tumor malignancy detection problem, replace 'benign' as 0 and 'malignant' as 1. This way, the models can be used interchangeably, though it may not work always.
- There are multiple factors to be considered while selecting a model. For example, while selecting the model for prediction, the training data size is an important factor to be considered. If the training data set is small, low variance models like Naïve Bayes are supposed to perform better because model overfitting needs to be avoided in this situation. Similarly, when the training data is large, low bias models like logistic regression should be preferred because they can represent complex relationships in a more effective way.

Few models like Support Vector Machines and Neural Network can be used for both classifications as well as for regression.

■

3.2.2 Descriptive models

Models for unsupervised learning or descriptive models are used to describe a data set or gain insight from a data set. There is no target feature or single feature of interest in case of unsupervised learning. Based on the value of all features, interesting patterns or insights are derived about the data set.

Descriptive models which group together similar data instances, i.e. data instances having a similar value of the different features are called clustering models. Examples of clustering include

(i) Customer grouping or segmentation based on social, demographic, ethnic, etc. factors
(ii) Grouping of music based on different aspects like genre, language, time-period, etc.
(iii) Grouping of commodities in an inventory

The most popular model for clustering is *k*-Means.

Descriptive models related to pattern discovery is used for market basket analysis of transactional data. In market basket analysis, based on the purchase pattern available in the transactional data, the possibility of purchasing one product based on the purchase of another product is determined. For example, transactional data may reveal a pattern that generally a customer who purchases milk also purchases biscuit at the same time. This can be useful for targeted promotions or in-store set up. Promotions related to biscuits can be sent to customers of milk products or vice versa. Also, in the store products related to milk can be placed close to biscuits.

3.3 TRAINING A MODEL (FOR SUPERVISED LEARNING)

3.3.1 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 (depicted in Figure 3.1), which is by holding back a part of the input data for validating the trained model is known as holdout method.

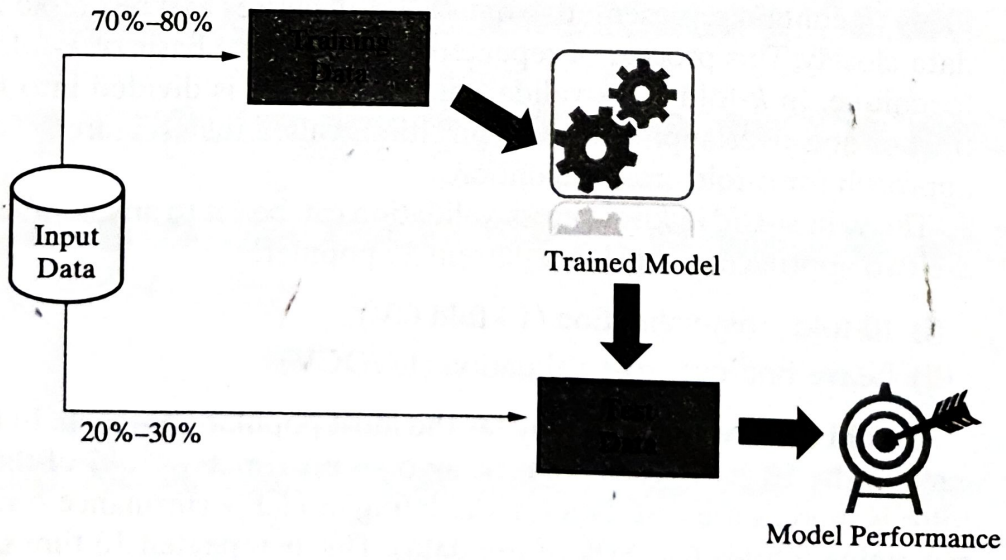

FIG. 3.1
Holdout method

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.

3.3.2 *K*-fold Cross-validation method

Holdout method employing stratified random sampling approach still heads into issues in certain specific situations. Especially, the smaller data sets 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. Figure 3.2 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:

(i) 10-fold cross-validation (10-fold CV)

(ii) 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 of 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. Figure 3.3 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.

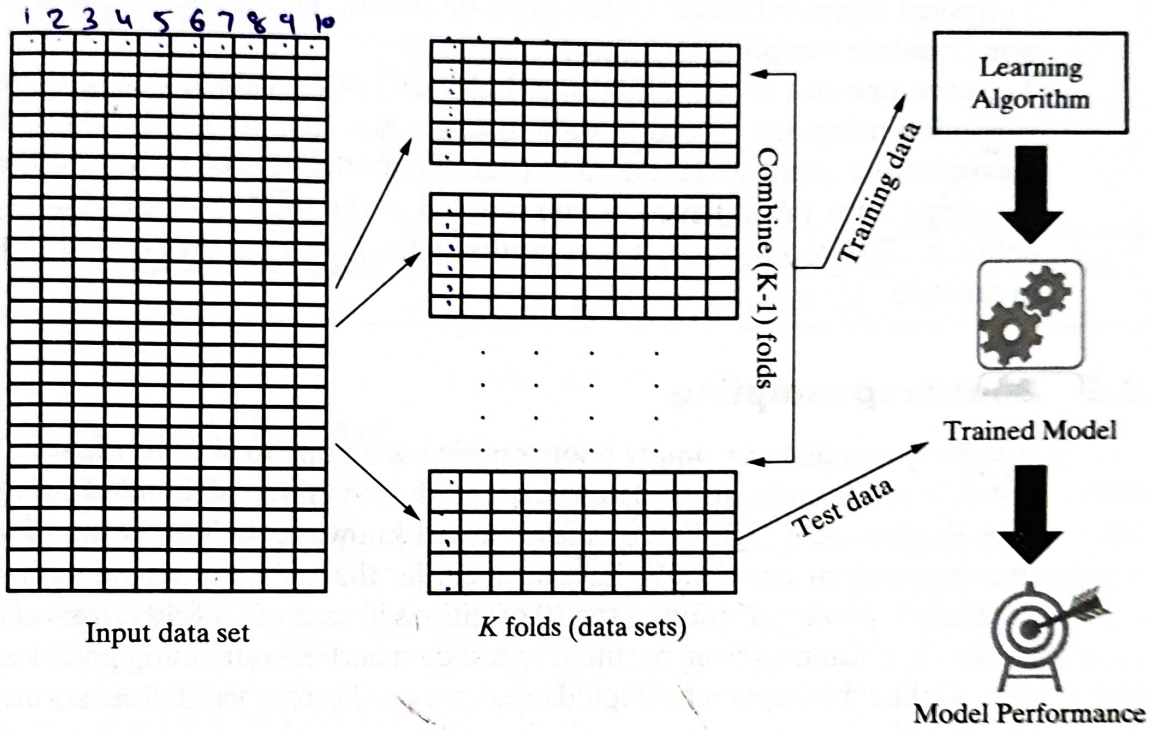

FIG. 3.2
Overall approach for *K*-fold cross-validation

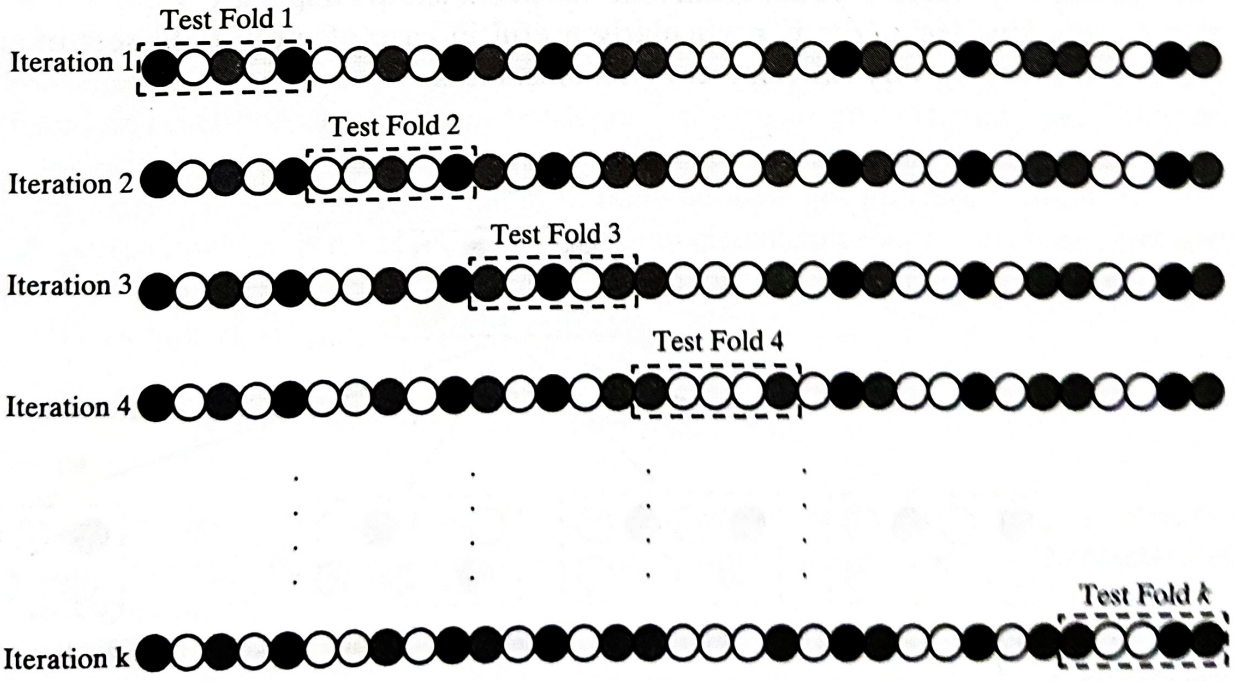

FIG. 3.3
Detailed approach for fold selection

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 test data set in each of the 'k' iterations is different. Also, note that though in figure 3.3 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.

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.

3.3.3 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. Figure 3.4 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.

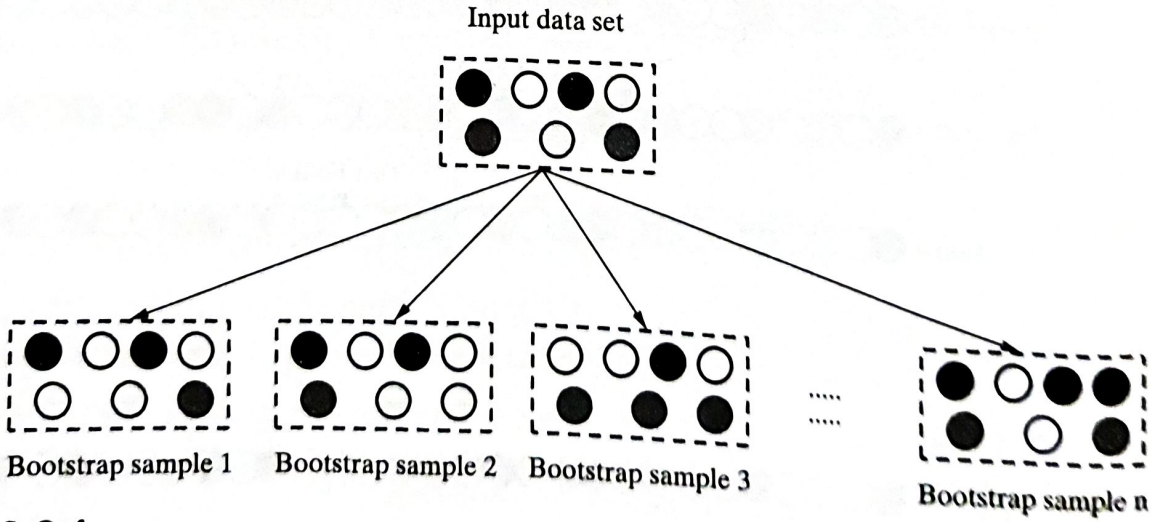

FIG. 3.4
Bootstrap sampling

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.

3.3.4 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 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, 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 algorithm for lazy learning is k-nearest neighbor.

Note:

Parametric learning models have finite number of parameters. In case of non-parametric models, quite contradicting to its name, the number of parameters is potentially infinite.

Models such as Linear Regression and Support Vector Machine, since the coefficients form the learning parameters, they are fixed in size. Hence, these models are clubbed as parametric.

On the other hand, in case of models such as *k*-Nearest Neighbor (*k*NN) and decision tree, number of parameters grows with the size of the training data. Hence, they are considered as non-parametric learning models.

3.4 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.

3.4.1 Underfitting

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 underfitting may occur when trying to represent a non-linear data with a linear model as demonstrated by both cases of underfitting shown in figure 3.5. Many times underfitting happens due to unavailability of sufficient training data. Underfitting results in both poor performance with training data as well as poor generalization to test data. Underfitting can be avoided by

1. using more training data
2. reducing features by effective feature selection

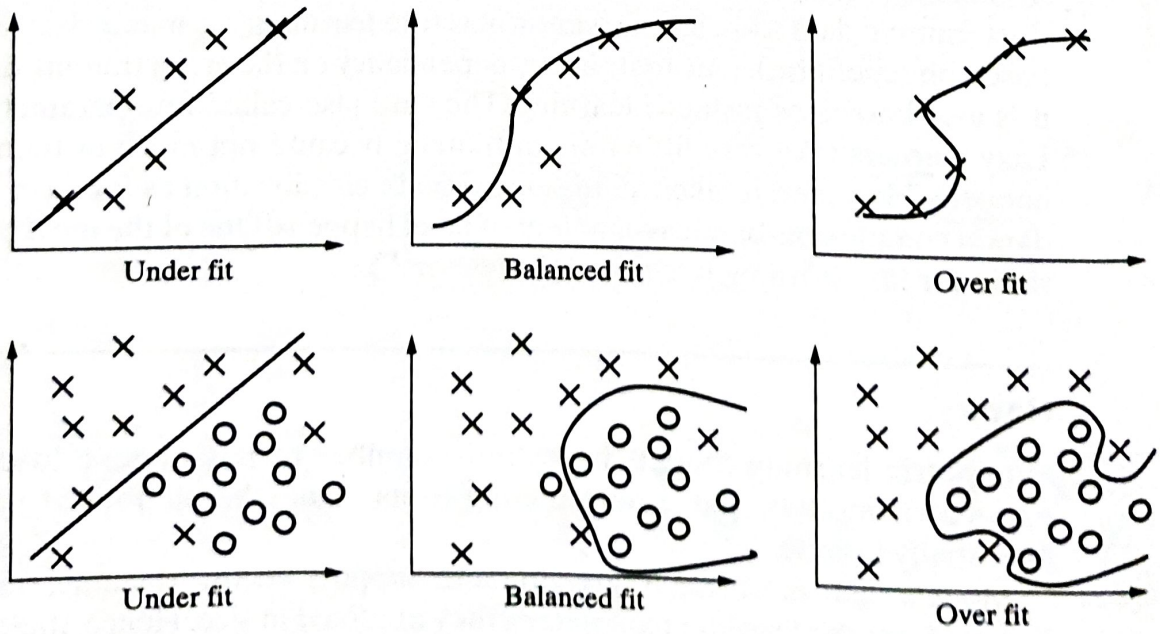

FIG. 3.5
Underfitting and Overfitting of models

3.4.2 Overfitting

Overfitting 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. Overfitting, 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 in figure 3.5. 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.

3.4.3 Bias – variance trade-off

In supervised learning, the class value assigned by the learning model built based on the training data may differ from the actual class value. This error in learning can be of two types – errors due to 'bias' and error due to 'variance'. Let's try to understand each of them in details.

3.4.3.1 Errors due to 'Bias'

Errors due to bias arise from simplifying assumptions made by the model to make the target function less complex or easier to learn. In short, it is due to underfitting of the model. Parametric models generally have high bias making them easier to understand/interpret and faster to learn. These algorithms have a poor performance on data sets, which are complex in nature and do not align with the simplifying assumptions made by the algorithm. Underfitting results in high bias.

3.4.3.2 Errors due to 'Variance'

Errors due to variance occur from difference in training data sets used to train the model. Different training data sets (randomly sampled from the input data set) are used to train the model. Ideally the difference in the data sets should not be significant and the model trained using different training data sets should not be too different. However, in case of overfitting, since the model closely matches the training data, even a small difference in training data gets magnified in the model.

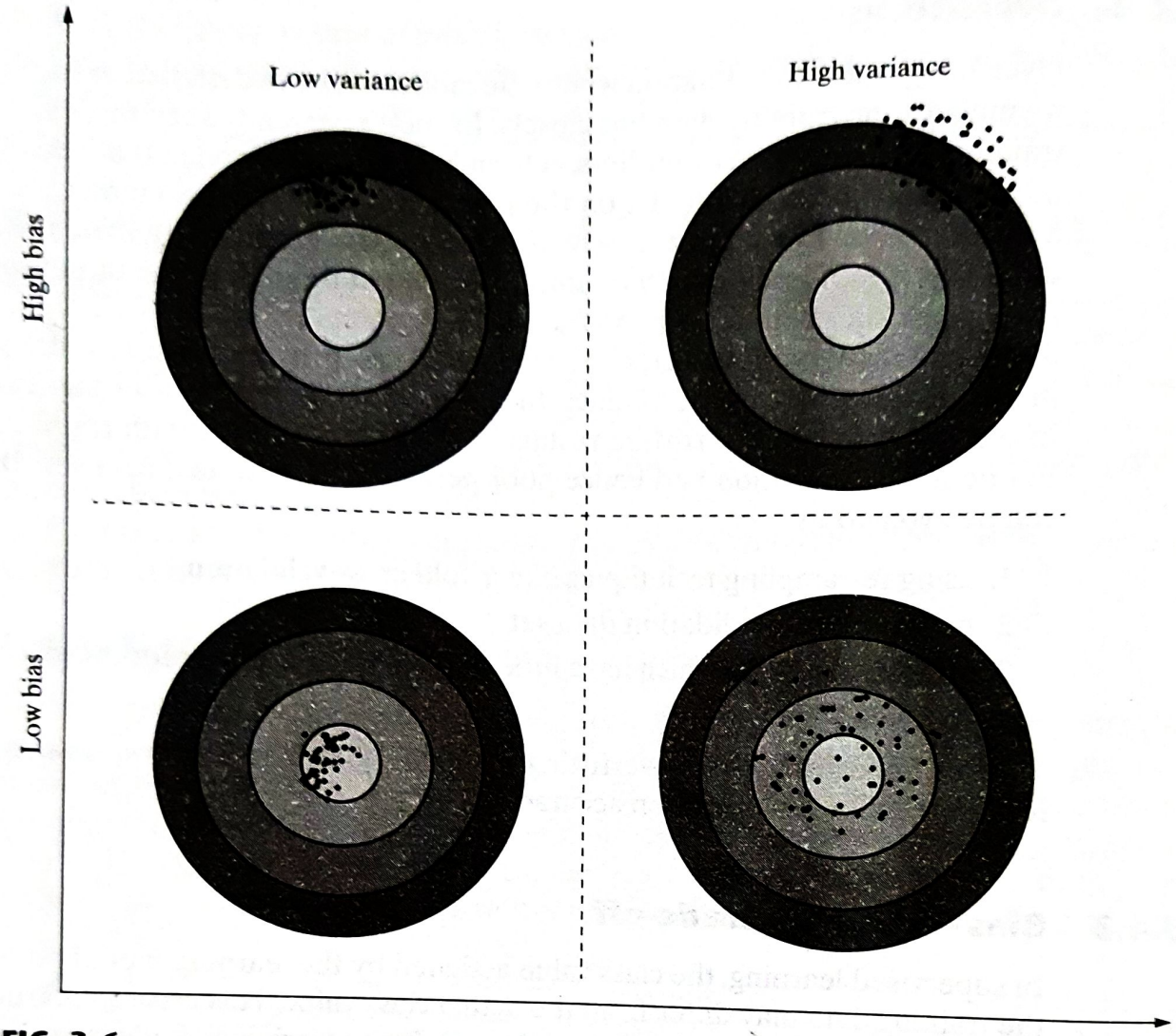

FIG. 3.6
Bias-variance trade-off

So, the problems in training a model can either happen because either (a) the model is too simple and hence fails to interpret the data grossly or (b) the model is extremely complex and magnifies even small differences in the training data.

As is quite understandable:

- Increasing the bias will decrease the variance, and
- Increasing the variance will decrease the bias

On one hand, parametric algorithms are generally seen to demonstrate high bias but low variance. On the other hand, non-parametric algorithms demonstrate low bias and high variance.

As can be observed in Figure 3.6, the best solution is to have a model with low bias as well as low variance. However, that may not be possible in reality. Hence, the goal of supervised machine learning is to achieve a balance between bias and variance. The learning algorithm chosen and the user parameters which can be configured helps in striking a trade-off between bias and variance. For example, in a popular supervised algorithm *k*-Nearest Neighbors or *k*NN, the user configurable parameter '*k*' can be used to do a trade-off between bias and variance. In one hand, when the value of '*k*'

is decreased, the model becomes simpler to fit and bias increases. On the other hand, when the value of 'k' is increased, the variance increases.

3.5 EVALUATING PERFORMANCE OF A MODEL

3.5.1 Supervised learning - classification

In supervised learning, one major task is classification. The responsibility of the classification model is to assign class label to the target feature based on the value of the predictor features. For example, in the problem of predicting the win/loss in a cricket match, the classifier will assign a class value win/loss to target feature based on the values of other features like whether the team won the toss, number of spinners in the team, number of wins the team had in the tournament, etc. To evaluate the performance of the model, the number of correct classifications or predictions made by the model has to be recorded. A classification is said to be correct if, say for example in the given problem, it has been predicted by the model that the team will win and it has actually won.

Based on the number of correct and incorrect classifications or predictions made by a model, the accuracy of the model is calculated. If 99 out of 100 times the model has classified correctly, e.g. if in 99 out of 100 games what the model has predicted is same as what the outcome has been, then the model accuracy is said to be 99%. However, it is quite relative to say whether a model has performed well just by looking at the accuracy value. For example, 99% accuracy in case of a sports win predictor model may be reasonably good but the same number may not be acceptable as a good threshold when the learning problem deals with predicting a critical illness. In this case, even the 1% incorrect prediction may lead to loss of many lives. So the model performance needs to be evaluated in light of the learning problem in question. Also, in certain cases, erring on the side of caution may be preferred at the cost of overall accuracy. For that reason, we need to look more closely at the model accuracy and also at the same time look at other measures of performance of a model like sensitivity, specificity, precision, etc. So, let's start with looking at model accuracy more closely. And let's try to understand it with an example.

There are four possibilities with regards to the cricket match win/loss prediction:

1. the model predicted win and the team won
2. the model predicted win and the team lost
3. the model predicted loss and the team won
4. the model predicted loss and the team lost

In this problem, the obvious class of interest is 'win'.

The first case, i.e. the model predicted win and the team won is a case where the model has correctly classified data instances as the class of interest. These cases are referred as True Positive (TP) cases.

The second case, i.e. the model predicted win and the team lost is a case where the model incorrectly classified data instances as the class of interest. These cases are referred as False Positive (FP) cases.

The third case, i.e. the model predicted loss and the team won is a case where the model has incorrectly classified as not the class of interest. These cases are referred as False Negative (FN) cases.

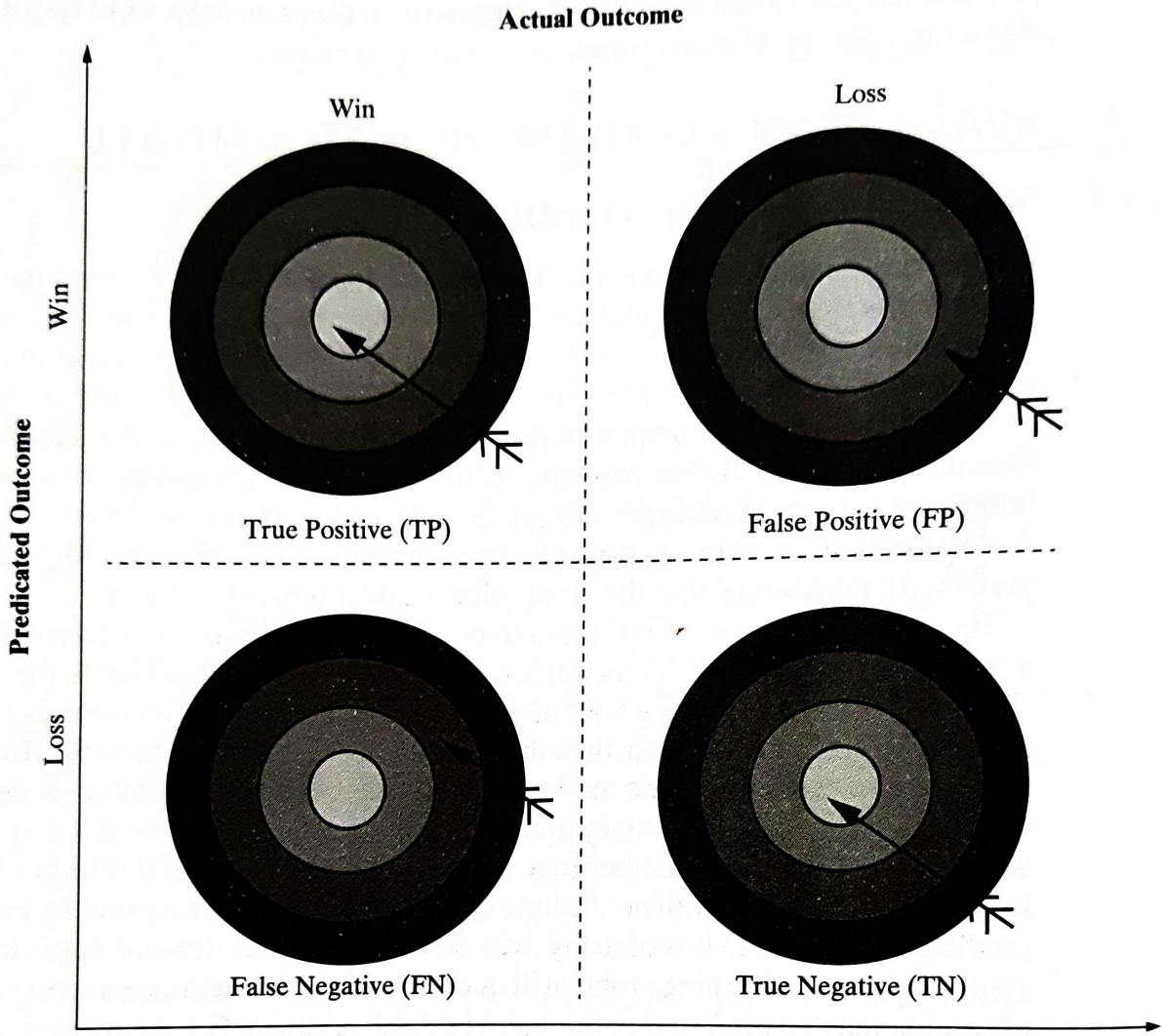

FIG. 3.7
Details of model classification

The fourth case, i.e. the model predicted loss and the team lost is a case where the model has correctly classified as not the class of interest. These cases are referred as True Negative (TN) cases. All these four cases are depicted in Figure 3.7.

For any classification model, **model accuracy** is given by total number of correct classifications (either as the class of interest, i.e. True Positive or as not the class of interest, i.e. True Negative) divided by total number of classifications done.

$$\text{Model accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

A matrix containing correct and incorrect predictions in the form of TPs, FPs, FNs and TNs is known as **confusion matrix**. The win/loss prediction of cricket match has two classes of interest – win and loss. For that reason it will generate a 2 × 2 confusion matrix. For a classification problem involving three classes, the confusion matrix would be 3 × 3, etc.

Let's assume the confusion matrix of the win/loss prediction of cricket match problem to be as below:

	ACTUAL WIN	ACTUAL LOSS
Predicted Win	85	4
Predicted Loss	2	9

In context of the above confusion matrix, total count of TPs = 85, count of FPs = 4, count of FNs = 2 and count of TNs = 9.

$$\therefore \text{Model accuracy} = \frac{TP + TN}{TP + FP + FN + TN} = \frac{85 + 9}{85 + 4 + 2 + 9} = \frac{94}{100} = 94\%$$

The percentage of misclassifications is indicated using **error rate** which is measured as

$$\text{Error rate} = \frac{FP + FN}{TP + FP + FN + TN}$$

In context of the above confusion matrix,

$$\text{Error rate} = \frac{FP + FN}{TP + FP + FN + TN} = \frac{4 + 2}{85 + 4 + 2 + 9} = \frac{6}{100} = 6\%$$

$$= 1 - \text{Model accuracy}$$

Sometimes, correct prediction, both TPs as well as TNs, may happen by mere coincidence. Since these occurrences boost model accuracy, ideally it should not happen. **Kappa** value of a model indicates the adjusted the model accuracy. It is calculated using the formula below:

$$\text{Kappa value (k)} = \frac{P(a) - P(p_r)}{1 - P(p_r)}$$

P(a) = Proportion of observed agreement between actual and predicted in overall data set

$$= \frac{TP + TN}{TP + FP + FN + TN}$$

$P(p_r)$ = Proportion of expected agreement between actual and predicted data both in case of class of interest as well as the other classes

$$= \frac{TP + FP}{TP + FP + FN + TN} \times \frac{TP + FN}{TP + FP + FN + TN} + \frac{FN + TN}{TP + FP + FN + TN} \times \frac{FP + TN}{TP + FP + FN + TN}$$

In context of the above confusion matrix, total count of TPs = 85, count of FPs = 4, count of FNs = 2 and count of TNs = 9.

$$\therefore P(a) = \frac{TP + TN}{TP + FP + FN + TN} = \frac{85 + 9}{85 + 4 + 2 + 9} = \frac{94}{100} = 0.94$$

$$P(p_r) = \frac{85 + 4}{85 + 4 + 2 + 9} \times \frac{85 + 2}{85 + 4 + 2 + 9} + \frac{2 + 9}{85 + 4 + 2 + 9} \times \frac{4 + 9}{85 + 4 + 2 + 9}$$

$$= \frac{89}{100} \times \frac{87}{100} + \frac{11}{100} \times \frac{13}{100} = 0.89 \times 0.87 + 0.11 \times 0.13 = 0.7886$$

$$\therefore k = \frac{0.94 - 0.7886}{1 - 0.7886} = 0.7162$$

Note:

Kappa value can be 1 at the maximum, which represents perfect agreement between model's prediction and actual values.

As discussed earlier, in certain learning problems it is critical to have extremely low number of FN cases, if needed, at the cost of a conservative classification model. Though it is a clear case of misclassification and will impact model accuracy adversely, it is still required as missing each class of interest may have serious consequence. This happens more in problems from medical domains like disease prediction problem. For example, if a tumor is malignant but wrongly classified as benign by the classifier, then the repercussion of such misclassification is fatal. It does not matter if higher number of tumours which are benign are wrongly classified as malignant. In these problems there are some measures of model performance which are more important than accuracy. Two such critical measurements are sensitivity and specificity of the model.

The **sensitivity** of a model measures the proportion of TP examples or positive cases which were correctly classified. It is measured as

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

In the context of the above confusion matrix for the cricket match win prediction problem,

$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{85}{85 + 2} = \frac{85}{87} = 97.7\%$$

So, again taking the example of the malignancy prediction of tumours, class of interest is 'malignant'. Sensitivity measure gives the proportion of tumours which are actually malignant and have been predicted as malignant. It is quite obvious that for such problems the most critical measure of the performance of a good model is sensitivity. A high value of sensitivity is more desirable than a high value of accuracy being excessively conservative or excessively aggressive. Specificity of a model measures the proportion of negative examples which have been correctly classified. In the context, of malignancy prediction of tumours, specificity gives the proportion of benign tumours which have been correctly classified. In the context of the above confusion matrix for the cricket match win prediction problem,

Specificity is also another good measure to indicate a good balance of a model

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} = \frac{9}{9 + 4} = \frac{9}{13} = 69.2\%$$

A higher value of specificity will indicate a better model performance. However, it is quite understandable that a conservative approach to reduce False Negatives might actually push up the number of FPs. Reason for this is that the model, in order to reduce FNs, is going to classify more tumours as malignant. So the chance that benign tumours will be classified as malignant or FPs will increase.

There are two other performance measures of a supervised learning model which are similar to sensitivity and specificity. These are **precision** and **recall**. While precision gives the proportion of positive predictions which are truly positive, recall gives the proportion of TP cases over all actually positive cases.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision indicates the reliability of a model in predicting a class of interest. When the model is related to win / loss prediction of cricket, precision indicates how often it predicts the win correctly. In context of the above confusion matrix for the cricket match win prediction problem,

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{85}{85 + 4} = \frac{85}{89} = 95.5\%$$

It is quite understandable that a model with higher precision is perceived to be more reliable.

Recall indicates the proportion of correct prediction of positives to the total number of positives. In case of win/loss prediction of cricket, recall resembles what proportion of the total wins were predicted correctly.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

In the context of the above confusion matrix for the cricket match win prediction problem,

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{85}{85 + 2} = \frac{85}{87} = 97.7\%$$

3.5.1.1 *F-measure*

F-measure is another measure of model performance which combines the precision and recall. It takes the harmonic mean of precision and recall as calculated as

$$F\text{-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

In context of the above confusion matrix for the cricket match win prediction problem,

$$F\text{-measure} = \frac{2 \times 0.955 \times 0.977}{0.955 + 0.977} = \frac{1.866}{1.932} = 96.6\%$$

As a combination of multiple measures into one, *F*-score gives the right measure using which performance of different models can be compared. However, one assumption the calculation is based on is that precision and recall have equal weight, which may not always be true in reality. In certain problems, the disease prediction problems, e.g., precision may be given far more weightage. In that case, different weightages may be assigned to precision and recall. However, there may be a serious dilemma regarding what value to be adopted for each and what is the basis for the specific value adopted.

3.5.1.1.1 Receiver operating characteristic (ROC) curves

As we have seen till now, though accuracy is the most popular measure, there are quite a number of other measures to evaluate the performance of a supervised learning model. However, visualization is an easier and more effective way to understand the model performance. It also helps in comparing the efficiency of two models.

Receiver Operating Characteristic (ROC) curve helps in visualizing the performance of a classification model. It shows the efficiency of a model in the detection of true positives while avoiding the occurrence of false positives. To refresh our memory, true positives are those cases where the model has correctly classified data instances as the class of interest. For example, the model has correctly classified the tumours as malignant, in case of a tumour malignancy prediction problem. On the other hand, FPs are those cases where the model incorrectly classified data instances as the class of interest. Using the same example, in this case, the model has incorrectly classified the tumours as malignant, i.e. tumours which are actually benign have been classified as malignant.

$$\text{True Positive Rate TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False Positive Rate FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

In the ROC curve, the FP rate is plotted (in the horizontal axis) against true positive rate (in the vertical axis) at different classification thresholds. If we assume a lower value of classification threshold, the model classifies more items as positive. Hence, the values of both False Positives and True Positives increase. The area under curve (AUC) value, as shown in figure 3.8a, is the area of the two-dimensional space under the curve extending from (0, 0) to (1, 1), where each point on the curve gives a set of true and false positive values at a specific classification threshold. This curve gives an indication of the predictive quality of a model. AUC value ranges from 0 to 1, with an AUC of less than 0.5 indicating that the classifier has no predictive ability.

Figure 3.8b shows the curves of two classifiers – classifier 1 and classifier 2. Quite obviously, the AUC of classifier 1 is more than the AUC of classifier 2. So, we can draw the inference that classifier 1 is better than classifier 2.

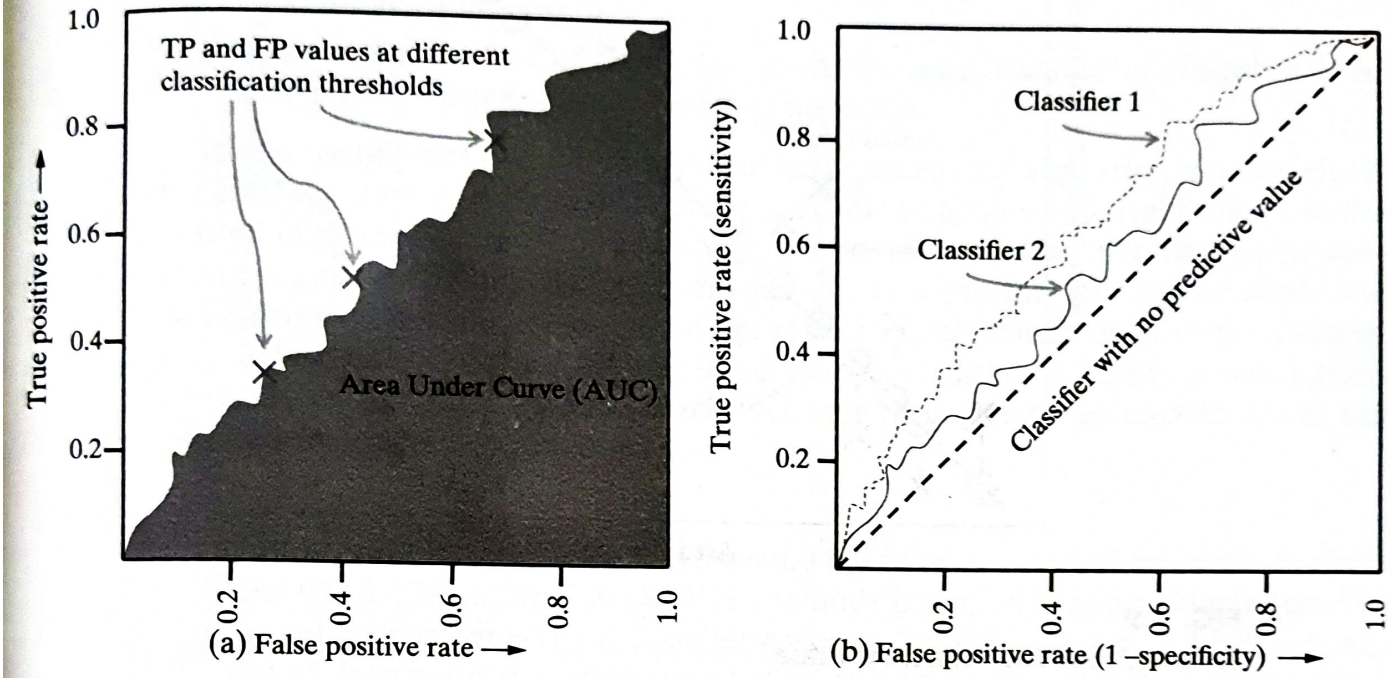

FIG. 3.8
ROC curve

A quick indicative interpretation of the predictive values from 0.5 to 1.0 is given below:

- 0.5 – 0.6 → Almost no predictive ability
- 0.6 – 0.7 → Weak predictive ability
- 0.7 – 0.8 → Fair predictive ability
- 0.8 – 0.9 → Good predictive ability
- 0.9 – 1.0 → Excellent predictive ability

3.5.2 Supervised learning – regression

A well-fitted regression model churns out predicted values close to actual values. Hence, a regression model which ensures that the difference between predicted and actual values is low can be considered as a good model. Figure 3.9 represents a very simple problem of real estate value prediction solved using linear regression model. If 'area' is the predictor variable (say x) and 'value' is the target variable (say y), the linear regression model can be represented in the form:

$$y = \alpha + \beta x$$

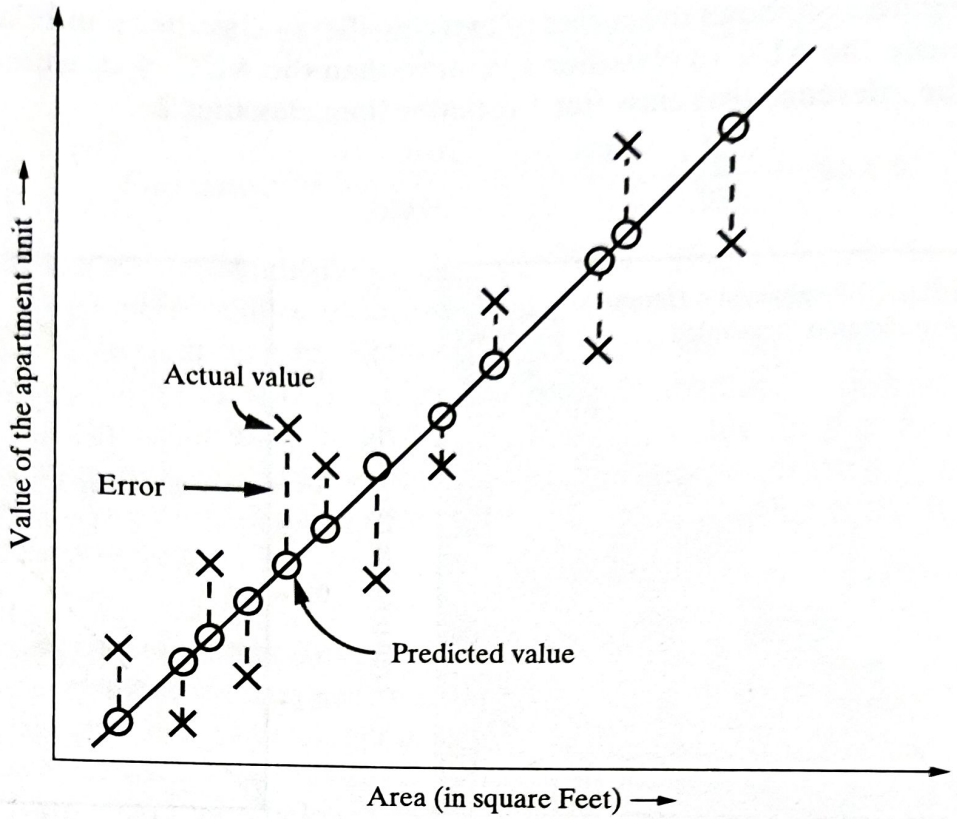

FIG. 3.9
Error – Predicted vs. actual value

For a certain value of x, say $\hat{x}$, the value of y is predicted as $\hat{y}$ whereas the actual value of y is Y (say). The distance between the actual value and the fitted or predicted value, i.e. $\hat{y}$ is known as **residual**. The regression model can be considered to be fitted well if the difference between actual and predicted value, i.e. the residual value is less.

R-squared is a good measure to evaluate the model fitness. It is also known as the coefficient of determination, or for multiple regression, the coefficient of multiple determination. The R-squared value lies between 0 to 1 (0%–100%) with a larger value representing a better fit. It is calculated as:

$$R^2 = \frac{SST - SSE}{SST}$$

Sum of Squares Total (SST) = squared differences of each observation from the overall mean = $\sum_{i=1}^{n}(y_i - \bar{y})^2$ where $\bar{y}$ is the mean.

Sum of Squared Errors (SSE) (of prediction) = sum of the squared residuals = $\sum_{i=1}^{n}(Y_i - \hat{y})^2$ where $\hat{y}_i$ is the predicted value of y_i and Y_i is the actual value of y_i.

3.5.3 Unsupervised learning - clustering

Clustering algorithms try to reveal natural groupings amongst the data sets. However, it is quite tricky to evaluate the performance of a clustering algorithm. Clustering, by

nature, is very subjective and whether the cluster is good or bad is open for interpretations. It was noted, 'clustering is in the eye of the beholder'. This stems from the two inherent challenges which lie in the process of clustering:

1. It is generally not known how many clusters can be formulated from a particular data set. It is completely open-ended in most cases and provided as a user input to a clustering algorithm.
2. Even if the number of clusters is given, the same number of clusters can be formed with different groups of data instances.

In a more objective way, it can be said that a clustering algorithm is successful if the clusters identified using the algorithm is able to achieve the right results in the overall problem domain. For example, if clustering is applied for identifying customer segments for a marketing campaign of a new product launch, the clustering can be considered successful only if the marketing campaign ends with a success, i.e. it is able to create the right brand recognition resulting in steady revenue from new product sales. However, there are couple of popular approaches which are adopted for cluster quality evaluation.

(a) *Internal evaluation*

In this approach, the cluster is assessed based on the underlying data that was clustered. The internal evaluation methods generally measure cluster quality based on homogeneity of data belonging to the same cluster and heterogeneity of data belonging to different clusters. The homogeneity/heterogeneity is decided by some similarity measure. For example, **silhouette coefficient**, which is one of the most popular internal evaluation methods, uses distance (Euclidean or Manhattan distances most commonly used) between data elements as a similarity measure. The value of silhouette width ranges between –1 and +1, with a high value indicating high intra-cluster homogeneity and inter-cluster heterogeneity.

For a data set clustered into 'k' clusters, silhouette width is calculated as:

$$\text{Silhouette width} = \frac{b(i) - a(i)}{\max\{a(i),\ b(i)\}}$$

$a(i)$ is the average distance between the i th data instance and all other data instances belonging to the same cluster and $b(i)$ is the lowest average distance between the i-the data instance and data instances of all other clusters.

Let's try to understand this in context of the example depicted in figure 3.10. There are four clusters namely cluster 1, 2, 3, and 4. Let's consider an arbitrary data element 'i' in cluster 1, resembled by the asterisk. $a(i)$ is the average of the distances $a_{i1}, a_{i2}, \ldots, a_{in_1}$ of the different data elements from the i th data element in cluster 1, assuming there are n_1 data elements in cluster 1. Mathematically,

$$a(i) = \frac{a_{i1} + a_{i2} + \ldots + a_{in_1}}{n_1}$$

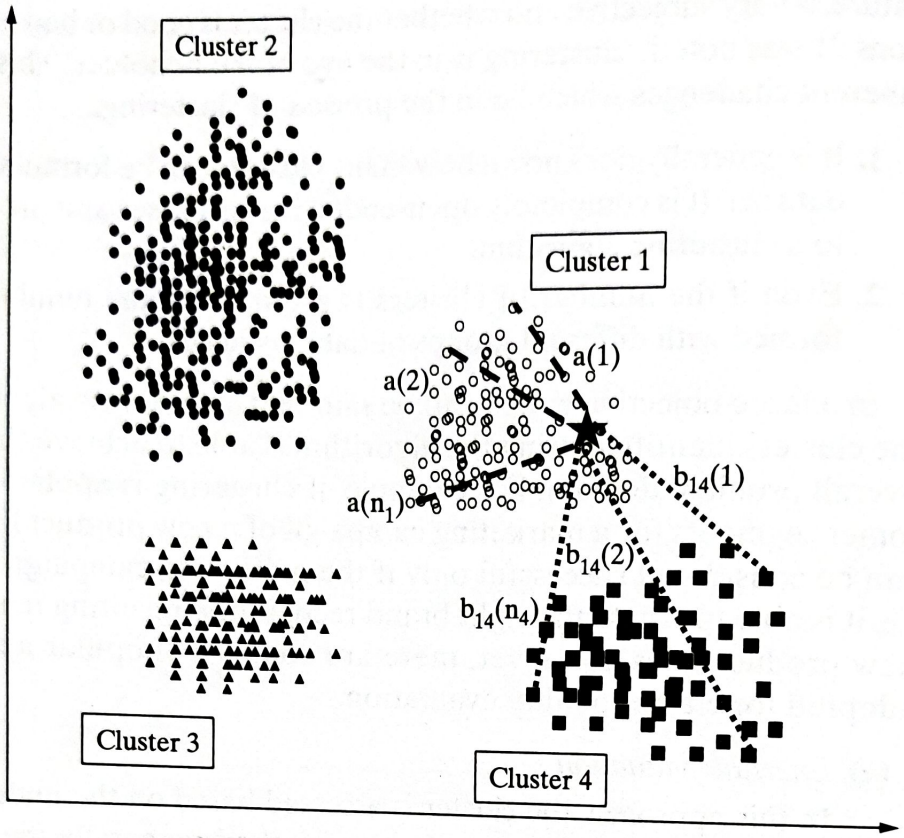

FIG. 3.10
Silhouette width calculation

In the same way, let's calculate the distance of an arbitrary data element '*i*' in cluster 1 with the different data elements from another cluster, say cluster 4 and take an average of all those distances. Hence,

$$b_{14}\,(\text{average}) = \frac{b_{14}(1) + b_{14}(2) + \ldots + b_{14}(n_4)}{(n_4)}$$

where n_4 is the total number of elements in cluster 4. In the same way, we can calculate the values of b_{12} (average) and b_{13} (average). $b\ (i)$ is the minimum of all these values. Hence, we can say that,

$$b(\text{i}) = \text{minimum}\ [b_{12}(\text{average}),\ b_{13}(\text{average}),\ b_{14}(\text{average})]$$

(b) *External evaluation*

In this approach, class label is known for the data set subjected to clustering. However, quite obviously, the known class labels are not a part of the data used in clustering. The cluster algorithm is assessed based on how close the results are compared to those known class labels. For example, **purity** is one of the most popular measures of cluster algorithms – evaluates the extent to which clusters contain a single class.

For a data set having '*n*' data instances and '*c*' known class labels which generates '*k*' clusters, purity is measured as:

$$\text{Purity} = \frac{1}{n}\sum_{k}\max(k \cap c)$$

3.6 IMPROVING PERFORMANCE OF A MODEL

Now we have almost reached the end of the journey of building learning models. We have got some idea about what modelling is, how to approach about it to solve a learning problem and how to measure the success of our model. Now comes a million dollar question. Can we improve the performance of our model? If so, then what are the levers for improving the performance? In fact, even before that comes the question of model selection – which model should be selected for which machine learning task? We have already discussed earlier that the model selection is done one several aspects:

1. Type of learning the task in hand, i.e. supervised or unsupervised
2. Type of the data, i.e. categorical or numeric
3. Sometimes on the problem domain
4. Above all, experience in working with different models to solve problems of diverse domains

So, assuming that the model selection is done, what are the different avenues to improve the performance of models?

One effective way to improve model performance is by tuning model parameter. **Model parameter tuning** is the process of adjusting the model fitting options. For example, in the popular classification model *k*-Nearest Neighbour (*k*NN), using different values of '*k*' or the number of nearest neighbours to be considered, the model can be tuned. In the same way, a number of hidden layers can be adjusted to tune the performance in neural networks model. Most machine learning models have at least one parameter which can be tuned.

As an alternate approach of increasing the performance of one model, several models may be combined together. The models in such combination are complimentary to each other, i.e. one model may learn one type data sets well while struggle with another type of data set. Another model may perform well with the data set which the first one struggled with. This approach of combining different models with diverse strengths is known as **ensemble** (depicted in Figure 3.11). Ensemble helps in averaging out biases of the different underlying models and also reducing the variance. Ensemble methods combine weaker learners to create stronger ones. A performance boost can be expected even if models are built as usual and then ensembled. Following are the typical steps in ensemble process:

- Build a number of models based on the training data
- For diversifying the models generated, the training data subset can be varied using the allocation function. Sampling techniques like bootstrapping may be used to generate unique training data sets.
- Alternatively, the same training data may be used but the models combined are quite varying, e.g, SVM, neural network, *k*NN, etc.
- The outputs from the different models are combined using a combination function. A very simple strategy of combining, say in case of a prediction task using ensemble, can be majority voting of the different models combined. For example, 3 out of 5 classes predict 'win' and 2 predict 'loss' – then the final outcome of the ensemble using majority vote would be a 'win'.

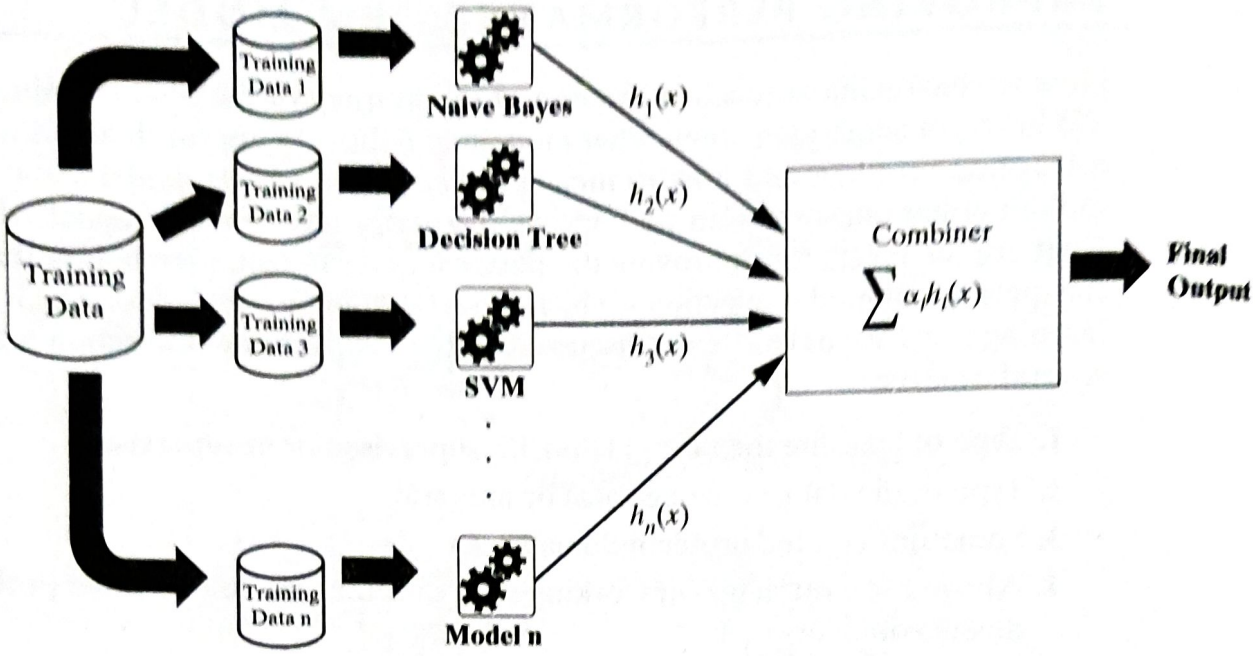

FIG. 3.11
Ensemble

One of the earliest and most popular ensemble models is **bootstrap aggregating** or **bagging**. Bagging uses bootstrap sampling method (refer section 3.3.3) to generate multiple training data sets. These training data sets are used to generate (or train) a set of models using the same learning algorithm. Then the outcomes of the models are combined by majority voting (classification) or by average (regression). Bagging is a very simple ensemble technique which can perform really well for unstable learners like a decision tree, in which a slight change in data can impact the outcome of a model significantly.

Just like bagging, **boosting** is another key ensemble-based technique. In this type of ensemble, weaker learning models are trained on resampled data and the outcomes are combined using a weighted voting approach based on the performance of different models. **Adaptive boosting** or **AdaBoost** is a special variant of boosting algorithm. It is based on the idea of generating weak learners and slowly learning

Random forest is another ensemble-based technique. It is an ensemble of decision trees – hence the name random forest to indicate a forest of decision trees. It has been discussed in more details in chapter 7.

In this chapter, you have been introduced to the crux of machine learning, i.e. modelling. Thorough understanding of the technical aspects elaborated in this chapter is extremely crucial for the success of any machine learning project. For example, the first dilemma comes about which model to select. Again, in case of supervised learning, how can we deal with the unavailability of sufficient training data. In the same way, once the model is trained in case of supervised learning or the grouping is done in case of clustering, how we can understand whether the model training (for supervised) or grouping done (for unsupervised) is good or bad. All these and more have been addressed as a part of this chapter.

3.7 SUMMARY

- Structured representation of raw input data to the meaningful pattern is called a model.
- The process of fitting a specific model to a data set is called model training.
- Models for supervised learning or predictive models try to predict certain value using the input data set.
- Models for unsupervised learning or descriptive models are used to describe a data set or gain insight from a data set.
- The method of partitioning the input data into two parts – training and test data, which is holding back a part of the input data for validating the trained model is known as holdout method.
- In *k*-fold cross-validation technique, the data set is divided into *k*- completely separate random partitions called folds. It is basically repeated holdout into 'k' folds. The value of '*k*' in *k*-fold cross-validation can be set to any number. Two extremely popular approaches are:
  - ❖ 10-fold cross-validation (10-fold CV)
  - ❖ Leave-one-out cross-validation (LOOCV)
- 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). Bootstrapping randomly picks data instances from the input data set, with the possibility of the same data instance to be picked multiple times.
- Target function of a model is the function defining the relationship between the input (also called predictor or independent) variables and the output (also called response or dependent or target) variable. It is represented in the general form: $Y = f(X) + e$, where Y is the output variable, X represents the input variables and 'e' is a random error term.
- Fitness of a target function approximated by a learning algorithm determines how correctly it is able to predict the value or class for a set of data it has never seen.
- If the target function is kept too simple, it may not be able to capture the essential nuances and represent the underlying data well. This known as underfitting.
- Overfitting 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 nuance 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.
- In supervised learning, the value predicted by the learning model built based on the training data may differ from the actual class value. This error in learning can be of two types – errors due to 'bias' and error due to 'variance'. Errors due to bias arise from simplifying assumptions made by the model whereas errors due to variance occur from over-aligning the model with the training data sets.
- For any classification model, model accuracy is the primary indicator of the goodness of the model. It is given by a total number of correct classifications (either as the class of interest, or as not the class of interest) divided by total number of

classifications done. There are other indicators like error rate, sensitivity, specificity, precision and recall.

- For unsupervised learning (clustering), silhouette coefficient (or width) is one of the most popular internal evaluation methods. A high value of silhouette width indicates high intra-cluster homogeneity and inter-cluster heterogeneity. In case, class label is known for the data set, purity is another popular measure which evaluates the extent to which clusters contain a single class.
- Model parameter tuning is the process of adjusting the model fitting options. For example, in the popular classification model *k*-Nearest Neighbour (*k*NN), using different values of '*k*' or the number of nearest neighbours to be considered, the model can be tuned.
- The approach of combining different models with diverse strengths is known as ensemble. Ensemble methods combine weaker learners to create stronger ones.
- One of the earliest and most popular ensemble models is bootstrap aggregating or bagging. Bagging uses bootstrapping to generate multiple training data sets. These training data sets are used to generate a set of models using the same learning algorithm.
- Just like bagging, boosting is another key ensemble-based technique. In boosting, weaker learning models are trained on resampled data and the outcomes are combined using a weighted voting approach based on the performance of different models.
- Adaptive boosting or AdaBoost is a special variant of boosting algorithm.

SAMPLE QUESTIONS

MULTIPLE-CHOICE QUESTIONS (1 MARK QUESTIONS):

1. Structured representation of raw input data to meaningful ___ is called a model.
(a) pattern (b) data
(c) object (d) none of the above

2. For supervised learning we have ____ model.
(a) interactive (b) predictive
(c) descriptive (d) prescriptive

3. For unsupervised learning we have ____ model.
(a) interactive (b) predictive
(c) descriptive (d) prescriptive

4. Which of the following measure is not used for a classification model?
(a) Accuracy (b) Recall
(c) Purity (d) Error rate

5. Which of the following is a performance measure for regression?
(a) Accuracy (c) RMSE
(b) Recall (d) Error rate

6. Which of the following is the measure of cluster quality?
   (a) Purity
   (b) Distance
   (c) Accuracy
   (d) all of the above

7. Out of 200 emails, a classification model correctly predicted 150 spam emails and 30 ham emails. What is the accuracy of the model?
   (a) 10%
   (b) 90%
   (c) 80%
   (d) none of the above

8. Out of 200 emails, a classification model correctly predicted 150 spam emails and 30 ham emails. What is the error rate of the model?
   (a) 10%
   (b) 90%
   (c) 80%
   (d) none of the above

9. There is no one model that works best for every machine learning problem. This is stated as
   (a) Fit gap model theorem
   (b) One model theorem
   (c) Free lunch theorem
   (d) No free lunch theorem

10. LOOCV in machine learning stands for
   (a) Love one-out cross validation
   (b) Leave-one-out cross-validation
   (c) Leave-object oriented cross-validation
   (d) Leave-one-out class-validation