

# 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  $X_1, X_2, X_3, \dots, X_n$  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
  1. Classification
  2. Regression
2. Unsupervised
  1. Clustering
  2. 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:

1. Predicting win/loss in a cricket match
2. Predicting whether a transaction is fraud
3. 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:

1. Prediction of revenue growth in the succeeding year
2. Prediction of rainfall amount in the coming monsoon
3. 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

1. Customer grouping or segmentation based on social, demographic, ethnic, etc. factors
2. Grouping of music based on different aspects like genre, language, time-period, etc.
3. Grouping of commodities in an inventory

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

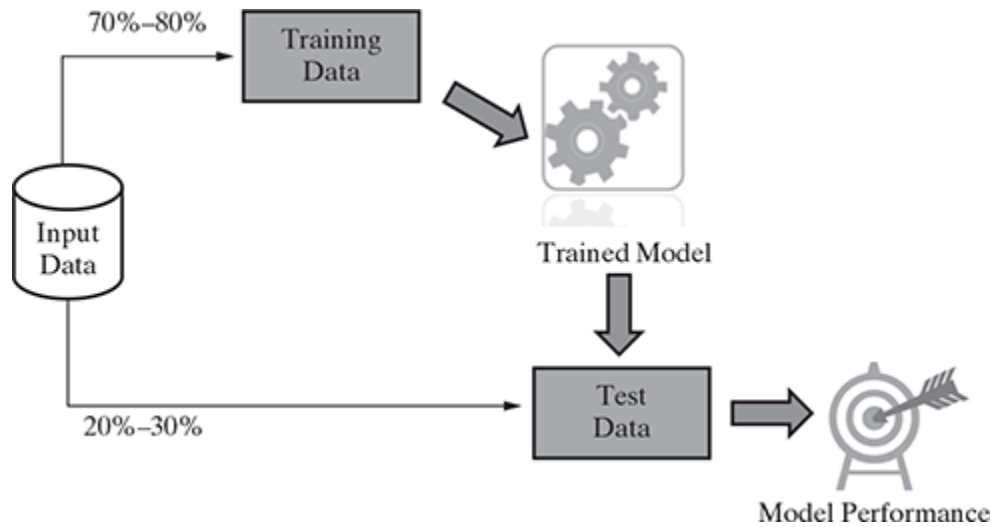
Descriptive models related to pattern discovery is used for market basket analysis of transactional data. In market basket analysis, based on the purchase pattern available in the transactional data, the possibility of purchasing one product based on the purchase of another product is determined. For 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:

1. 10-fold cross-validation (10-fold CV)
2. 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. 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.

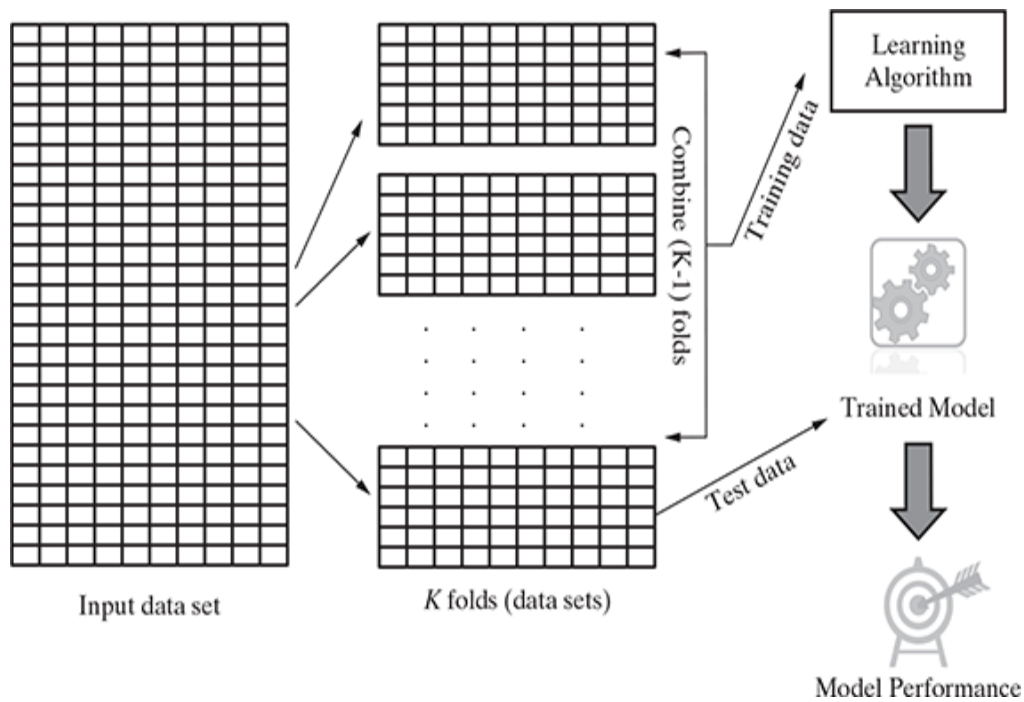


FIG. 3.2 Overall approach for  $K$ -fold cross-validation

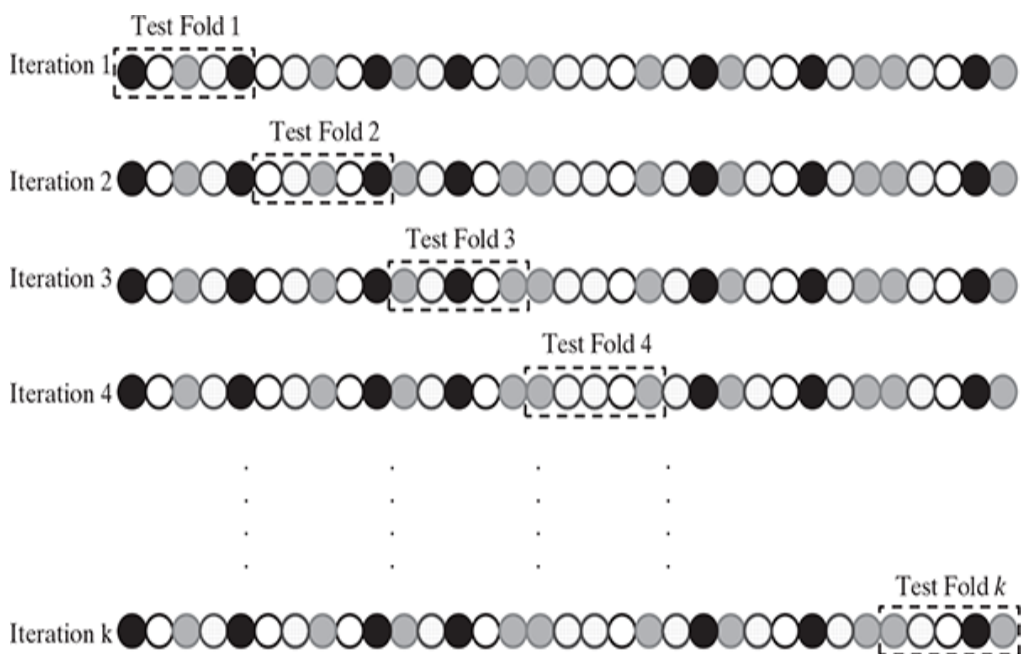


FIG. 3.3 Detailed approach for fold selection

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.

### 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.



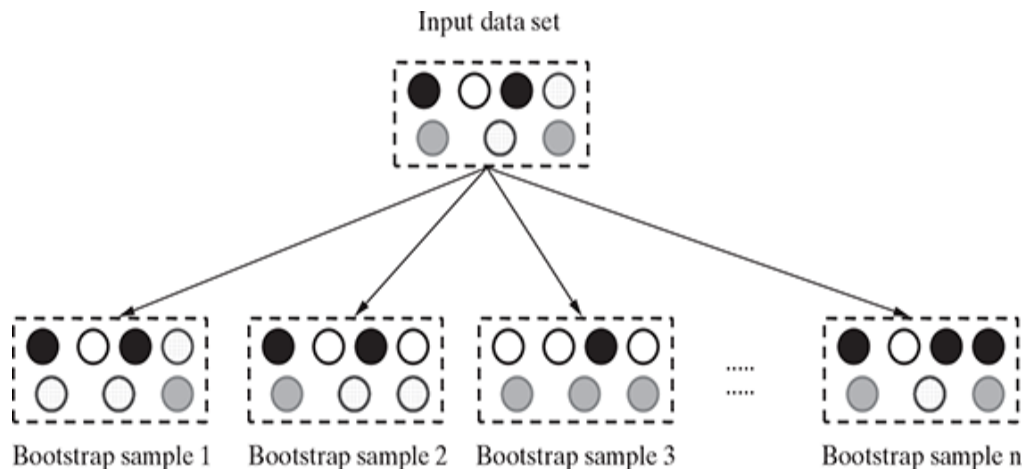


FIG. 3.4 Bootstrap sampling

CROSS-VALIDATION	BOOTSTRAPPING
<p>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 <math>\frac{n}{k}</math> number of unique data elements, where 'n' is the total number of data elements and 'k' is the total number of folds.</p> <p>The number of possible training/test data samples that can be drawn using this technique is finite.</p>	<p>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.</p> <p>In this technique, since elements can be repeated in the sample, possible number of training/test data samples is unlimited.</p>

### 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

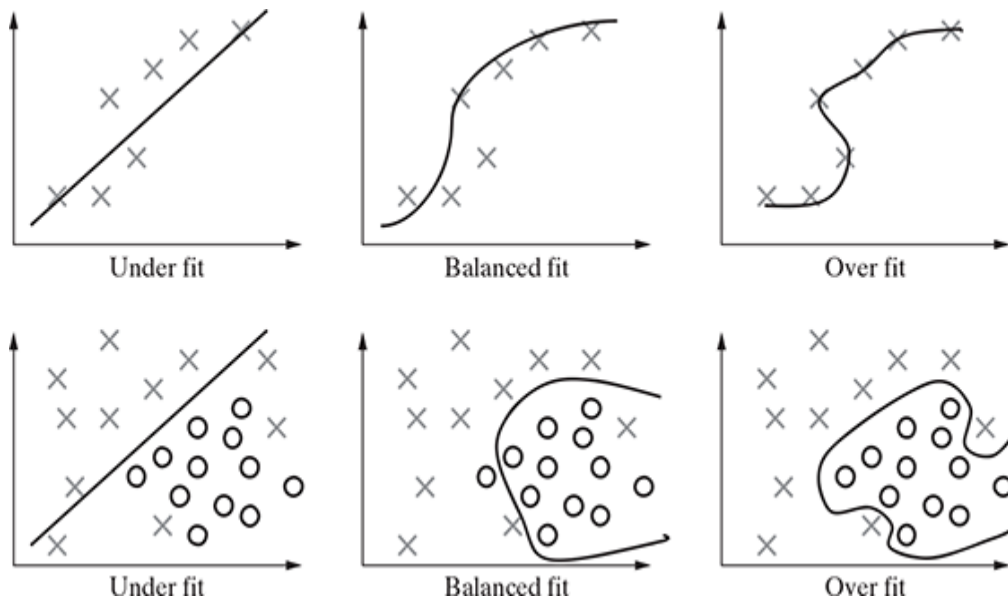
We have already seen that the goal of supervised machine learning is to learn or derive a target function which can best determine the target variable from the set of input variables. A key consideration in learning the target function from the training data is the extent of generalization. This is because 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’

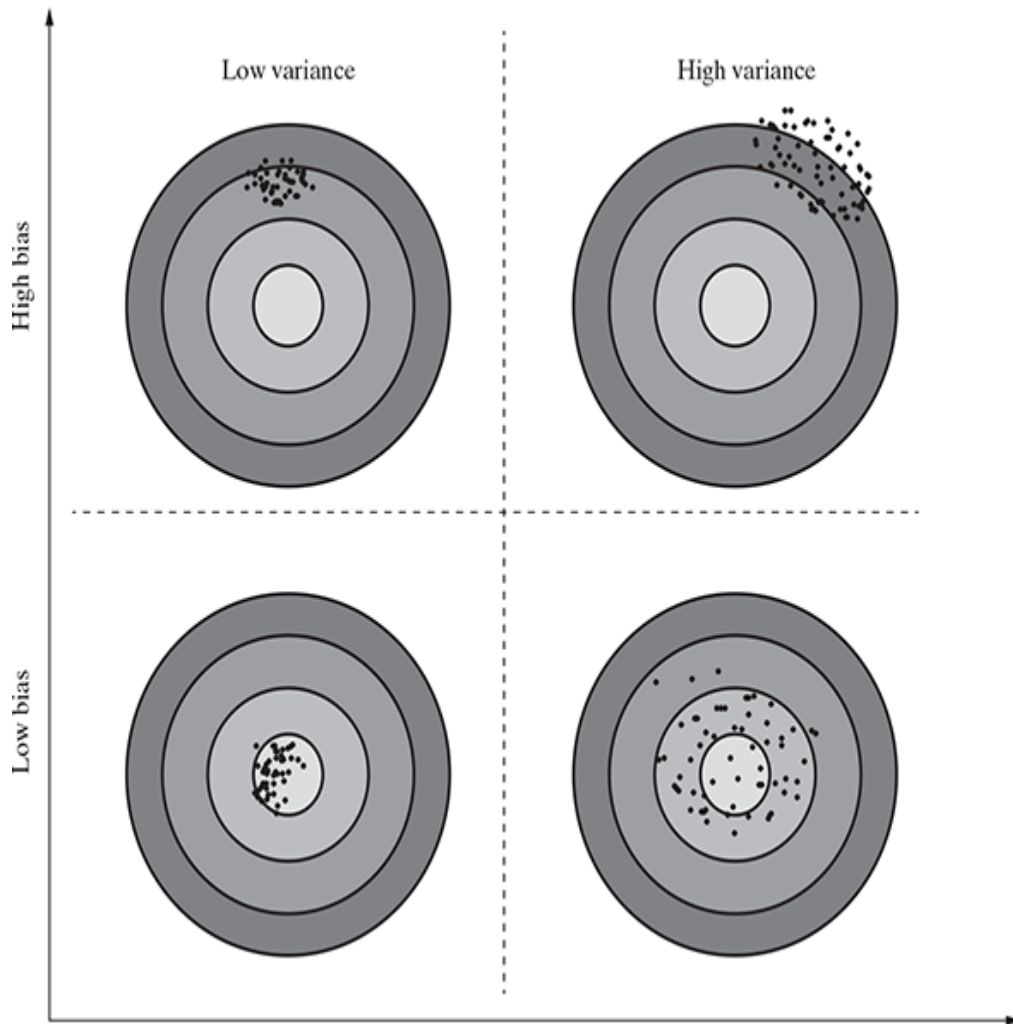
Errors due to bias arise from simplifying assumptions made by the model to make the target function less complex or easier to learn. In short, it is due to underfitting of the model.

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:



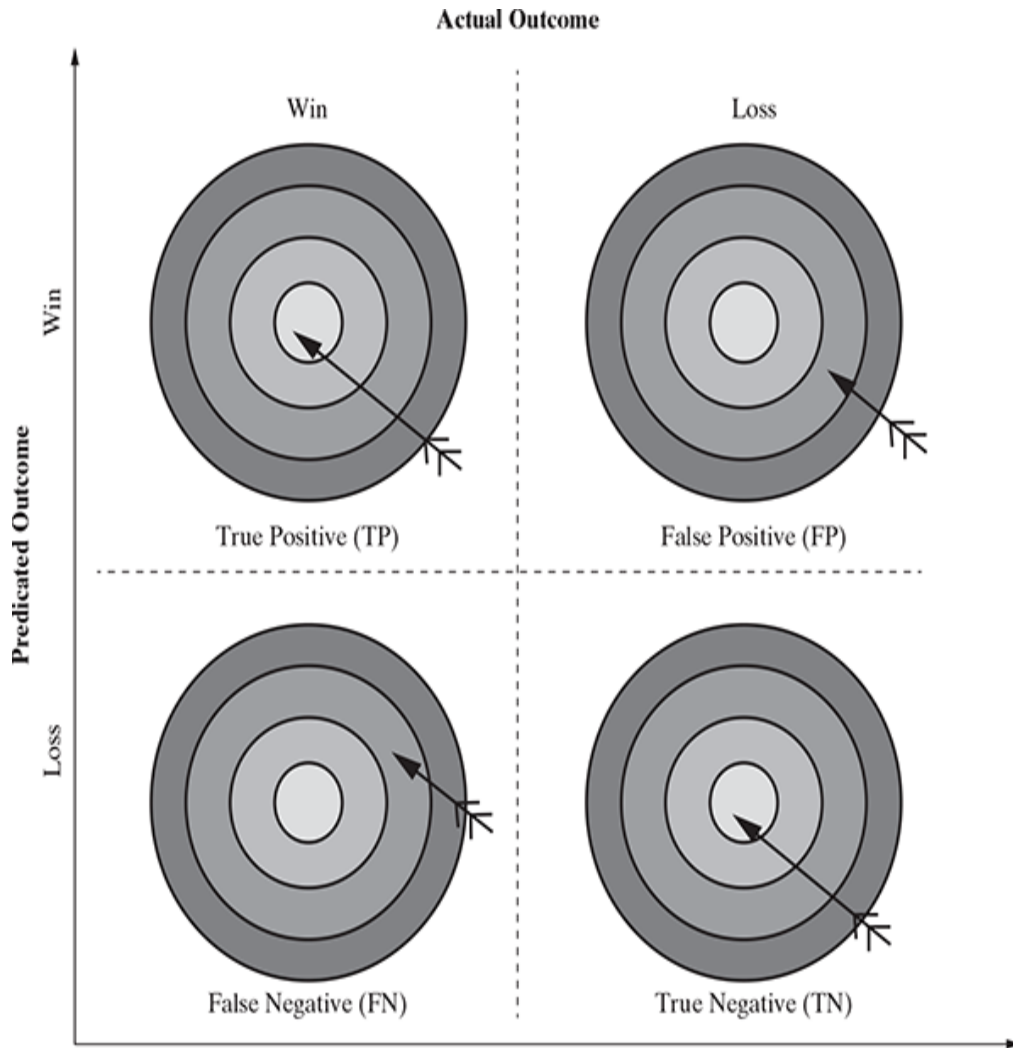
1. the model predicted win and the team won
2. the model predicted win and the team lost
3. the model predicted loss and the team won
4. the model predicted loss and the team lost

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.

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

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  $2 \times 2$  confusion matrix. For a classification problem involving three classes, the confusion matrix would be  $3 \times 3$ , 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.

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

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

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

In context of the above confusion matrix,

$$\begin{aligned}\text{Error rate} &= \frac{\text{FP} + \text{FN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} = \frac{4 + 2}{85 + 4 + 2 + 9} = \frac{6}{100} = 6\% \\ &= 1 - \text{Model accuracy}\end{aligned}$$

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

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

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

$$\begin{aligned}&= \frac{\text{TP} + \text{FP}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \times \frac{\text{TP} + \text{FN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} + \frac{\text{FN} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \\ &\quad \times \frac{\text{FP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}\end{aligned}$$

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.

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

$$\begin{aligned} P(p_r) &= \frac{85 + 4}{85 + 4 + 2 + 9} \times \frac{85 + 2}{85 + 4 + 2 + 9} + \frac{2 + 9}{85 + 4 + 2 + 9} \times \frac{4 + 9}{85 + 4 + 2 + 9} \\ &= \frac{89}{100} \times \frac{87}{100} + \frac{11}{100} \times \frac{13}{100} = 0.89 \times 0.87 + 0.11 \times 0.13 = 0.7886 \end{aligned}$$

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

### 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

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

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

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

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.

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

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

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*

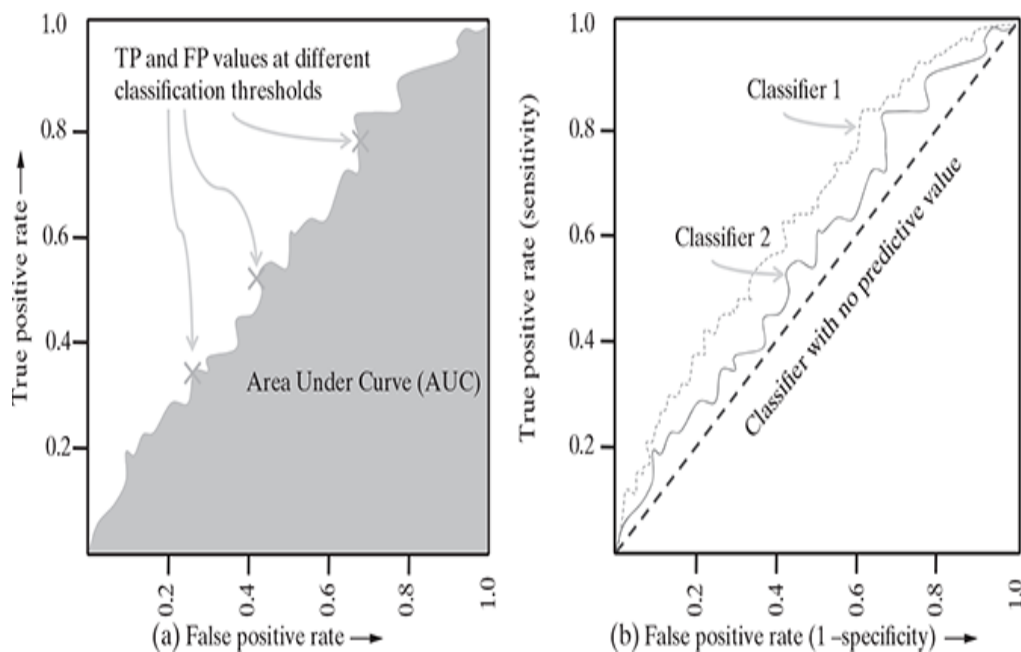
As we have seen till now, though accuracy is the most popular measure, there are quite a number of other measures to evaluate the performance of a supervised learning model. However, visualization is an easier and more effective way to understand the model performance. It also helps in comparing the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

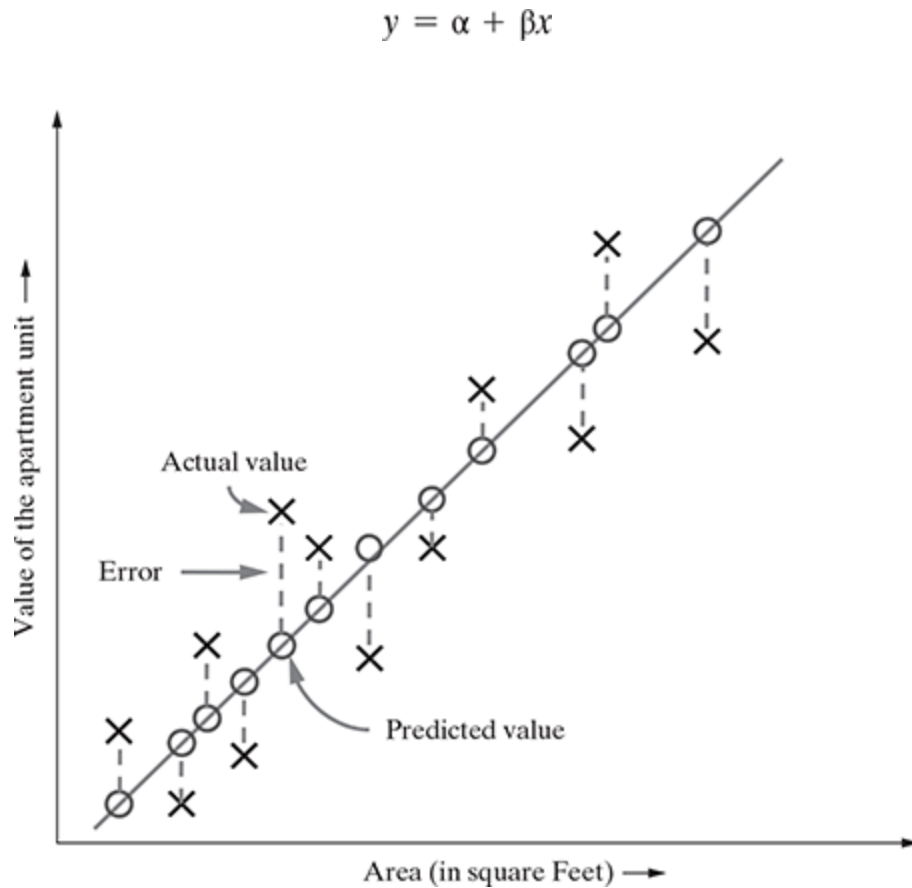


FIG. 3.9 Error – Predicted vs. actual value

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.

#### 1. *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  $i$ -th 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  $a_{i1}, a_{i2}, \dots, a_{in1}$  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,

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

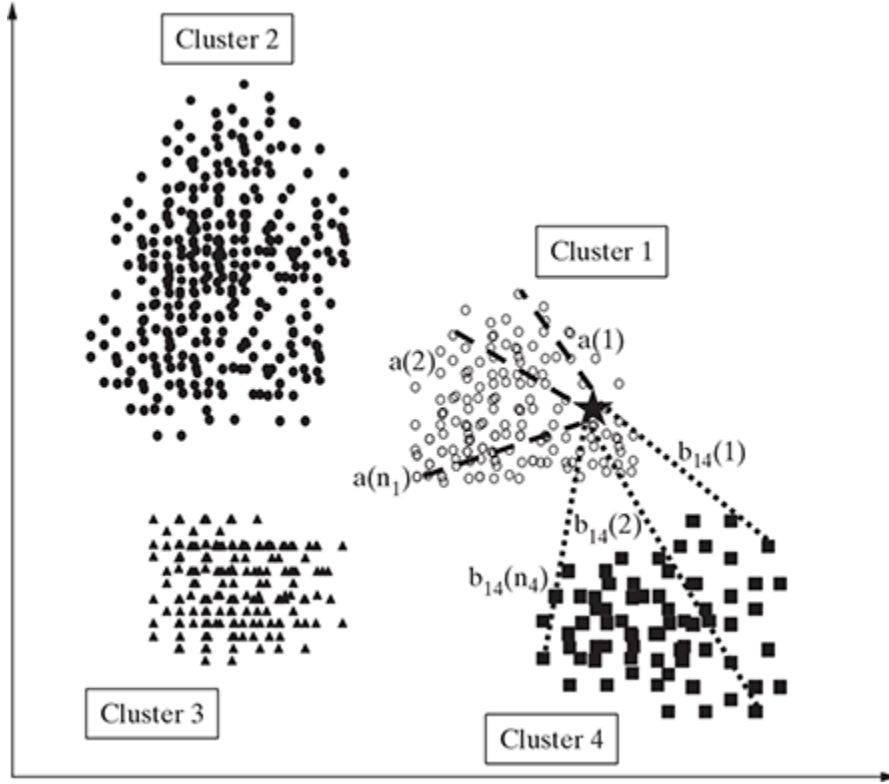


FIG. 3.10 Silhouette width calculation

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,

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

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

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

## 2. 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:

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

### 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 on 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, kNN, 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'.

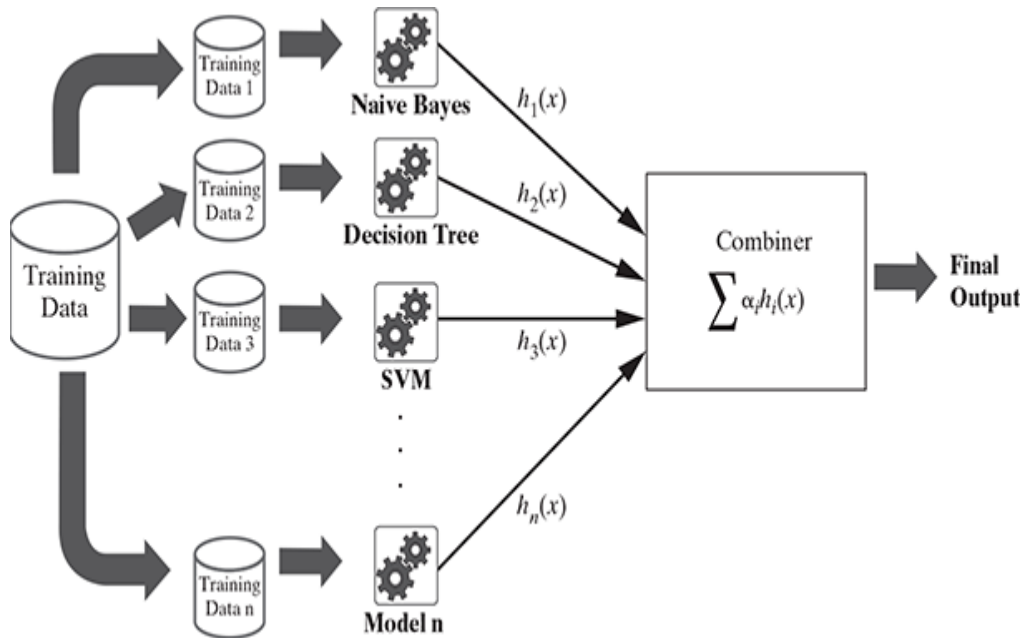


FIG. 3.11 Ensemble

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.
  1. pattern
  2. data
  3. object
  4. none of the above
2. For supervised learning we have \_\_\_\_ model.
  1. interactive
  2. predictive
  3. descriptive
  4. prescriptive
3. For unsupervised learning we have \_\_\_\_ model.
  1. interactive
  2. predictive
  3. descriptive
  4. prescriptive
4. Which of the following measure is not used for a classification model?
  1. Accuracy
  2. Recall
  3. Purity
  4. Error rate
5. Which of the following is a performance measure for regression?
  1. Accuracy
  2. Recall
  3. RMSE
  4. Error rate
6. Which of the following is the measure of cluster quality?

1. Purity
  2. Distance
  3. Accuracy
  4. 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?
1. 10%
  2. 90%
  3. 80%
  4. 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?
1. 10%
  2. 90%
  3. 80%
  4. none of the above
9. There is no one model that works best for every machine learning problem. This is stated as
1. Fit gap model theorem
  2. One model theorem
  3. Free lunch theorem
  4. No free lunch theorem
10. LOOCV in machine learning stands for
1. Love one-out cross validation
  2. Leave-one-out cross-validation
  3. Leave-object oriented cross-validation
  4. Leave-one-out class-validation

#### SHORT-ANSWER TYPE QUESTIONS (5 MARKS QUESTIONS):

1. What is a model in context of machine learning? How can you train a model?
2. Explain “No Free Lunch” theorem in context of machine learning.
3. Explain, in details, the process of  $K$ -fold cross-validation.
4. Explain the bootstrap sampling. Why is it needed?
5. Why do we need to calculate Kappa value for a classification model?  
Show, with a sample set of data, how to calculate Kappa value of a classification model.
6. Explain the process of ensemble of models. What role does it play in machine learning?
7. What is the main purpose of a descriptive model? State some real-world problems solved using descriptive models.
8. Explain the process of evaluating a linear regression model.
9. Differentiate (any two):

1. Predictive vs. descriptive models
  2. Model underfitting vs. overfitting
  3. Cross-validation vs. bootstrapping
10. Write short notes on any two:
1. LOOCV
  2. *F*-measure
  3. Silhouette width
  4. ROC curve

#### LONG-ANSWER TYPE QUESTIONS (10 MARKS QUESTIONS):

1. What is a target function? Express target function in context of a real-life example. How is the fitness of a target function measured?
2. What are predictive models? What are descriptive models? Give examples of both types of models. Explain the difference between these types of models.
3. Explain, in details, the process of evaluating the performance of a classification model. Explain the different parameters of measurement.
4.
  1. What is underfitting in context of machine learning models? What is the major cause of underfitting?
  2. What is overfitting? When does it happen?
  3. Explain bias-variance trade-off in context of model fitting.
5. Can the performance of a learning model be improved? If yes, explain how.
6. How would you evaluate the success of an unsupervised learning model? What are the most popular measures of performance for an unsupervised learning model?
7. Is there a way to use a classification model for a numerical data or a regression model on a categorical data? Explain your answer.
8. Describe the process of predictive modelling for numerical values. How is it different from predictive modelling for categorical values?
9. While predicting malignancy of tumour of a set of patients using a classification model, following are the data recorded:
  1. Correct predictions – 15 malignant, 75 benign
  2. Incorrect predictions – 3 malignant, 7 benign
 Calculate the error rate, Kappa value, sensitivity, precision, and *F*-measure of the model.
10.
  1. Write short notes on any two:
    1. Holdout method
    2. 10-fold cross-validation
    3. Parameter tuning
  2. Write the difference between (any two):
    1. Purity vs. Silhouette width
    2. Bagging vs. Boosting

### 3. Lazy vs. Eager learner



# Chapter 4

## Basics of Feature Engineering

### OBJECTIVE OF THE CHAPTER

In the last three chapters, you have been introduced to the basic concepts of machine learning. Also, the process to start modelling a problem has also been discussed in details. With this context in mind, in this chapter, we will introduce you to another very important aspect of machine learning, that is feature engineering. Though not a core part of the machine learning processes, feature engineering is a critical allied task that we need to perform to make learning more effective. It has three key components – feature construction, feature selection, and feature transformation, each of which will be covered in details in this chapter.

#### 4.1 INTRODUCTION

In the last three chapters, we had a jumpstart to the machine learning process. We first started with what human learning is and how the different types of machine learning emulate the aspects of human learning. We had a detailed view of the

different types of problem that can be solved using machine learning techniques. Before applying machine learning to solve the problems, there are certain preparatory steps. These preparatory steps have been covered in details. After that, we have done a step-by-step navigation of the different activities of modelling a problem using machine learning. Modelling alone doesn't help us to realize the effectiveness of machine learning as a problem-solving tool. So we also learnt how to measure the effectiveness of machine learning models in solving problems. In case a specific model is not effective, we can use different levers to boost the effectiveness. Those levers of boosting the model performance were also covered.

Now that we are ready (well almost ready!) to start solving problems using machine learning, we need to touch upon another key aspect which plays a critical role in solving any machine learning problem – feature engineering. Though feature engineering is a part of the preparatory activities which have already been covered in Chapter 2, the criticality and vastness of the area call for treating it separately. This area deals with features of the data set, which form an important input of any machine learning problem – be supervised or unsupervised learning. Feature engineering is a critical preparatory process in machine learning. It is responsible for taking raw input data and converting that to well-aligned features which are ready to be used by the machine learning models.

But before we start discussing feature engineering, let's try to understand more clearly what feature is.

**Did you know?**

**Unstructured data** is raw, unorganized data which doesn't follow a specific format or hierarchy. Typical examples of unstructured data include text data from social networks, e.g. Twitter, Facebook, etc. or data from server logs, etc.

#### 4.1.1 What is a feature?

A feature is an attribute of a data set that is used in a machine learning process. There is a view amongst certain machine learning practitioners that only those attributes which are meaningful to a machine learning problem are to be called as features, but this view has to be taken with a pinch of salt. In fact, selection of the subset of features which are meaningful for machine learning is a sub-area of feature engineering which draws a lot of research interest. The features in a data set are also called its dimensions. So a data set having ' $n$ ' features is called an  $n$ -dimensional data set.

Let's take the example of a famous machine learning data set, Iris, introduced by the British statistician and biologist Ronald Fisher, partly shown in [Figure 4.1](#). It has five attributes or features namely Sepal.Length, Sepal.Width, Petal.Length, Petal. Width and Species. Out of these, the feature 'Species' represent the class variable and the remaining features are the predictor variables. It is a five-dimensional data set.

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
6.7	3.3	5.7	2.5	Virginica
4.9	3	1.4	0.2	Setosa
5.5	2.6	4.4	1.2	Versicolor
6.8	3.2	5.9	2.3	Virginica
5.5	2.5	4	1.3	Versicolor
5.1	3.5	1.4	0.2	Setosa
6.1	3	4.6	1.4	versicolor

**FIG. 4.1** Data set features

### 4.1.2 What is feature engineering?

Feature engineering refers to the process of translating a data set into features such that these features are able to represent the data set more effectively and result in a better learning performance.

As we know already, feature engineering is an important pre-processing step for machine learning. It has two major elements:

1. feature transformation
2. feature subset selection

**Feature transformation** transforms the data – structured or unstructured, into a new set of features which can represent the underlying problem which machine learning is trying to solve. There are two variants of feature transformation:

1. feature construction
2. feature extraction

Both are sometimes known as feature discovery.

**Feature construction** process discovers missing information about the relationships between features and augments the feature space by creating additional features. Hence, if there are ' $n$ ' features or dimensions in a data set, after feature construction ' $m$ ' more features or dimensions may get added. So at the end, the data set will become ' $n + m$ ' dimensional.

**Feature extraction** is the process of extracting or creating a new set of features from the original set of features using some functional mapping.

Unlike feature transformation, in case of **feature subset selection** (or simply **feature selection**) no new feature is generated. The objective of feature selection is to derive a subset of features from the full feature set which is most meaningful in the context of a specific machine learning problem. So, essentially the job of feature selection is to derive a subset  $F_j (F_1, F_2, \dots, F_m)$  of  $F_i (F_1, F_2, \dots, F_n)$ , where  $m < n$ , such that  $F_j$  is most meaningful and gets the best result for a machine learning problem. We will discuss these concepts in detail in the next section.

### Points to Ponder

Data scientists and machine learning practitioners spend significant amount of time in different feature engineering activities. Selecting the right features has a critical role to play in the success of a machine learning model.

It is quite evident that feature construction expands the feature space, while feature extraction and feature selection reduces the feature space.

## 4.2 FEATURE TRANSFORMATION

Engineering a good feature space is a crucial prerequisite for the success of any machine learning model. However, often it is not clear which feature is more important. For that reason, all available attributes of the data set are used as features and the problem of identifying the important features is left to the learning model. This is definitely not a feasible approach, particularly for certain domains e.g. medical image classification, text categorization, etc. In case a model has to be trained to classify a document as spam or non-spam, we can represent a document as a bag of words. Then the feature space will contain all unique words occurring across all documents. This will easily be a feature space of a few hundred thousand features. If we start including bigrams or trigrams along with words, the count of features will run in millions. To deal with this problem, feature transformation comes into play. Feature transformation is used as an effective tool for dimensionality reduction and hence for boosting learning model performance. Broadly, there are two distinct goals of feature transformation:

- Achieving best reconstruction of the original features in the data set
- Achieving highest efficiency in the learning task

**Did you know?**

In the field of natural language processing, ' $n$ -gram' is a contiguous set of  $n$  items for example words in a text block or document. Using numerical prefixes,  $n$ -gram of size 1 is called unigram (i.e. a single word), size 2 is called bigram (i.e. a two-word phrase), size 3 is called trigram (i.e. a three-word phrase) etc.

#### 4.2.1 Feature construction

Feature construction involves transforming a given set of input features to generate a new set of more powerful features. To understand more clearly, let's take the example of a real estate data set having details of all apartments sold in a specific region.

The data set has three features – apartment length, apartment breadth, and price of the apartment. If it is used as an input to a regression problem, such data can be training data for the regression model. So given the training data, the model should be able to predict the price of an apartment whose price is not known or which has just come up for sale. However, instead of using length and breadth of the apartment as a predictor, it is much convenient and makes more sense to use the area of the apartment, which is not an existing feature of the data set. So such a feature, namely apartment area, can be added to the data set. In other words, we transform the three-dimensional data set to a four-dimensional data set, with the newly 'discovered' feature apartment area being added to the original data set. This is depicted in [Figure 4.2](#).

apartment_ length	apartment_ breadth	apartment_ price		apartment_ length	apartment_ breadth	apartment_ area	apartment_ price
80	59	23,60,000	→	80	59	4,720	23,60,000
54	45	12,15,000		54	45	2,430	12,15,000
78	56	21,84,000		78	56	4,368	21,84,000
63	63	19,84,000		63	63	3,969	19,84,500
83	74	30,71,000		83	74	6,142	30,71,000
92	86	39,56,000		92	86	7,912	39,56,000

**FIG. 4.2** Feature construction (example 1)

**Note:** Though for the sake of simplicity the features apartment length and apartment breadth have been retained in Figure 4.2, in reality, it makes more sense to exclude these features when building the model.

There are certain situations where feature construction is an essential activity before we can start with the machine learning task. These situations are

- when features have categorical value and machine learning needs numeric value inputs
- when features having numeric (continuous) values and need to be converted to ordinal values
- when text-specific feature construction needs to be done

#### 4.2.1.1 Encoding categorical (nominal) variables

Let's take the example of another data set on athletes, as presented in Figure 4.3a. Say the data set has features age, city of origin, parents athlete (i.e. indicate whether any one of the parents was an athlete) and Chance of Win. The feature chance of a win is a class variable while the others are predictor variables. We know that any machine learning algorithm, whether it's a classification algorithm (like  $k$ NN) or a regression algorithm, requires numerical figures to learn from. So there are three features – City of origin, Parents athlete, and



Chance of win, which are categorical in nature and cannot be used by any machine learning task.

In this case, feature construction can be used to create new dummy features which are usable by machine learning algorithms. Since the feature 'City of origin' has three unique values namely City A, City B, and City C, three dummy features namely origin\_city\_A, origin\_city\_B, and origin\_city\_C is created. In the same way, dummy features parents\_athlete\_Y and parents\_athlete\_N are created for feature 'Parents athlete' and win\_chance\_Y and win\_chance\_N are created for feature 'Chance of win'. The dummy features have value 0 or 1 based on the categorical value for the original feature in that row. For example, the second row had a categorical value 'City B' for the feature 'City of origin'. So, the newly created features in place of 'City of origin', i.e. origin\_city\_A, origin\_city\_B and origin\_city\_C will have values 0, 1 and 0, respectively. In the same way, parents\_athlete\_Y and parents\_athlete\_N will have values 0 and 1, respectively in row 2 as the original feature 'Parents athlete' had a categorical value 'No' in row 2. The entire set of transformation for athletes' data set is shown in Figure 4.3b.

Age (Years)	City of origin	Parents athlete	Chance of win
18	City A	Yes	Y
20	City B	No	Y
23	City B	Yes	Y
19	City A	No	N
18	City C	Yes	N
22	City B	Yes	Y

(a)

Age (Years)	origin_city_A	origin_city_B	origin_city_C	parents_athlete_Y	parents_athlete_N	win_chance_Y	win_chance_N
18	1	0	0	1	0	1	0
20	0	1	0	0	1	1	0
23	0	1	0	1	0	1	0
19	1	0	0	0	1	0	1
18	0	0	1	1	0	0	1
22	0	1	0	1	0	1	0

(b)

Age (Years)	origin_city_A	origin_city_B	origin_city_C	parents_athlete_Y	win_chance_Y
18	1	0	0	1	1
20	0	1	0	0	1
23	0	1	0	1	1
19	1	0	0	0	0
18	0	0	1	1	0
22	0	1	0	1	1

(c)

FIG. 4.3 Feature construction (encoding nominal variables)

However, examining closely, we see that the features ‘Parents athlete’ and ‘Chance of win’ in the original data set can have two values only. So creating two features from them is a kind of duplication, since the value of one feature can be decided from the value of the other. To avoid this duplication, we can just leave one feature and eliminate the other, as shown in Figure 4.3c.

#### 4.2.1.2 Encoding categorical (ordinal) variables

Let's take an example of a student data set. Let's assume that there are three variable – science marks, maths marks and grade as shown in Figure 4.4a. As we can see, the grade is an ordinal variable with values A, B, C, and D. To transform this variable to a numeric variable, we can create a feature `num_grade` mapping a numeric value against each ordinal value. In the context of the current example, grades A, B, C, and D in Figure 4.4a is mapped to values 1, 2, 3, and 4 in the transformed variable shown in Figure 4.4b.

marks_science	marks_maths	Grade
78	75	B
56	62	C
87	90	A
91	95	A
45	42	D
62	57	B

(a)

marks_science	marks_maths	num_grade
78	75	2
56	62	3
87	90	1
91	95	1
45	42	4
62	57	2

(b)

FIG. 4.4 Feature construction (encoding ordinal variables)

#### 4.2.1.3 Transforming numeric (continuous) features to categorical features

Sometimes there is a need of transforming a continuous numerical variable into a categorical variable. For example, we may want to treat the real estate price prediction problem, which is a regression problem, as a real estate price category prediction, which is a classification problem. In that case, we can 'bin' the numerical data into multiple categories based on the data range. In the context of the real estate price prediction example, the original data set has a numerical feature `apartment_price` as shown in Figure 4.5a. It can be

transformed to a categorical variable price-grade either as shown in [Figure 4.5b](#) or as shown in [Figure 4.5c](#).

#### *4.2.1.4 Text-specific feature construction*

In the current world, text is arguably the most predominant medium of communication. Whether we think about social networks like Facebook or micro-blogging channels like Twitter or emails or short messaging services such as Whatsapp, text plays a major role in the flow of information. Hence, text mining is an important area of research – not only for technology practitioners but also for industry practitioners. However, making sense of text data, due to the inherent unstructured nature of the data, is not so straightforward. In the first place, the text data chunks that we can think about do not have readily available features, like structured data sets, on which machine learning tasks can be executed. All machine learning models need numerical data as input. So the text data in the data sets need to be transformed into numerical features.

apartment_area	apartment_price	apartment_area	apartment_grade
4,720	23,60,000	4,720	Medium
2,430	12,15,000	2,430	Low
4,368	21,84,000	4,368	Medium
3,969	19,84,500	3,969	Low
6,142	30,71,000	6,142	High
7,912	39,56,000	7,912	High

(a) (b)

apartment_area	apartment_grade
4,720	2
2,430	1
4,368	2
3,969	1
6,142	3
7,912	3

(c)

**FIG. 4.5** Feature construction (numeric to categorical)

Text data, or corpus which is the more popular keyword, is converted to a numerical representation following a process is known as vectorization. In this process, word occurrences in all documents belonging to the corpus are consolidated in the form of bag-of-words. There are three major steps that are followed:

1. tokenize
2. count
3. normalize

In order to tokenize a corpus, the blank spaces and punctuations are used as delimiters to separate out the words, or tokens. Then the number of occurrences of each token is counted, for each document. Lastly, tokens are weighted with

reducing importance when they occur in the majority of the documents. A matrix is then formed with each token representing a column and a specific document of the corpus representing each row. Each cell contains the count of occurrence of the token in a specific document. This matrix is known as a document-term matrix (also known as a term-document matrix). Figure 4.6 represents a typical document-term matrix which forms an input to a machine learning model.

<b>This</b>	<b>House</b>	<b>Build</b>	<b>Feeling</b>	<b>Well</b>	<b>Theatre</b>	<b>Movie</b>	<b>Good</b>	<b>Lonely</b>	<b>...</b>
2	1	1	0	0	1	1	1	0	
0	0	0	1	1	0	0	0	0	
1	0	0	2	1	1	0	0	1	
0	0	0	0	1	0	1	1	0	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	
.	.	.	.	.	.	.	.	.	

**FIG. 4.6** Feature construction (text-specific)

## 4.2.2 Feature extraction

In feature extraction, new features are created from a combination of original features. Some of the commonly used operators for combining the original features include

1. For Boolean features: Conjunctions, Disjunctions, Negation, etc.
2. For nominal features: Cartesian product, M of N, etc.
3. For numerical features: Min, Max, Addition, Subtraction, Multiplication, Division, Average, Equivalence, Inequality, etc.

Let's take an example and try to understand. Say, we have a data set with a feature set  $F_i$  ( $F_1, F_2, \dots, F_n$ ). After feature extraction using a mapping function  $f$  ( $F_1, F_2, \dots, F_n$ ) say, we

will have a set of features  $\hat{F}_i(\hat{F}_1, \hat{F}_2, \dots, \hat{F}_m)$  such that  $\hat{F}_i = f(F_i)$

and  $m < n$ . For example,  $\hat{F}_1 = k_1 F_1 + k_2 F_2$ . This is depicted in

Figure 4.7.

Feat <sub>A</sub>	Feat <sub>B</sub>	Feat <sub>C</sub>	Feat <sub>D</sub>		Feat <sub>1</sub>	Feat <sub>2</sub>
34	34.5	23	233		41.25	185.80
44	45.56	11	3.44		54.20	53.12
78	22.59	21	4.5	→	43.73	35.79
22	65.22	11	322.3		65.30	264.10
22	33.8	355	45.2		37.02	238.42
11	122.32	63	23.2		113.39	167.74

Feat <sub>1</sub> = 0.3 × Feat <sub>A</sub> + 0.9 × Feat <sub>A</sub>	
Feat <sub>2</sub> = Feat <sub>A</sub> + 0.5 Feat <sub>B</sub> + 0.6 × Feat <sub>C</sub>	

FIG. 4.7 Feature extraction

Let's discuss the most popular feature extraction algorithms used in machine learning:

#### 4.2.2.1 Principal Component Analysis

Every data set, as we have seen, has multiple attributes or dimensions – many of which might have similarity with each other. For example, the height and weight of a person, in general, are quite related. If the height is more, generally weight is more and vice versa. So if a data set has height and weight as two of the attributes, obviously they are expected to be having quite a bit of similarity. In general, any machine learning algorithm performs better as the number of related attributes or features reduced. In other words, a key to the

success of machine learning lies in the fact that the features are less in number as well as the similarity between each other is very less. This is the main guiding philosophy of principal component analysis (PCA) technique of feature extraction.

In PCA, a new set of features are extracted from the original features which are quite dissimilar in nature. So an  $n$ -dimensional feature space gets transformed to an  $m$ -dimensional feature space, where the dimensions are orthogonal to each other, i.e. completely independent of each other. To understand the concept of orthogonality, we have to step back and do a bit of dip dive into vector space concept in linear algebra.

We all know that a vector is a quantity having both magnitude and direction and hence can determine the position of a point relative to another point in the Euclidean space (i.e. a two or three or 'n' dimensional space). A vector space is a set of vectors. Vector spaces have a property that they can be represented as a linear combination of a smaller set of vectors, called basis vectors. So, any vector 'v' in a vector space can be represented as

$$v = \sum_{i=1}^n a_i u_i$$

where,  $a_i$  represents 'n' scalars and  $u_i$  represents the basis vectors. Basis vectors are orthogonal to each other. Orthogonality of vectors in  $n$ -dimensional vector space can be thought of an extension of the vectors being perpendicular in a two-dimensional vector space. Two orthogonal vectors are completely unrelated or independent of each other. So the transformation of a set of vectors to the corresponding set of



basis vectors such that each vector in the original set can be expressed as a linear combination of basis vectors helps in decomposing the vectors to a number of independent components.

Now, let's extend this notion to the feature space of a data set. The feature vector can be transformed to a vector space of the basis vectors which are termed as principal components. These principal components, just like the basis vectors, are orthogonal to each other. So a set of feature vectors which may have similarity with each other is transformed to a set of principal components which are completely unrelated. However, the principal components capture the variability of the original feature space. Also, the number of principal component derived, much like the basis vectors, is much smaller than the original set of features.

The objective of PCA is to make the transformation in such a way that

1. The new features are distinct, i.e. the covariance between the new features, i.e. the principal components is 0.
2. The principal components are generated in order of the variability in the data that it captures. Hence, the first principal component should capture the maximum variability, the second principal component should capture the next highest variability etc.
3. The sum of variance of the new features or the principal components should be equal to the sum of variance of the original features.

PCA works based on a process called eigenvalue decomposition of a covariance matrix of a data set. Below are the steps to be followed:

1. First, calculate the covariance matrix of a data set.
2. Then, calculate the eigenvalues of the covariance matrix.
3. The eigenvector having highest eigenvalue represents the direction in which there is the highest variance. So this will help in identifying the

first principal component.

4. The eigenvector having the next highest eigenvalue represents the direction in which data has the highest remaining variance and also orthogonal to the first direction. So this helps in identifying the second principal component.
5. Like this, identify the top ' $k$ ' eigenvectors having top ' $k$ ' eigenvalues so as to get the ' $k$ ' principal components.

#### 4.2.2.2 Singular value decomposition

Singular value decomposition (SVD) is a matrix factorization technique commonly used in linear algebra. SVD of a matrix  $A$  ( $m \times n$ ) is a factorization of the form:

$$A = U \Sigma V$$

where,  $U$  and  $V$  are orthonormal matrices,  $U$  is an  $m \times m$  unitary matrix,  $V$  is an  $n \times n$  unitary matrix and  $\Sigma$  is an  $m \times n$  rectangular diagonal matrix. The diagonal entries of  $\Sigma$  are known as singular values of matrix  $A$ . The columns of  $U$  and  $V$  are called the left-singular and right-singular vectors of matrix  $A$ , respectively.

SVD is generally used in PCA, once the mean of each variable has been removed. Since it is not always advisable to remove the mean of a data attribute, especially when the data set is sparse (as in case of text data), SVD is a good choice for dimensionality reduction in those situations.

SVD of a data matrix is expected to have the properties highlighted below:

1. Patterns in the attributes are captured by the right-singular vectors, i.e. the columns of  $V$ .
2. Patterns among the instances are captured by the left-singular, i.e. the columns of  $U$ .

3. Larger a singular value, larger is the part of the matrix  $A$  that it accounts for and its associated vectors.
4. New data matrix with ' $k$ ' attributes is obtained using the equation

$$D' = D \times [v_1, v_2, \dots, v_k]$$

Thus, the dimensionality gets reduced to  $k$   
SVD is often used in the context of text data.

#### 4.2.2.3 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is another commonly used feature extraction technique like PCA or SVD. The objective of LDA is similar to the sense that it intends to transform a data set into a lower dimensional feature space. However, unlike PCA, the focus of LDA is not to capture the data set variability. Instead, LDA focuses on class separability, i.e. separating the features based on class separability so as to avoid over-fitting of the machine learning model.

Unlike PCA that calculates eigenvalues of the covariance matrix of the data set, LDA calculates eigenvalues and eigenvectors within a class and inter-class scatter matrices. Below are the steps to be followed:

1. Calculate the mean vectors for the individual classes.
2. Calculate intra-class and inter-class scatter matrices.
3. Calculate eigenvalues and eigenvectors for  $S_W^{-1}$  and  $S_B$ , where  $S_W$  is the intra-class scatter matrix and  $S_B$  is the inter-class scatter matrix

$$S_W = \sum_{i=1}^c S_i;$$

$$S_i = \sum_{x \in D_i}^n (x - m_i)(x - m_i)^T$$

where,  $m_i$  is the mean vector of the  $i$ -th class

$$S_B = \sum_{i=1}^c N_i (m_i - m) (m_i - m)^T$$

where,  $m_i$  is the sample mean for each class,  $m$  is the overall mean of the data set,  $N_i$  is the sample size of each class

4. Identify the top ' $k$ ' eigenvectors having top ' $k$ ' eigenvalues

### 4.3 FEATURE SUBSET SELECTION

Feature selection is arguably the most critical pre-processing activity in any machine learning project. It intends to select a subset of system attributes or features which makes a most meaningful contribution in a machine learning activity. Let's quickly discuss a practical example to understand the philosophy behind feature selection. Say we are trying to predict the weight of students based on past information about similar students, which is captured in a 'student weight' data set. The student weight data set has features such as Roll Number, Age, Height, and Weight. We can well understand that roll number can have no bearing, whatsoever, in predicting student weight. So we can eliminate the feature roll number and build a feature subset to be considered in this machine learning problem. The subset of features is expected to give better results than the full set. The same has been depicted in Figure 4.8.

Roll Number	Age	Height	Weight		Age	Height	Weight
12	12	1.1	23	→	12	1.1	23
14	11	1.05	21.6		11	1.05	21.6
19	13	1.2	24.7		13	1.2	24.7
32	11	1.07	21.3		11	1.07	21.3
38	14	1.24	25.2		14	1.24	25.2
45	12	1.12	23.4		12	1.12	23.4

FIG. 4.8 Feature selection

But before we go forward with more detailed discussion on feature selection, let's try to understand the issues which have

made feature selection such a relevant problem to be solved.

#### 4.3.1 Issues in high-dimensional data

With the rapid innovations in the digital space, the volume of data generated has increased to an unbelievable extent. At the same time, breakthroughs in the storage technology area have made storage of large quantity of data quite cheap. This has further motivated the storage and mining of very large and high-dimensionality data sets.

##### **Points to Ponder**

‘High-dimensional’ refers to the high number of variables or attributes or features present in certain data sets, more so in the domains like DNA analysis, geographic information systems (GIS), social networking, etc. The high-dimensional spaces often have hundreds or thousands of dimensions or attributes, e.g. DNA microarray data can have up to 450,000 variables (gene probes).

Alongside, two new application domains have seen drastic development. One is that of biomedical research, which includes gene selection from microarray data. The other one is text categorization which deals with huge volumes of text data from social networking sites, emails, etc. The first domain, i.e. biomedical research generates data sets having a number of features in the range of a few tens of thousands. The text data generated from different sources also have extremely high dimensions. In a large document corpus having few thousand documents embedded, the number of unique word tokens

which represent the feature of the text data set, can also be in the range of a few tens of thousands. To get insight from such high-dimensional data may be a big challenge for any machine learning algorithm. On one hand, very high quantity of computational resources and high amount of time will be required. On the other hand the performance of the model – both for supervised and unsupervised machine learning task, also degrades sharply due to unnecessary noise in the data. Also, a model built on an extremely high number of features may be very difficult to understand. For this reason, it is necessary to take a subset of the features instead of the full set.

The objective of feature selection is three-fold:

- Having faster and more cost-effective (i.e. less need for computational resources) learning model
- Improving the efficiency of the learning model
- Having a better understanding of the underlying model that generated the data

#### 4.3.2 Key drivers of feature selection – feature relevance and redundancy

##### *4.3.2.1 Feature relevance*

In supervised learning, the input data set which is the training data set, has a class label attached. A model is induced based on the training data set – so that the induced model can assign class labels to new, unlabelled data. Each of the predictor variables, is expected to contribute information to decide the value of the class label. In case a variable is not contributing any information, it is said to be irrelevant. In case the information contribution for prediction is very little, the variable is said to be weakly relevant. Remaining variables, which make a significant contribution to the prediction task are said to be strongly relevant variables.

In unsupervised learning, there is no training data set or labelled data. Grouping of similar data instances are done and similarity of data instances are evaluated based on the value of different variables. Certain variables do not contribute any useful information for deciding the similarity or dissimilarity of data instances. Hence, those variables make no significant information contribution in the grouping process. These variables are marked as irrelevant variables in the context of the unsupervised machine learning task.

To get a perspective, we can think of the simple example of the student data set that we discussed at the beginning of this section. Roll number of a student doesn't contribute any significant information in predicting what the Weight of a student would be. Similarly, if we are trying to group together students with similar academic capabilities, Roll number can really not contribute any information whatsoever. So, in context of the supervised task of predicting student Weight or the unsupervised task of grouping students with similar academic merit, the variable Roll number is quite irrelevant.

Any feature which is irrelevant in the context of a machine learning task is a candidate for rejection when we are selecting a subset of features. We can consider whether the weakly relevant features are to be rejected or not on a case-to-case basis.

#### *4.3.2.2 Feature redundancy*

A feature may contribute information which is similar to the information contributed by one or more other features. For example, in the weight prediction problem referred earlier in the section, both the features Age and Height contribute similar information. This is because with an increase in Age,

Weight is expected to increase. Similarly, with the increase of Height also Weight is expected to increase. Also, Age and Height increase with each other. So, in context of the Weight prediction problem, Age and Height contribute similar information. In other words, irrespective of whether the feature height is present as a part of the feature subset, the learning model will give almost same results. In the same way, without age being part of the predictor variables, the outcome of the learning model will be more or less same. In this kind of a situation when one feature is similar to another feature, the feature is said to be potentially redundant in the context of the learning problem.

All features having potential redundancy are candidates for rejection in the final feature subset. Only a small number of representative features out of a set of potentially redundant features are considered for being a part of the final feature subset.

So, in a nutshell, the main objective of feature selection is to remove all features which are irrelevant and take a representative subset of the features which are potentially redundant. This leads to a meaningful feature subset in context of a specific learning task.

Now, the question is how to find out which of the features are irrelevant or which features have potential redundancy. For that multiple measures are being used, some of which have been covered in the next sub-section.

### 4.3.3 Measures of feature relevance and redundancy

#### *4.3.3.1 Measures of feature relevance*



As mentioned earlier, feature relevance is to be gauged by the amount of information contributed by a feature. For supervised learning, mutual information is considered as a good measure of information contribution of a feature to decide the value of the class label. That's why it is a good indicator of the relevance of a feature with respect to the class variable. Higher the value of mutual information of a feature, more relevant is that feature. Mutual information can be calculated as follows:

$$MI(C, f) = H(C) + H(f) - H(C, f)$$

where, marginal entropy of the class,  $H(C) =$

$$-\sum_{i=1}^k p(C_i) \log_2 p(C_i)$$

marginal entropy of the feature 'x',  $H(f) =$

$$-\sum_c p(f = x) \log_2 p(f = x)$$

and  $K$  = number of classes,  $C$  = class variable,  $f$  = feature set that take discrete values.

In case of unsupervised learning, there is no class variable. Hence, feature-to-class mutual information cannot be used to measure the information contribution of the features. In case of unsupervised learning, the entropy of the set of features without one feature at a time is calculated for all the features. Then, the features are ranked in a descending order of

information gain from a feature and top ‘ $\beta$ ’ percentage (value of ‘ $\beta$ ’ is a design parameter of the algorithm) of features are selected as relevant features. The entropy of a feature  $f$  is calculated using Shannon’s formula below:

$$H(f) = - \sum_x p(f = x) \log_2 p(f = x)$$

$\sum_x$  is used only for features that take discrete values. For

continuous features, it should be replaced by discretization performed first to estimate probabilities  $p(f = x)$ .

#### *4.3.3.2 Measures of Feature redundancy*

Feature redundancy, as we have already discussed, is based on similar information contribution by multiple features. There are multiple measures of similarity of information contribution, salient ones being

1. Correlation-based measures
2. Distance-based measures, and
3. Other coefficient-based measure

##### *1. Correlation-based similarity measure*

Correlation is a measure of linear dependency between two random variables. Pearson’s product moment correlation coefficient is one of the most popular and accepted measures of correlation between two random variables. For two random feature variables  $F_1$  and  $F_2$ , Pearson correlation coefficient is defined as:

$$\alpha = \frac{cov(F_1, F_2)}{\sqrt{var(F_1).var(F_2)}}$$

$$cov(F_1, F_2) = \sum (F_{1_i} - \bar{F}_1).(F_{2_i} - \bar{F}_2)$$

$$var(F_1) = \sum (F_{1_i} - \bar{F}_1)^2, \text{ where } \bar{F}_1 = \frac{1}{n} \cdot \sum F_{1_i}$$

$$var(F_2) = \sum (F_{2_i} - \bar{F}_2)^2, \text{ where } \bar{F}_2 = \frac{1}{n} \cdot \sum F_{2_i}$$

Correlation values range between +1 and -1. A correlation of 1 (+ / -) indicates perfect correlation, i.e. the two features having a perfect linear relationship. In case the correlation is 0, then the features seem to have no linear relationship. Generally, for all feature selection problems, a threshold value is adopted to decide whether two features have adequate similarity or not.

## 2. Distance-based similarity measure

The most common distance measure is the **Euclidean distance**, which, between two features  $F_1$  and  $F_2$  are calculated as:

$$d(F_1, F_2) = \sqrt{\sum_{i=1}^n (F_{1_i} - F_{2_i})^2}$$

where  $F_1$  and  $F_2$  are features of an  $n$ -dimensional data set. Refer to the Figure 4.9. The data set has two features, aptitude ( $F_1$ ) and communication ( $F_2$ ) under consideration. The Euclidean distance between the features has been calculated using the formula provided above.

Aptitude ( $F_1$ )	Communication ( $F_2$ )	$(F_1 - F_2)$	$(F_1 - F_2)^2$
2	6	-4	16
3	5.5	-2.5	6.25
6	4	2	4
7	2.5	4.5	20.25
8	3	5	25
6	5.5	0.5	0.25
6	7	-1	1
7	6	1	1
8	6	2	4
9	7	2	4
			81.75

FIG. 4.9 Distance calculation between features

A more generalized form of the Euclidean distance is the **Minkowski distance**, measured as

$$d(F_1, F_2) = \sqrt[r]{\sum_{i=1}^n (F_{1_i} - F_{2_i})^r}$$

Minkowski distance takes the form of Euclidean distance (also called  **$L_2$  norm**) when  $r = 2$ .

At  $r = 1$ , it takes the form of **Manhattan distance** (also called  **$L_1$  norm**), as shown below:

$$d(F_1, F_2) = \sum_{i=1}^n |F_{1_i} - F_{2_i}|$$

A specific example of Manhattan distance, used more frequently to calculate the distance between binary vectors is the **Hamming distance**. For example, the Hamming distance

between two vectors 01101011 and 11001001 is 3, as illustrated in Figure 4.10a.

### 3. Other similarity measures

**Jaccard index/coefficient** is used as a measure of similarity between two features. The **Jaccard distance**, a measure of dissimilarity between two features, is complementary of Jaccard index.

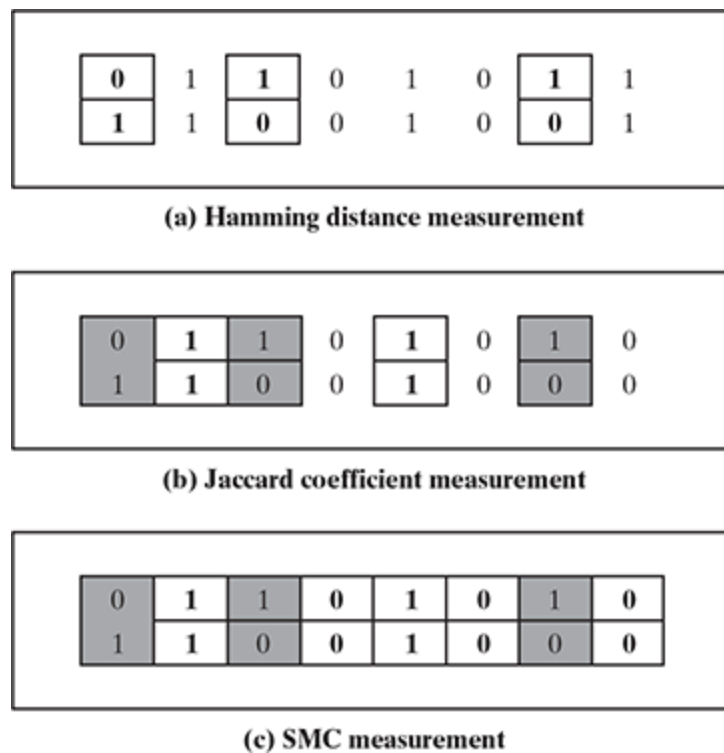


FIG. 4.10 Distance measures between features

For two features having binary values, Jaccard index is measured as

$$J = \frac{n_{11}}{n_{01} + n_{10} + n_{11}}$$

where,  $n_{11}$  = number of cases where both the features have value 1

$n_{01}$  = number of cases where the feature 1 has value 0 and feature 2 has value 1

$n_{10}$  = number of cases where the feature 1 has value 1 and feature 2 has value 0

Jaccard distance,  $d_J = 1 - J$

Let's consider two features  $F_1$  and  $F_2$  having values (0, 1, 1, 0, 1, 0, 1, 0) and (1, 1, 0, 0, 1, 0, 0, 0). Figure 4.10b shows the identification of the values of  $n_{11}$ ,  $n_{01}$  and  $n_{10}$ . As shown, the cases where both the values are 0 have been left out without border – as an indication of the fact that they will be excluded in the calculation of Jaccard coefficient.

Jaccard coefficient of  $F_1$  and  $F_2$ ,  $J =$

$$\frac{n_{11}}{n_{01} + n_{10} + n_{11}} = \frac{2}{1 + 2 + 2} = \frac{2}{5} \text{ or } 0.4.$$

$\therefore$  Jaccard distance between  $F_1$  and  $F_2$ ,  $d_J = 1 - J = \frac{1}{2}$  or 0.6.

**Simple matching coefficient (SMC)** is almost same as Jaccard coefficient except the fact that it includes a number of cases where both the features have a value of 0.

$$SMC = \frac{n_{11} + n_{00}}{n_{00} + n_{01} + n_{10} + n_{11}}$$

where,  $n_{11}$  = number of cases where both the features have value 1

$n_{01}$  = number of cases where the feature 1 has value 0 and feature 2 has value 1

$n_{10}$  = number of cases where the feature 1 has value 1 and feature 2 has value 0

$n_{00}$  = number of cases where both the features have value 0

Quite understandably, the total count of rows,  $n = n_{00} + n_{01} + n_{10} + n_{11}$ . As shown in [Figure 4.10c](#), all values have been included in the calculation of SMC.

$$\therefore \text{SMC of } F_1 \text{ and } F_2 = \frac{n_{11} + n_{00}}{n_{00} + n_{01} + n_{10} + n_{11}} = \frac{2 + 3}{3 + 1 + 2 + 2} = \frac{1}{2} \text{ or } 0.5.$$

One more measure of similarity using similarity coefficient calculation is **Cosine Similarity**. Let's take the example of a typical text classification problem. The text corpus needs to be first transformed into features with a word token being a feature and the number of times the word occurs in a document comes as a value in each row. There are thousands of features in such a text data set. However, the data set is sparse in nature as only a few words do appear in a document, and hence in a row of the data set. So each row has very few non-zero values. However, the non-zero values can be anything integer value as the same word may occur any number of times. Also, considering the sparsity of the data set, the 0-0 matches (which obviously is going to be pretty high) need to be ignored. Cosine similarity which is one of the most popular measures in text classification is calculated as:

$$\cos(x, y) = \frac{x \cdot y}{\|x\| \cdot \|y\|}$$

where,  $x \cdot y$  = vector dot product of  $x$  and  $y = \sum_{i=1}^n x_i y_i$

$$\|x\| = \sqrt{\sum_{i=1}^n x_i^2} \text{ and } \|y\| = \sqrt{\sum_{i=1}^n y_i^2}$$

Let's calculate the cosine similarity of  $x$  and  $y$ , where  $x = (2, 4, 0, 0, 2, 1, 3, 0, 0)$  and  $y = (2, 1, 0, 0, 3, 2, 1, 0, 1)$ .

In this case,  $x \cdot y = 2*2 + 4*1 + 0*0 + 0*0 + 2*3 + 1*2 + 3*1 + 0*0 + 0*1 = 19$

$$\|x\| = \sqrt{2^2 + 4^2 + 0^2 + 0^2 + 2^2 + 1^2 + 3^2 + 0^2 + 0^2} = \sqrt{34} = 5.83$$

$$\|y\| = \sqrt{2^2 + 1^2 + 0^2 + 0^2 + 3^2 + 2^2 + 1^2 + 0^2 + 1^2} = \sqrt{20} = 4.47$$

$$\therefore \cos(x, y) = \frac{19}{5.83*4.47} = 0.729$$

Cosine similarity actually measures the angle (refer to [Fig. 4.11](#)) between  $x$  and  $y$  vectors. Hence, if cosine similarity has a value 1, the angle between  $x$  and  $y$  is  $0^\circ$  which means  $x$  and  $y$  are same except for the magnitude. If cosine similarity is 0, the angle between  $x$  and  $y$  is  $90^\circ$ . Hence, they do not share any similarity (in case of text data, no term/word is common). In the above example, the angle comes to be  $43.2^\circ$ .



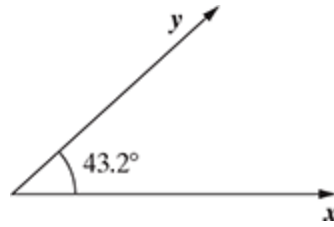


FIG. 4.11 Cosine similarity

#### 4.3.4 Overall feature selection process

Feature selection is the process of selecting a subset of features in a data set. As depicted in Figure 4.12, a typical feature selection process consists of four steps:

1. generation of possible subsets
2. subset evaluation
3. stop searching based on some stopping criterion
4. validation of the result

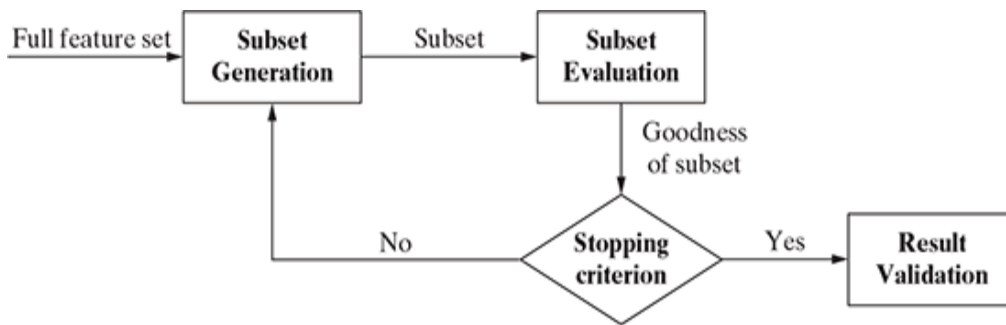


FIG. 4.12 Feature selection process

**Subset generation**, which is the first step of any feature selection algorithm, is a search procedure which ideally should produce all possible candidate subsets. However, for an  $n$ -dimensional data set,  $2^n$  subsets can be generated. So, as the value of ' $n$ ' becomes high, finding an optimal subset from all the  $2^n$  candidate subsets becomes intractable. For that reason,

different approximate search strategies are employed to find candidate subsets for evaluation. On one hand, the search may start with an empty set and keep adding features. This search strategy is termed as a sequential forward selection. On the other hand, a search may start with a full set and successively remove features. This strategy is termed as sequential backward elimination. In certain cases, search start with both ends and add and remove features simultaneously. This strategy is termed as a bi-directional selection.

Each candidate subset is then evaluated and compared with the previous best performing subset based on certain **evaluation criterion**. If the new subset performs better, it replaces the previous one.

This cycle of subset generation and evaluation continues till a pre-defined **stopping criterion** is fulfilled. Some commonly used stopping criteria are

1. the search completes
2. some given bound (e.g. a specified number of iterations) is reached
3. subsequent addition (or deletion) of the feature is not producing a better subset
4. a sufficiently good subset (e.g. a subset having better classification accuracy than the existing benchmark) is selected

Then the selected best subset is **validated** either against prior benchmarks or by experiments using real-life or synthetic but authentic data sets. In case of supervised learning, the accuracy of the learning model may be the performance parameter considered for validation. The accuracy of the model using the subset derived is compared against the model accuracy of the subset derived using some other benchmark algorithm. In case of unsupervised, the cluster quality may be the parameter for validation.

### 4.3.5 Feature selection approaches

There are four types of approach for feature selection:

1. Filter approach
2. Wrapper approach
3. Hybrid approach
4. Embedded approach

In the **filter approach** (as depicted in Fig. 4.13), the feature subset is selected based on statistical measures done to assess the merits of the features from the data perspective. No learning algorithm is employed to evaluate the goodness of the feature selected. Some of the common statistical tests conducted on features as a part of filter approach are – Pearson's correlation, information gain, Fisher score, analysis of variance (ANOVA), Chi-Square, etc.

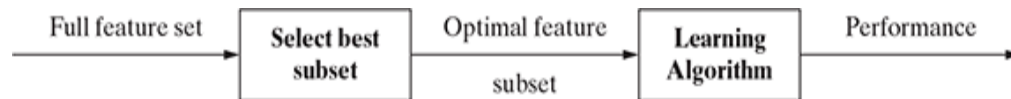


FIG. 4.13 Filter approach

In the **wrapper approach** (as depicted in Fig. 4.14), identification of best feature subset is done using the induction algorithm as a black box. The feature selection algorithm searches for a good feature subset using the induction algorithm itself as a part of the evaluation function. Since for every candidate subset, the learning model is trained and the result is evaluated by running the learning algorithm, wrapper approach is computationally very expensive. However, the performance is generally superior compared to filter approach.

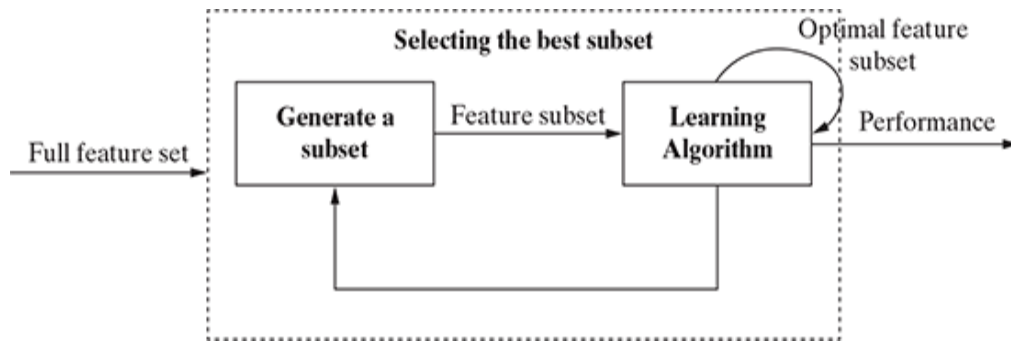


FIG. 4.14 Wrapper approach

**Hybrid approach** takes the advantage of both filter and wrapper approaches. A typical hybrid algorithm makes use of both the statistical tests as used in filter approach to decide the best subsets for a given cardinality and a learning algorithm to select the final best subset among the best subsets across different cardinalities.

**Embedded approach** (as depicted in Fig. 4.15) is quite similar to wrapper approach as it also uses an inductive algorithm to evaluate the generated feature subsets. However, the difference is it performs feature selection and classification simultaneously.

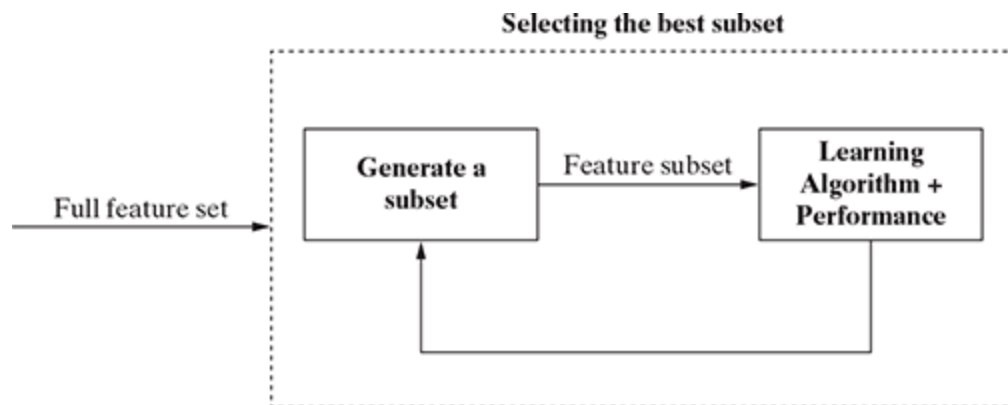


FIG. 4.15 Embedded approach

## 4.4 SUMMARY

- A feature is an attribute of a data set that is used in a machine learning process.
- Feature engineering is an important pre-processing step for machine learning, having two major elements:
  1. feature transformation
  2. feature subset selection
- Feature transformation transforms data into a new set of features which can represent the underlying machine learning problem
- There are two variants of feature transformation:
  1. feature construction
  2. feature extraction
- Feature construction process discovers missing information about the relationships between features and augments the feature space by creating additional features.
- Feature extraction is the process of extracting or creating a new set of features from the original set of features using some functional mapping.
- Some popular feature extraction algorithms used in machine learning:
  1. Principal Component Analysis (PCA)
  2. Singular Value Decomposition (SVD)
  3. Linear Discriminant Analysis (LDA)
- Feature subset selection is intended to derive a subset of features from the full feature set. No new feature is generated.
- The objective of feature selection is three-fold:
  1. Having faster and more cost-effective (i.e. less need for computational resources) learning model
  2. Improving the efficiency of the learning model
  3. Having a better understanding of the underlying model that generated the data

Feature selection intends to remove all features which are irrelevant and take a representative subset of the features which are potentially redundant. This leads to a meaningful feature subset in context of a specific learning task.

- Feature relevance is indicated by the information gain from a feature measured in terms of relative entropy.
- Feature redundancy is based on similar information contributed by multiple features measured by feature-to-feature:
  1. Correlation
  2. Distance (Minkowski distances, e.g. Manhattan, Euclidean, etc. used as most popular measures)
  3. Other coefficient-based (Jaccard, SMC, Cosine similarity, etc.)
- Main approaches for feature selection are
  1. Filter
  2. Wrapper
  3. Hybrid
  4. Embedded

## SAMPLE QUESTIONS

### MULTIPLE-CHOICE QUESTIONS (1 MARK QUESTIONS)

1. Engineering a good feature space is a crucial \_\_\_\_ for the success of any machine learning model.
  1. Pre-requisite
  2. Process
  3. Objective
  4. None of the above
2. n-gram of size 1 is called
  1. Bigram
  2. Unigram
  3. Trigram
  4. None of the above
3. Feature \_\_\_\_ involves transforming a given set of input features to generate a new set of more powerful features.
  1. Selection
  2. Engineering
  3. Transformation
  4. Re-engineering
4. Conversion of a text corpus to a numerical representation is done using \_\_\_\_ process.
  1. Tokenization
  2. Normalization
  3. Vectorization

4. None of the above
5. \_\_\_\_ approach uses induction algorithm for subset validation.
  1. Filter
  2. Hybrid
  3. Wrapper
  4. Embedded
6. In feature extraction, some of the commonly used \_\_\_\_ are used for combining the original features.
  1. Operators
  2. Delimiters
  3. Words
  4. All of the above
7. Hamming distance between binary vectors 1001 and 0101 is
  1. 1
  2. 2
  3. 3
  4. 4
8. PCA is a technique for
  1. Feature extraction
  2. Feature construction
  3. Feature selection
  4. None of the above
9. The new features created in PCA are known as
  1. Principal components
  2. Eigenvectors
  3. Secondary components
  4. None of the above
10. In LDA, intra-class and inter-class \_\_\_\_ matrices are calculated.
  1. Scatter
  2. Adjacency
  3. Similarity
  4. None of the above
11. Cosine similarity is most popularly used in
  1. Text classification
  2. Image classification
  3. Feature selection
  4. None of the above
12. This approach is quite similar to wrapper approach as it also uses and inductive algorithm to evaluate the generated feature subsets.
  1. Embedded approach
  2. Filter approach
  3. Pro Wrapper approach
  4. Hybrid approach
13. In \_\_\_\_ approach, identification of best feature subset is done using the induction algorithm as a black box.

1. Embedded
2. Filter
3. Wrapper
4. Hybrid

#### SHORT-ANSWER TYPE QUESTIONS (5 MARKS QUESTIONS)

1. What is a feature? Explain with an example.
2. What are the different situations which necessitate feature construction?
3. Explain the process of encoding nominal variables.
4. Explain the process of transforming numeric features to categorical features.
5. Explain the wrapper approach of feature selection. What are the merits and de-merits of this approach?
6. When can a feature be termed as irrelevant? How can it be measured?
7. When can a feature be termed as redundant? What are the measures to determine the potentially redundant features?
8. What are the different distance measures that can be used to determine similarity of features?
9. Compare Euclidean distance with Manhattan distance?
10. Differentiate feature transformation with feature selection
11. Write short notes on any two:
  1. SVD
  2. Hybrid method of feature selection
  3. Silhouette width
  4. ROC curve

#### LONG-ANSWER TYPE QUESTIONS (10 MARKS QUESTIONS)

1. What is feature engineering? Explain, in details, the different aspects of feature engineering?
2. What is feature selection? Why is it needed? What are the different approaches of feature selection?
3. Explain the filter and wrapper approaches of feature selection. What are the merits and demerits of these approaches?
4.
  1. Explain the overall process of feature selection
  2. Explain, with an example, the main underlying concept of feature extraction. What are the most popular algorithms for feature extraction?
5. Explain the process of feature engineering in context of a text categorization problem.
6. Why is cosine similarity a suitable measure in context of text categorization? Two rows in a document-term matrix have values - (2,



3, 2, 0, 2, 3, 3, 0, 1) and (2, 1, 0, 0, 3, 2, 1, 3, 1). Find the cosine similarity.

7.
  1. How can we calculate Hamming distance? Find the Hamming distance between 10001011 and 11001111.
  2. Compare the Jaccard index and similarity matching coefficient of two features having values (1, 1, 0, 0, 1, 0, 1, 1) and (1, 0, 0, 1, 1, 0, 0, 1).
8. What do you understand by a high-dimensional data set? Give a few practical examples? What is the challenge while applying machine learning technique on a high-dimensional data set? How can that be addressed?
9.
  1. Write short notes on any two:
    1. PCA
    2. Vectorization
    3. Embedded method
  2. Write the difference between (any two):
    1. Sequential forward selection vs. sequential backward elimination
    2. Filter vs. wrapper method of feature selection
    3. Jaccard coefficient vs. SMC