UNIT III Bayesian Concept Learning & Supervised Learning: Classification Lecture 10Hrs
Introduction, Why Bayesian Methods are Important? Bayes' Theorem, Bayes' Theorem and Concept Learning, Bayesian Belief Network Supervised Learning: Classification: Introduction**,** Example of Supervised Learning**,** Classification Model**,** Classification Learning Steps**,** Common Classification Algorithms-*k*-Nearest Neighbour(*k*NN), Decision tree, Random forest model, Support vector machines

6.1 INTRODUCTION

- Bayesian theorem was derived from the work of the 18th century mathematician Thomas Bayes. He developed the foundational mathematical principles, known as Bayesian methods, which describe the probability of events, and more importantly, how probabilities should be revised when there is additional information available.

6.2WHY BAYESIAN METHODS ARE IMPORTANT?

- Bayesian learning algorithms, like the naive Bayes classifier, are highly practical approaches to certain types of learning problems as they can calculate explicit probabilities for hypotheses. In many cases, they are equally competitive or even outperform the other learning algorithms, including decision tree and neural network algorithms.
- Bayesian classifiers use a simple idea that the training data are utilized to calculate an observed probability of each class based on feature values. When the same classifier is used later for unclassified data, it uses the observed probabilities to predict the most likely class forthe new features. The application of the observations from the training data can also be thought of as applyingour prior knowledge or prior belief to the probability of an outcome, so that it has higher probability of meeting the actual or real-life outcome. This simple concept is used in Bayes' rule and applied for training a machine inmachine learning terms. Some of the real-life uses of Bayesian classifiers are as follows:
    - Text-based classification such as spam or junk mail filtering, author identification, or topic categorization
    - Medical diagnosis such as given the presence of a set of observed symptoms during a disease, identifying the probability of new patients having the disease
    - Network security such as detecting illegal intrusion or anomaly incomputer networks
- **One of the strengths of Bayesian classifiers** is that they utilize all available parameters to subtly change the predictions, while many other algorithms tend to ignore the features that have weak effects. Bayesian classifiers assume that even if few individual parameters have small effect on the outcome, the collective effect of those parameters could be quite large. For such learning tasks, the naive Bayes classifier is most effective.
- Some of the features of Bayesian learning methods that have made them popular are as follows:
    - Prior knowledge of the candidate hypothesis is combined with the observed data for arriving at the final probability of a hypothesis. So, two important components are the prior probability of each candidate hypothesis and the probability distribution over the observed data set for each possible hypothesis.
    - The Bayesian approach to learning is more flexible than the other approaches because each observed training pattern can influence the
    - outcome of the hypothesis by increasing or decreasing the estimated probability about the hypothesis, whereas most of the other algorithms tend to eliminate a hypothesis if that is inconsistent with the single training pattern.
    - Bayesian methods can perform better than the other methods while validating the hypotheses that make probabilistic predictions. For example, when starting a new software project, on the basis of the demographics of the project, we can predict the probability of encountering challenges during execution of the project.
    - Through the easy approach of Bayesian methods, it is possible to classify new instances by combining the predictions of multiple hypotheses, weighted by their respective probabilities.
    - In some cases, when Bayesian methods cannot compute the outcome deterministically, they can be used to create a standard for the optimal decision against which the performance of other methods can be measured.

As we discussed above, the success of the Bayesian method largely depends on the availability of initial knowledge about the probabilities of the hypothesis set. So, if these probabilities are not known to us in advance,we have to use some background knowledge, previous data or assumptions

about the data set, and the related probability distribution functions to apply this method. Moreover, it normally involves high computational cost to arrive at the optimal Bayes hypothesis.

6.3 BAYES' THEOREM

- Bayes theorem is one of the most popular machine learning concepts that helps to calculate the probability of occurring one event with uncertain knowledge while other one has already occurred.
- Bayes' theorem can be derived using product rule and conditional probability of event X with known event Y:
  - According to the product rule we can express as the probability of event X with known event Y as follows;

    P(X ? Y)= P(X|Y) P(Y) {equation 1}
  - Further, the probability of event Y with known event X:

    P(X ? Y)= P(Y|X) P(X) {equation 2}
  - The general statement of Bayes' theorem is "The conditional probability of an event A, given the occurrence of another event B, is equal to the product of the event of B, given A and the probability of A divided by the probability of event B." i.e.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- The above equation is called as Bayes Rule or Bayes Theorem.
  - P(A|B) is called as **posterior**, which we need to calculate. It is defined as updated probability after considering the evidence.
  - P(B|A) is called the **likelihood.** It is the probability of evidence when hypothesis is true.
  - P(A) is called the **prior probability**, probability of hypothesis before considering the evidence
  - P(B) is called marginal probability. It is defined as the probability of evidence under any consideration.

6.3.1 Prior

- The prior knowledge or belief about the probabilities of various hypotheses in *H* is called Prior in context of Bayes' theorem. For example, if we have to determine whether a particular type of tumour is malignant for a patient, the prior knowledge of such tumours becoming malignant can be used to validate our current hypothesis and is a prior probability or simply called Prior.
- We will assume that $P(h)$ is the initial probability of a hypothesis '*h*' that the patient has a malignant tumour based only on the malignancy test, without considering the prior knowledge of the correctness of the test process or the so-called training data. Similarly, $P(T)$ is the prior probability that the training data will be observed or, in this case, the probability of positive malignancy test results. We will denote $P(T|h)$ as the probability of observing data *T* in a space where '*h*' holds true, which means the probability of the test results showing a positive value when the tumour is actually malignant.

6.3.2 Posterior

- The probability that a particular hypothesis holds for a data set based on the Prior is called the posterior probability or simply Posterior.
- In the above example, the probability of the hypothesis that the patient has a malignant tumour considering the **Prior** of correctness of the malignancy test is a **posterior probability**.
- In our notation, we will say that we are interested in finding out $P(h|T)$, which means whether the hypothesis holds true given the observed training data *T*. This is called the posterior probability or simply Posterior in machine learning language. So, the prior probability $P(h)$, which represents the probability of the hypothesis independent of the training data (Prior), now gets refined with the introduction of influence of the training data as $P(h|T)$.

According to Bayes' theorem

$$P(h|T) = \frac{P(T|h)P(h)}{P(T)}$$

combines the prior and posterior probabilitiestogether.

From the above equation, we can deduce that $P(h|T)$ increases as $P(h)$ and $P(T|h)$ increases and also as $P(T)$ decreases. The simple explanation is that when there is more probability that T can occur independently of h then it is less probable that h can get support from T inits occurrence.

It is a common question in machine learning problems to find out the maximum probable hypothesis h from a set of hypotheses H ($h \in H$) given the observed training data T. This maximally probable hypothesis is called the **maximum a posteriori (MAP)** hypothesis. By using Bayes' theorem, we can identify the MAP hypothesis from the posterior probability of each candidate hypothesis:

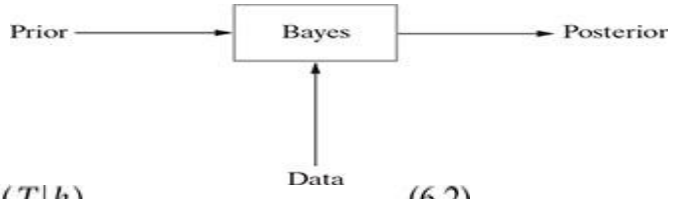
$$h_{\text{MAP}} = \text{argmax}_{h \in H} P(h|T)$$
$$= \text{argmax}_{h \in H} \frac{P(T|h)P(h)}{P(T)}$$

and as $P(T)$ is a constant independent of h, in this case,we can write

$$= \text{argmax}_{h \in H} P(T|h)P(h) \qquad (6.1)$$

6.3.3 Likelihood

In certain machine learning problems, we can further simplify equation 6.1 if every hypothesis in H has equal probable priori as $P(h_i) = P(h_j)$, and then, we can determine $P(h|T)$ from the probability $P(T|h)$ only. Thus, $P(T|h)$ is called the likelihood of data T given h, and any hypothesis that maximizes $P(T|h)$ is called the maximumlikelihood (ML) hypothesis, h_{ML}. See figure 6.1 and 6.2 for the conceptual and mathematical representation of Bayes theorem and the relationship of Prior, Posterior and Likelihood.



$$h_{\text{ML}} = \text{argmax}_{h \in H} P(T|h) \qquad (6.2)$$

FIG. 6.1 Bayes' theorem

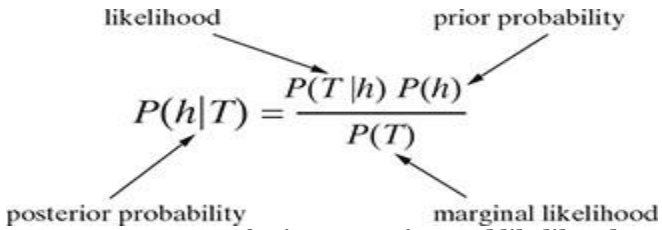


FIG. 6.2 Concept of prior, posterior, and likelihood

Example. Let us take the example of malignancy identification in a particular patient's tumour as an application for Bayes rule. We will calculate how the prior knowledge of the percentage of cancer cases in a sample population and probability of the test result being correct influence the probability outcome of the correct diagnosis. We have two alternative hypotheses:

(1) a particular tumour is of malignant type and (2) aparticular tumour is non-malignant type. The priori available are—1. only 0.5% of the population has this

kind of tumour which is malignant, 2. the laboratory report has some amount of incorrectness as it could detect the malignancy was present only with 98% accuracy whereas could show the malignancy was not present correctly only in 97% of cases. This means the test predicted malignancy was present which actually was a false alarm in 2% of the cases, and also missed detecting the real malignant tumour in 3% of the cases.

Solution: Let us denote Malignant Tumour = MT, Positive Lab Test = PT, Negative Lab Test = NT

h_1 = the particular tumour is of malignant type = MT in our example

h_2 = the particular tumour is not malignant type = !MT in our example

$P(\text{MT}) = 0.005$ $P(\text{!MT}) = 0.995$

$P(\text{PT}|\text{MT}) = 0.98$ $P(\text{PT}|\text{!MT}) = 0.02$

$P(\text{NT}|\text{!MT}) = 0.97$ $P(\text{NT}|\text{MT}) = 0.03$

So, for the new patient, if the laboratory test report shows positive result, let us see if we should declare this as the malignancy case or not:

$$\begin{aligned} P(h_1|\text{PT}) &= \frac{P(\text{PT}|h_1).P(h_1)}{P(\text{PT})} \\ &= P(\text{PT}|\text{MT})P(\text{MT}) \\ &= 0.98 \times 0.005 \\ &= 0.0049 \\ &= 0.49\% \end{aligned}$$

$$\begin{aligned} P(h_2|\text{PT}) &= \frac{P(\text{PT}|h_2).P(h_2)}{P(\text{PT})} \\ &= P(\text{PT}|\text{!MT})P(\text{!MT}) \\ &= 0.02 \times 0.995 \\ &= 0.0199 \\ &= 1.99\% \end{aligned}$$

As $P(h_2|\text{PT})$ is higher than $P(h_1|\text{PT})$, it is clear that the hypothesis h_2 has more probability of being true. So, $h\text{MAP} = h2 = \text{!MT}$.

This indicates that even if the posterior probability of malignancy is significantly higher than that of non- malignancy, the probability of this patient not having malignancy is still higher on the basis of the prior knowledge. Also, it should be noted that through Bayes' theorem, we identified the probability of one hypothesis being higher than the other hypothesis, and we did not completely accept or reject the hypothesis by this theorem. Furthermore, there is very high dependency on the availability of the prior data for successful application of Bayes' theorem.

6.4 BAYES' THEOREM AND CONCEPT LEARNING

One simplistic view of concept learning can be that if we feed the machine with the training data, then it can calculate the posterior probability of the hypotheses and outputs the most probable hypothesis. This is also called brute-force Bayesian learning algorithm, and it is also observed that consistency in providing the right probable hypothesis by this algorithm is very comparable to the other algorithms.

6.4.1 Brute-force Bayesian algorithm

We will now discuss how to use the MAP hypothesis output to design a simple learning algorithm called brute-force map learning algorithm. Let us assume that the learner considers a finite hypothesis space H in which the learner will try to learn some target concept $c{:}X \rightarrow \{0,1\}$ where X is the instance space corresponding to H. The sequence of training examples is $\{(x_1, t_1), (x_2, t_2),\ldots, (x_m, t_m)\}$, where x_i is the instance of X and t_i is the target concept of x_i defined as $t_i = c(x_i)$. Without impacting the efficiency of the algorithm, we can assume that the sequence of instances of x $\{x_1,\ldots, x_m\}$ is held fixed, and then, the sequence of target values becomes $T = \{t_1,\ldots, t_m\}$.

For calculating the highest posterior probability, we can use Bayes' theorem as discussed earlier in this chapter:

Calculate the posterior probability of each hypothesis h in H:

$$P(h|T) = \frac{P(T|h)P(h)}{P(T)}$$

Identify the h_{MAP} with the highest posteriorprobability

$$h\text{MAP} = \text{argmax}h \in HP(h|T)$$

Please note that calculating the posterior probability for each hypothesis requires a very high volume of computation, and for a large volume of hypothesis space,this may be difficult to achieve.

Let us try to connect the concept learning problem with the problem of identifying the h_{MAP}. On the basis ofthe probability distribution of $P(h)$ and $P(T|h)$, we can derive the prior knowledge of the learning task. There are few important assumptions to be made as follows:

1. The training data or target sequence T is noise free, which means that it is a direct function of X only (i.e. $t_i = c(x_i)$)
2. The concept c lies within the hypothesis space H
3. Each hypothesis is equally probable and independent of eachother

On the basis of assumption 3, we can say that each hypothesis h within the space H has equal prior probability, and also because of assumption 2, we can say that these prior probabilities sum up to 1. So, we canwrite

$$P(h) = \frac{1}{|H|} \text{ for all } h \text{ within } H \qquad (6.3)$$

$P(T|h)$ is the probability of observing the target valuest_i in the fixed set of instances $\{x_i,\ldots, x_m\}$ in the space where h holds true and describes the concept c correctly.
Using assumption 1 mentioned above, we can say that if$T$ is consistent with h, then the probability of data T given the hypothesis h is 1 and is 0 otherwise:

$$P(T|h) = \begin{cases} 1 & \text{if } t_i = h(x_i) \text{ for all } t_i \text{ within } T \\ 0 & \text{otherwise} \end{cases} \qquad (6.4)$$

Using Bayes' theorem to identify the posteriorprobability

$$P(h|T) = \frac{P(T|h)P(h)}{P(T)} \qquad (6.5)$$

For the cases when h is inconsistent with the trainingdata T, using 6.5 we get

$$P(h|T) = \frac{0 \times P(h)}{P(T)} = 0$$

with T, when h is inconsistent and when h is consistent with T

$$P(h|T) = \frac{1 \times \frac{1}{|H|}}{P(T)} = \frac{1}{|H|P(T)} \qquad (6.6)$$

Now, if we define a subset of the hypothesis H which isconsistent with T as H_D, then by using the total probability equation, we get

$$P(T) = \sum_{h_i \in H_D} P(T|h_i)P(h_i)$$
$$= \sum_{h_i \in H_D} 1.\frac{1}{|H|} + \sum_{h_i \notin H_D} 0.\frac{1}{|H|}$$
$$= \sum_{h_i \in H_D} 1.\frac{1}{|H|}$$
$$= \frac{|H_D|}{|H|}$$

This makes 6.5 as

$$P(h|T) = \frac{1}{|H|.\frac{|H_D|}{|H|}}$$
$$= \frac{1}{|H_D|}$$

So, with our set of assumptions about $P(h)$ and $P(T|h)$,we get the posterior probability $P(h|T)$ as

$$P(h|T) = \begin{cases} \frac{1}{|H_D|} & \text{if } h \text{ is consistent with } T \\ 0 & \text{otherwise} \end{cases} \qquad (6.7)$$

where H_D is the number of hypotheses from the space H which are consistent with target data set T. The interpretation of this evaluation is that initially, each hypothesis has equal probability and, as we introduce thetraining data, the posterior probability of inconsistent hypotheses becomes zero and the total probability that sums up to 1 is distributed equally among the consistent hypotheses in the set. So, under this condition, each consistent hypothesis is a MAP hypothesis with posterior probability $\frac{1}{|H_D|}$.

6.4.2 Concept of consistent learners

From the above discussion, we understand the behaviour of the general class of learner whom we call as consistent learners. So, the group of learners who commit zero error over the training data and output the hypothesis are called *consistent learners*. If the training data is noise free and deterministic (i.e. $P(D|h) = 1$ if D and h are consistent and 0 otherwise) and if there is uniform prior probability distribution over H (so, $P(h_m) = P(h_n)$ for all m, n), then every consistent learner outputs the MAP hypothesis. An important application of this conclusion is that Bayes' theorem can characterize the behaviour of learning algorithms even when the algorithm does not explicitly manipulate the probability. As it can help to identify the optimal distributions of $P(h)$ and $P(T|h)$ under which the algorithm outputs the MAP hypothesis, the knowledge can be used to characterize the assumptions under which the algorithms behave optimally.

Though we discussed in this section a special case of Bayesian output which corresponds to the noise-free training data and deterministic predictions of hypotheseswhere $P(T|h)$ takes on value of either 1 or 0, the theorem can be used with the same effectiveness for noisy training data and additional assumptions about the probability distribution governing the noise.

6.4.3 Bayes optimal classifier

- What is the most probable classification of the new instance given the training data. To illustrate the concept, let us assume three hypotheses h_1, h_2, and h_3 in the hypothesis space h. Let the posterior probability of these hypotheses be 0.4, 0.3, and 0.3, respectively.
- There is a new instance x,which is classified as true by h_1, but false by h_2 and h_3.
- Then the most probable classification of the new instance (x) can be obtained by combining

the predictions of all hypotheses weighed by their corresponding posterior probabilities. By denoting the possible classification of the new instance as c_i from the set C, the probability $P(c_i|T)$ that the correct classification for the new instance is c_i is

$$P(c_i|T) = \sum_{h_i \in H} P(c_i|h_i)P(h_i|T)$$

- The optimal classification is for which $P(c_i|T)$ ismaximum is

$$\text{Bayes optimal classifier} = \underset{c_i \in C}{\text{argmax}} \sum_{h_i \in H} P(c_i|h_i)P(h_i|T)$$

- So, extending the above example, The set of possible outcomes for the new instance x is within the set C = {True, False} and

  $P(h1 \mid T) = 0.4$, $P(\text{False} \mid h1) = 0$, $P(\text{True} \mid h1) = 1 P(h2 \mid T) = 0.3$, $P(\text{False} \mid h2) = 1$, $P(\text{True} \mid h2) = 0 P(h3 \mid T) = 0.3$, $P(\text{False} \mid h3) = 1$, $P(\text{True} \mid h3) = 0$

Then,

$$\sum_{h_i \in H} P(\text{True}|h_i)P(h_i|T) = 0.4$$

$$\sum_{h_i \in H} P(\text{False}|h_i)P(h_i|T) = 0.6$$

and

$$\underset{c_i \in \{\text{True, False}\}}{\text{argmax}} \sum_{h_i \in H} P(c_i|h_i)P(h_i|T) = \text{False}$$

- This method maximizes the probability that the new instance is classified correctly when the available training data, hypothesis space and the prior probabilities of the hypotheses are known. **This is thus also called Bayes optimal classifier.**

6.4.4 Naïve Bayes classifier

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.
- Naïve Bayes is a simple technique for building classifiers: models that assign class labels to problem instances. The basic idea of Bayes rule is that the outcome of a hypothesis can be predicted on the basis of some evidence (E) that can be observed.
- From Bayes rule, we observed that
  - A prior probability of hypothesis h or $P(h)$: This is the probability of an event or hypothesis before the evidence is observed.
  - A posterior probability of h or $P(h|D)$: This is the probability of an event after the evidence is observed within the population D.

$$\text{Posterior probability} = \frac{\text{(Prior probability} \times \text{Conditional Probability)}}{\text{Evidence}}$$

- **Posterior Probability is of the format 'What is the probability that a particular object belongs to class i given its observed feature values?'**
- For example, a person has height and weight of 182 cmand 68 kg, respectively. What is the probability that this person belongs to the class 'basketball player'? This can be predicted using the Naïve Bayes classifier. **This is known as probabilistic classifications.**
- Let us see the basis of deriving the principles of Naïve Bayes classifiers. We take a learning task where each instance x has some attributes and the target function ($f(x)$) can take any value from the finite set of classification values C. We also have a set of training examples

for target function, and the set of attributes $\{a_1, a_2, \ldots, a_n\}$ for the new instance are known to us. Our task is to predict the classification of the new instance.

- According to the approach in Bayes' theorem, the classification of the new instance is performed by assigning the most probable target classification C_{MAP} on the basis of the attribute values of the new instance $\{a_1, a_2, \ldots, a_n\}$. So,

$$C_{MAP} = \underset{c_i \in C}{\operatorname{argmax}} \sum_{h_i \in H} P(c_i | a_1, a_2, \ldots, a_n)$$

which can be rewritten using Bayes' theorem as

$$C_{MAP} = \underset{c_i \in C}{\operatorname{argmax}} \sum_{h_i \in H} \frac{P(a_1, a_2, \ldots, a_n | c_i) P(c_i)}{P(a_1, \ldots a_n)}$$

As combined probability of the attributes defining thenew instance fully is always 1

$$C_{MAP} = \underset{c_i \in C}{\operatorname{argmax}} \sum_{h_i \in H} P(a_1, a_2, \ldots, a_n | c_i) P(c_i) \qquad (6.8)$$

- So, to get the most probable classifier, we have to evaluate the two terms $P(a_1, a_2, \text{c}, a_n|c_i)$ and $P(ci)$. In a practical scenario, it is possible to calculate $P(ci)$ by calculating the frequency of each target value c_i in the training data set. But the $P(a_1, a_2, \text{c}, a_n|c_i)$ cannot be estimated easily and needs a very high effort of calculation.
- Thus, the Naïve Bayes classifier makes a simple assumption that the attribute values are conditionally independent of each other for the target value. So, applying this simplification, we can now say that for a target value of an instance, the probability of observing the combination $a_1, a_2, \ldots, a_n$ is the product of probabilities of individual attributes $P(a_i|c_j)$.

$$P(a_1, a_2, \ldots, a_n | c_j) = \prod_i P(a_i | c_j)$$

- Then, from 6.8, we get the approach for theNaïve Bayes classifier as

$$C_{NB} = \underset{c_i \in C}{\operatorname{argmax}} \sum_{h_i \in H} P(c_i) \prod_i P(a_i | c_j) \qquad (6.9)$$

Strengths and weaknesses of NaïveBayes classifiers are :

Table 6.1 *Strengths and Weaknesses of Bayes Classifiers*

Strengths	Weakness
Simple and fast in calculation but yet effective in result	The basis assumption of equal importance and independence often does not hold true
In situations where there are noisy and missing data, it performs well	If the target dataset contains large numbers of numeric features, then the reliability of the outcome becomes limited
Works equally well when smaller number of data is present for training as well as very large number of training data is available	Though the predicted classes have a high reliability, estimated probabilities have relatively lower reliability
Easy and straightforward way to obtain the estimated probability of a prediction	

Example. Let us assume that we want to predict the outcome of a football world cup match on the basis of the past performance data of the playing teams. We have training data available (refer Fig. 6.3) for actual match outcome, while four parameters are considered – Weather Condition (Rainy, Overcast, or Sunny), how many matches won were by this team out of the last three matches (one match, two matches, or three matches), Humidity Condition (High or Normal), and whether they won the toss (True or False). Using Naïve Bayesian, you need to classify the conditions when this team wins and then predict the probability of this team winning a particular match when Weather Conditions = Rainy, they won two of the last three matches, Humidity = Normal and they won the toss in the particular match.

Weather Condition	Wins in last 3 matches	Humidity	Win toss	Won match?
Rainy	3 wins	High	FALSE	No
Rainy	3 wins	High	TRUE	No
OverCast	3 wins	High	FALSE	Yes
Sunny	2 wins	High	FALSE	Yes
Sunny	1 win	Normal	FALSE	Yes
Sunny	1 win	Normal	TRUE	No
OverCast	1 win	Normal	TRUE	Yes
Rainy	2 wins	High	FALSE	No
Rainy	1 win	Normal	FALSE	Yes
Sunny	2 wins	Normal	FALSE	Yes
Rainy	2 wins	Normal	TRUE	Yes
OverCast	2 wins	High	TRUE	Yes
OverCast	3 wins	Normal	FALSE	Yes
Sunny	2 wins	High	TRUE	No

FIG. 6.3 Training data for the Naïve Bayesian method

6.4.4.1 Naïve Bayes classifier steps

Step 1: First construct a frequency table. A frequency table is drawn for each attribute against the target outcome. For example, in Figure 6.3, the various attributes are (1) Weather Condition, (2) How many matches won by this team in last three matches, (3) Humidity Condition, and (4) whether they won the toss and the target outcome is will they win the match or not?

Step 2: Identify the cumulative probability for 'Won match = Yes' and the probability for 'Won match = No' on the basis of all the attributes. Otherwise, simply multiply probabilities of all favourable conditions to derive 'YES' condition. Multiply probabilities of all non-favourable conditions to derive 'No' condition.

Step 3: Calculate probability through normalization by applying the below formula

$$P(\text{Yes}) = \frac{P(\text{Yes})}{P(\text{Yes}) + P(\text{No})}$$

$$P(\text{No}) = \frac{P(\text{No})}{P(\text{Yes}) + P(\text{No})}$$

P(Yes) will give the overall probability of favourable condition in the given scenario.
P(No) will give the overall probability of non-favourable condition in the given scenario.

Solving the above problem with Naive Bayes

Step 1: Construct a frequency table. The posterior probability can be easily derived by constructing a frequency table for each attribute against the target. For example, frequency of Weather Condition variable with values 'Sunny' when the target value Won match is 'Yes', is, 3/(3+4+2) = 3/9.

Figure 6.4 shows the frequency table thus constructed.

	Won Match			Won Match	
Weather condition	Yes	No	**Humidity**	Yes	No
Sunny	3	2	High	3	4
OverCast	4	0	Normal	6	1
Rainy	2	3			
Total	9	5	Total	9	5

	Won Match			Won Match	
Wins in last 3 matches	Yes	No	Win toss	Yes	No
3 wins	2	2	FALSE	6	2
1 win	4	2	TRUE	3	3
2 wins	3	1			
Total	9	5	Total	9	5

FIG. 6.4 Construct frequency table

Step 2:

To predict whether the team will win for given weather conditions (a_1) = Rainy, Wins in last three matches (a_2) = 2 wins, Humidity (a_3) = Normal and Win toss (a_4) = True, we need to choose 'Yes' from the above table for the given conditions.

From Bayes' theorem, we get

$$P(\text{Win match}|a_1 \cap a_2 \cap a_3 \cap a_4) = \frac{P(a_1 \cap a_2 \cap a_3 \cap a_4|\text{Win match})P(\text{Win match})}{P(a_1 \cap a_2 \cap a_3 \cap a_4)}$$

This equation becomes much easier to resolve if we recall that Naïve Bayes classifier assumes independence among events. This is specifically true for class-conditional independence, which means that the events are independent so long as they are conditioned on the same class value. Also, we know that if the events are independent, then the probability rule says, $P(A \cap B) = P(A)\,P(B)$, which helps in simplifying the above equation significantly as

$P(\text{Win match}|a_1 \cap a_2 \cap a_3 \cap a_4)$

$$= \frac{P(a_1|\text{Win match})P(a_2|\text{Win match})P(a_3|\text{Win match})P(a_4|\text{Win match})P(\text{Win match})}{P(a_1)P(a_2)P(a_3)P(a_4)}$$

$= 2/9*4/9*6/9*9/14$

$= 0.014109347$

This should be compared with

$P(!\text{Win match}|a_1 \cap a_2 \cap a_3 \cap a_4)$

$$= \frac{P(a_1|!\text{Win match})P(a_2|!\text{Win match})P(a_3|!\text{Win match})P(a_4|!\text{Win match})P(!\text{Win match})}{P(a_1)P(a_2)P(a_3)P(a_4)}$$

$= 3/5*2/5*1/5*5/14$

$= 0.010285714$

Step 3: by normalizing the above two probabilities, we can ensure that the sum of these two probabilities is 1.

$$P(\text{Win match}) = \frac{P(\text{Win match})}{P(\text{Win match}) + P(\text{!Win match})}$$
$$= \frac{0.014109347}{0.014109347 + 0.010285714}$$
$$= 0.578368999$$

$$P(\text{!Win match}) = \frac{P(\text{!Win match})}{P(\text{Win match}) + P(\text{!Win match})}$$
$$= \frac{0.010285714}{0.014109347 + 0.010285714}$$
$$= 0.421631001$$

Conclusion: This shows that there is 58% probability that the team will win if the above conditions become true for that particular day. Thus, Naïve Bayes classifier provides a simple yet powerful way to consider the influence of multiple attributes on the target outcome and refine the uncertainty of the event on the basis of the prior knowledge because it is able to simplify the calculation through independence assumption.

6.4.5 Applications of Naïve Bayes classifier

- **Text classification:** Naïve Bayes classifier is among the most successful known algorithms for learning to classify text documents. It classifies the document where the probability of classifying the text is more. It uses the above algorithm to check the permutation and combination of the probability of classifying a document under a particular 'Title'. It has various applications in document categorization, language detection, and sentiment detection, which are very useful for traditional retailers, e-retailors, and other businesses on judging the sentiments of their clients on the basis of keywords in feedback forms, social media comments, etc.
- **Spam filtering:** Spam filtering is the best known use of Naïve Bayesian text classification. Presently, almost all the email providers have this as a built-in functionality, which makes use of a Naïve Bayes classifier to identify spam email on the basis of certain conditions and also the probability of classifying an email as 'Spam'. Naïve Bayesian spam sifting has turned into a mainstream mechanism to recognize illegitimate a spam email from an honest-to-goodness email (sometimes called 'ham'). Users can also install separate email filtering programmes. Server-side email filters such as DSPAM, Spam Assassin, Spam Bayes, and ASSP make use of Bayesian spam filtering techniques, and the functionality is sometimes embedded within the mail server software itself.
- **Hybrid Recommender System:** It uses Naïve Bayes classifier and collaborative filtering. Recommender systems (used by e-retailors like eBay, Alibaba, Target, Flipkart, etc.) apply machine learning and data mining techniques for filtering unseen information and can predict whether a user would like a given resource. For example, when we log in to these retailer websites, on the basis of the usage of texts used by the login and the historical data of purchase, it automatically recommends the product for the particular login persona. One of the algorithms is combining a Naïve Bayes classification approach with collaborative filtering, and experimental results show that this algorithm provides better performance regarding accuracy and coverage than other algorithms.
- **Online Sentiment Analysis:** The online applications use supervised machine learning (Naïve Bayes) and useful computing. In the case of sentiment analysis, let us assume there are three sentiments such as nice, nasty, or neutral, and Naïve Bayes classifier is used to distinguish between them. Simple emotion modelling combines a statistically based classifier with a dynamical model. The Naïve Bayes classifier employs 'single words' and 'word pairs' like features and determines the sentiments of the users. It allocates user utterances into nice,

nasty, and neutral classes, labelled as +1, −1, and 0, respectively. This binary output drives a simple first- order dynamical system, whose emotional state represents the simulated emotional state of the experiment's personification.

6.4.6 Handling Continuous Numeric Features in Naïve Bayes Classifier

- In the above example, we saw that the Naïve Bayes classifier model uses a frequency table of the training data for its calculation. Thus, each attribute data should be categorical in nature so that the combination of class and feature values can be created. But this is not possible in the case of continuous numeric data as it does not have the categories of data.
- The workaround that is applied in these cases is discretizing the continuous data on the basis of some data range. **This is also called binning as the individual categories are termed as bins.**
- For example, let us assume we want to market a certain credit card to all the customers who are visiting a particular bank. We have to classify the persons who are visiting a bank as either interested candidate for taking a new card or non- interested candidate for a new card, and on the basis of this classification, the representative will approach the customer for sale..
- If we plot the number of customers visiting the bank during the 8 hours of banking time, the distribution graph will be a continuous graph. But if we introduce a logic to categorize the customers according to their time of entering the bank, then we will be able to put the customers in 'bins' or buckets for our analysis.
- We can then try to assess what time range is best suited for targeting the customers who will have interest in the new credit card. The bins created by categorizing the customers by their time of entry looks like Figure 6.5.
- This creates eight natural bins for us (or we may change the number of bins by changing our categorizing criteria), which can now be used for Bayes analysis.

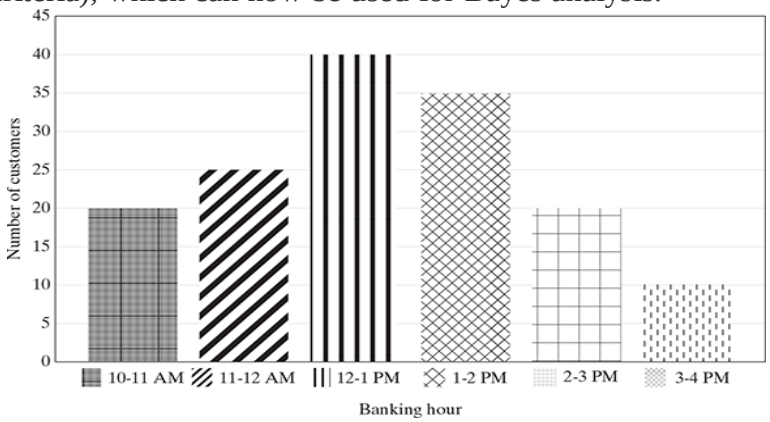


FIG. 6.5 The distribution of bins based on the time of entry of customers in the bank

6.5 BAYESIAN BELIEF NETWORK

- We must have noted that a significant assumption in the Naïve Bayes classifier was that the attribute values a_1, a_2,…, a_n are conditionally independent for a target value. The Naïve Bayes classifier generates optimal output when this condition is met. Though this assumption significantly reduces the complexity of computation, in many practical scenarios, this requirement of conditional independence becomes a difficult constraint for the application of this algorithm.
- So, in this section, we will discuss the approach of Bayesian Belief network, which assumes that within the set of attributes, the probability distribution can have conditional probability relationship as well as conditional independence assumptions. This is different from the Naïve Bayes assumption of conditional independence of all the attributes as the belief network provides the flexibility of declaring a subset of the attributes as conditionally dependent while leaving rest of the attributes to hold the assumptions of conditional independence. The prior knowledge or belief about the influence of one attribute over the other is handled through joint probabilities as discussed later in this section.

- Let us refresh our mind on the concept of conditional probability. If an uncertain event A is conditional on a knowledge or belief K, then the degree of belief in A with the assumption that K is known is expressed as $P(A|K)$. Traditionally, conditional probability is expressed by joint probability as follows:

$$P(A|K) = \frac{P(A,K)}{P(K)} \tag{6.10}$$

Rearranging (6.9), we get the product rule

$P(A, K) = P(A|K)P(K)$

This can be extended for three variables or attributes as

$P(A, K, C) = P(A|K, C)P(K, C) = P(A|K, C)P(K|C)P(C)$

For a set of n attributes, the generalized form of theproduct rule becomes

$P(A_1, A_2, \ldots, A_n) = P(A_1|A_2, \ldots, A_n)P(A_2|A_3, \ldots, A_n)P(A_n\text{- }1|A_n)P(A_n)$ (6.10)

This generalized version of the product rule is calledthe Chain Rule.

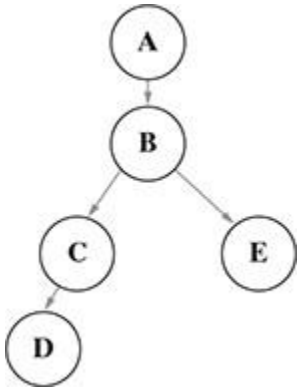


FIG. 6.6 Chain rule

Let us understand the chain rule by using the diagram in Figure 6.6. From the joint probability formula 6.10, wecan write

$P(A, B, C, D, E) = P(A|B, C, D, E)P(B|C, D, E)P(C|D, E)P(D|E)P(E)$

But from Figure 6.6, it is evident that E is not related to C and D, which means that the probabilities of variables C and D are not influenced by E and vice versa. Similarly, A is directly influenced only by B. By applying this knowledge of independence, we can simplify the above equation as

$P(A, B, C, D, E) = P(A|B)P(B|C, D)P(C|D)P(D)P(E)$

Let us discuss this concept of independence and conditional independence in detail in the next section.

6.5.1 Independence and conditional independence:

We represent the conditional probability of A with knowledge of K as $P(A|K)$. The variables A and K are said to be independent if $P(A|K) = P(A)$, which means that there is no influence of K on the uncertainty of A. Similarly, the joint probability can be written as $P(A,K) =P(A)P(K)$.

Extending this concept, the variables A and K are saidto be conditionally independent given C if $P(A|C) = P(A|K, C)$.

This concept of conditional independence can also be extended to a set of attributes. We can say that the set of variables $A_1, A_2,\ldots, A_n$ is conditionally independent of theset of variables $B_1, B_2,\ldots, B_m$ given the set of variables $C_1, C_2,\ldots, C_l$ if

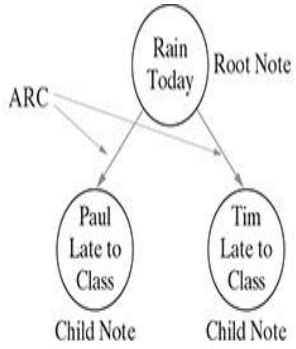
$P(A_1, A_2,\ldots, A_n|B_1, B_2,\ldots, B_m, C_1, C_2,\ldots, C_1) = P(A_1, A_2,\ldots, A_n|C_1, C_2,\ldots, C_l)$

If we compare this definition with our assumption in the Naïve Bayes classifier, we see that the Naïve Bayes classifier assumes that the instance attribute A_1 is conditionally independent of the instance attribute A_2, given the target value V, which can be written using the general product rule and application of conditional independence formula as

$$P(A_1, A_2|V) = P(A_1|A_2, V)P(A_2, V) = P(A_1, V)P(A_2, V)$$

A Bayesian Belief network describes the joint probability distribution of a set of attributes in their joint space. In Figure 6.7, a Bayesian Belief network is presented. The diagram consists of nodes and arcs. The nodes represent the discrete or continuous variables for which we are interested to calculate the conditional probabilities. The arc represents the causal relationship of the variables.

The two important information points we get from this network graph are used for the determining the joint probability of the variables. First, the arcs assert that the node variables are conditionally independent of its non- descendants in the network given its immediate predecessors in the network. If two variables A and B are connected through a directed path, then B is called the descendent of A. Second, the conditional probability table for each variable provides the probability distribution of that variable given the values of its immediate predecessors. We can use Bayesian probability to calculate different behaviours of the variables in Figure 6.7.



Rain Today (Root Node) Probability	
True	0.1
False	0.9

Node Conditional probability	Rain Today	True	False
Paul Late to Class	True	0.6	0.5
	False	0.4	0.5

Node Conditional probability	Rain Today	True	False
Tim Late to Class	True	0.8	0.1
	False	0.2	0.9

FIG. 6.7 Bayesian belief network

1. The unconditional probability that Tim is late to class –
   P(Tim is late to class)
   = P(Tim late|Rain Today)P(Rain Today) + P(Tim late|No Rain Today)*P(No Rain Today)
   = (0.8 × 0.1) × (0.1 × 0.9)
   = 0.17
   From this unconditional probability, the most important use of the Bayesian Belief network is to find out the revised probability on the basis of the prior knowledge. If we assume that there was rain today, then the probability table can quickly provide us the information about the probability of Paul being late to class or the probability of Tim being late to class from the probability distribution table itself. But if we do not know whether there was rain today or not, but we only know that Tim is late to class today, then we can arrive at the following probabilities –
2. The revised probability that there was rain today –

$$P(\text{Rain Today}|\text{Tim late to class}) = \frac{P(\text{Tim late}|\text{Rain today})\,P(\text{Rain today})}{P(\text{Tim late})}$$
$$= \frac{(0.8 \times 0.1)}{0.17}$$
$$= 0.47$$
$$= P(\text{Rain Today}) \text{ as Tim late to class is already known}$$

3. The revised probability that Paul will be late to class today –P(Paul late to class today)
   = P(Paul late|Rain today)P(Rain today) + P(Paul late|No rain today)P(No rain today)
   = (0.6 × 0.47) + (0.5 × (1-0.47))
   = 0.55

Here, we used the concept of hard evidence and soft evidence. Hard evidence (instantiation) of a node is evidence that the state of the variable is definitely as a particular value. In our above example, we had hard evidence that 'Tim is late to class'. If a particular node is instantiated, then it will block propagation of evidence further down to its child nodes. Soft evidence for a node is the evidence that provides the prior probability values for the node. The node 'Paul is late to class' is soft evidenced with the prior knowledge that 'Tim is late to class'.

The Bayesian Belief network can represent much more complex scenarios with dependence and independence concepts. There are three types of connections possible in a Bayesian Belief network. **Diverging Connection:** In this type of connection, the evidence can be transmitted between two child nodes of the same parent provided that the parent is not instantiated. In Figure 6.7, we already saw the behaviour of diverging connection.

Serial Connection: In this type of connection, any evidence entered at the beginning of the connection can be transmitted through the directed path provided that no intermediate node on the path is instantiated (see Fig. 6.8 for illustration).

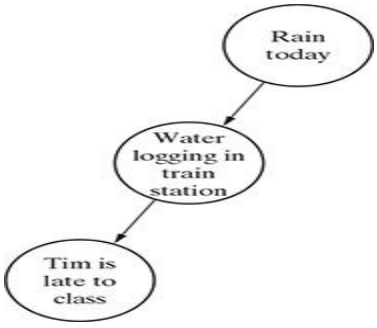


FIG. 6.8 Serial connection

Converging Connection: In this type of connection, the evidence can only be transmitted between two parents when the child (converging) node has received some evidence and that evidence can be soft or hard (see Fig. 6.9 for illustration).

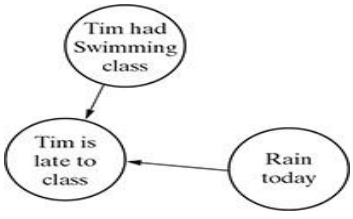


FIG. 6.9 Convergent connection

As discussed above, by using the Bayesian network, we would like to infer the value of a target variable on the basis of the observed values of some other variables.
Please note that it will not be possible to infer a single value in the case of random variables we are dealing with, but our intention is to infer the probability distribution of the target variable given the observed values of other variables. In general, the Bayesian network can be used to compute the probability distribution of any subset of node variables given the values or distribution of the remaining variables.

6.5.2 Use of the Bayesian Belief network in machine learning:

We have seen that the Bayesian network creates a complete model for the variables and their relationships and thus can be used to answer probabilistic queries about them. A common use of the network is to find out the updated knowledge about the state of a subset of variables, while the state of the other subset (known as the evidence variables) is observed. This concept, often known as probabilistic inference process of computing the posterior distribution of variables, given some evidences, provides a universal sufficient statistic for applications related to detections. Thus if one wants to choose the values for a subset of variables in order to minimize some expected loss functions or decision errors, then this method is quiet effective. In other words, the Bayesian network is a mechanism for automatically applying Bayes' theorem to complex problems. Bayesian networks are used for modelling beliefs in domains like computational biology and bioinformatics such as protein structure and gene regulatory networks, medicines, forensics, document classification, information retrieval, image processing, decision support systems, sports betting and gaming, property market analysis and various other fields.

SAMPLE QUESTIONS

SHORT ANSWER-TYPE QUESTIONS (5 MARKS EACH)

1. What is prior probability? Give an example.
2. What is posterior probability? Give an example.
3. What is likelihood probability? Give an example.
4. What is Naïve Bayes classifier? Why is it named so?
5. What is optimal Bayes classifier?
6. Write any two features of Bayesian learning methods.
7. Define the concept of consistent learners.
8. Write any two strengths of Bayes classifier.
9. Write any two weaknesses of Bayes classifier.
10. Explain how Naïve Bayes classifier is used for
    1. Text classification
    2. Spam filtering
    3. Market sentiment analysis

LONG ANSWER-TYPE QUESTIONS (10 MARKS EACH)

1. Explain the concept of Prior, Posterior, and Likelihood with an example.
2. How Bayes' theorem supports the concept learning principle?
3. Explain Naïve Bayes classifier with an example of its use in practical life.
4. Is it possible to use Naïve Bayes classifier for continuous numeric data? If so, how?
5. What are Bayesian Belief networks? Where are they used? Can they solve all types of problems?
6. In an airport security checking system, the passengers are checked to find out any intruder. Let I with $i \in \{0, 1\}$ be the random variable which indicates whether somebody is an intruder ($i = 1$) or not ($i = 0$) and A with $a \in \{0, 1\}$ be the variable indicating alarm. An alarm will be raised if an intruder is identified with probability $P(A = 1|I = 1) = 0.98$ and a non- intruder with probability $P(A = 1|I = 0) = 0.001$, which implies the error factor. In the population of passengers, the probability of someone is intruder is $P(I = 1) = 0.00001$. What is the probability that an alarm is raised when a person actually is an intruder?
7. An antibiotic resistance test (random variable T) has 1% false positives (i.e. 1% of those not

resistance to an antibiotic show positive result in the test) and 5% false negatives (i.e. 5% of those actually resistant to an antibiotic test negative). Let us assume that 2% of those tested are resistant to antibiotics. Determine the probability that somebody who tests positive is actually resistant (random variable *D*).

8. For preparation of the exam, a student knows that one question is to be solved in the exam which is either of types A, B, or C. The probabilities of A, B, or C appearing in the exam are 30%, 20%, and 50% respectively. During the preparation, the student solved 9 of 10 problems of type A, 2 of 10 problems of type B, and 6 of 10 problems of type C.
    1. What is the probability that the student will solve the problem of the exam?
    2. Given that the student solved the problem, what is the probability that it was of type A?
9. A CCTV is installed in a bank to monitor the incoming customers and take a photograph. Though there are continuous flows of customers, we create bins of timeframe of 5 min each. In each time frame of 5 min, there may be a customer moving into the bank with 5% probability or there is no customer (again, for simplicity, we assume that either there is 1 customer or none, not the case of multiple customers). If there is a customer, it will be detected by the CCTV with a probability of 99%. If there is no customer, the camera will take a false photograph by detecting other thing's movement with a probability of 10%.
    1. How many customers enter the bank on average per day (10 hours)?
    2. How many false photographs (there is a photograph taken even though there is no customer) and how many missed photographs (there is no photograph even though there is a customer) are there on average per day?
    3. If there is a photograph, what is the probability that there is indeed a customer?
10. Draw the Bayesian Belief network to represent the conditional independence assumptions of the Naïve Bayes classifier for the match winning prediction problem of Section 6.4.4. Construct the conditional probability table associated with the node Won Toss.

Supervised Learning: Classification

7.1 EXAMPLE OF SUPERVISED LEARNING

- In supervised learning, the labelled training data provide the basis for learning. According to the definition of machine learning, this labelled training data is the experience or prior knowledge or belief. It is called supervised learning because the process of learning from the training data by a machine can be related to a teacher supervising the learning process of a student who is new to the subject. Here, the teacher is the training data.
- Training data is the past information with known value of class field or '**label**'. Hence, we say that the '**training data is labelled**' in the case of supervised learning (refer Fig. 7.1). Contrary to this, there is no labelled training data for unsupervised learning. Semi-supervised learning, as depicted in Figure 7.1, uses a small amount of labelled data along with unlabelled data for training.

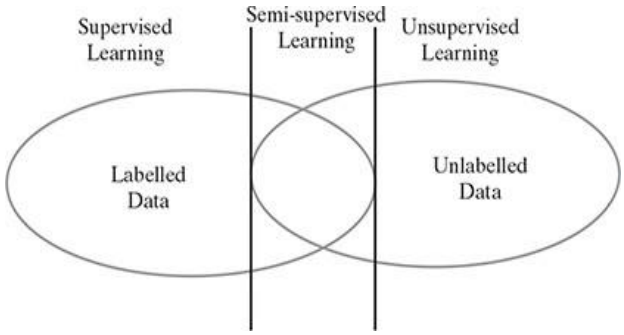


FIG. 7.1 Supervised learning vs. unsupervised learning

- In a hospital, many patients are treated in the general wards. In comparison, the number of beds in the Intensive Care Unit (ICU) is much less. So, it is always a cause of worry for the hospital management that if the health condition of a number of patients in the general ward suddenly aggravates and they would have to be moved to ICU. Without previous planning and preparations, such a spike in demand becomes difficult for the hospital to manage. This problem can be addressed in a much better way if it is possible to predict which of the patients in the normal wards have a possibility of their health condition deteriorating and thus need to be moved to ICU.
- This kind of prediction problem comes under the purview of supervised learning or, more specifically, under classification. The hospital already has all past patient records. The records of the patients whose health condition aggravated in the past and had to be moved to ICU can form the training data for this prediction problem. Test results of newly admitted patients are used to classify them as high-risk or low-risk patients.
- Some more examples of supervised learning are as follows:
  - Prediction of results of a game based on the past analysis of results
  - Predicting whether a tumour is malignant or benign on the basis of the analysis of data
  - Price prediction in domains such as real estate, stocks, etc.

7.2 CLASSIFICATION MODEL

- When we are trying to predict a categorical or nominal variable, the problem is known as a classification problem. A classification problem is one where the output variable is a category such as 'red' or 'blue' or 'malignant tumour' or 'benign tumour', etc.
- Whereas when we are trying to predict a numerical variable such as 'price', 'weight', etc. the problem falls under the category of regression.
- We can observe that in classification, the whole problem centres around assigning a class to a test data on the basis of the class information that is imparted by the training data. Because the target objective is to assign a class label, we call this type of problem as a classification problem.

- A critical classification problem in the context of the banking domain is identifying potentially fraudulent transactions. Because there are millions of transactionswhich have to be scrutinized to identify whether a particular transaction might be a fraud transaction, it is not possible for any human being to carry out this task. Machine learning is leveraged efficiently to do this task, and this is a classic case of classification. On the basis of the past transaction data, especially the ones labelled as fraudulent, all new incoming transactions are marked orlabelled as usual or suspicious. The suspicious transactions are subsequently segregated for a closer review.

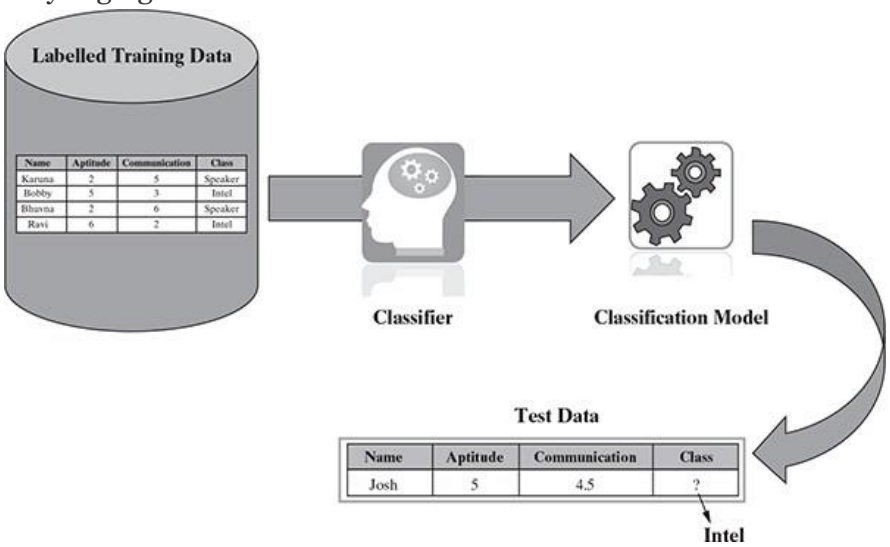


FIG. 7.2 Classification model

- Some typical classification problems include thefollowing:
  - Image classificationDisease prediction
  - Win–loss prediction of games
  - Prediction of natural calamity such as earthquake, flood, etc. Handwriting recognition

7.3: CLASSIFICATION LEARNING STEPS

First, there is a problem which is to be solved, and then, the required data (related to the problem, which is already stored in the system) is evaluated and pre- processed based on the algorithm. Algorithm selection is a critical point in supervised learning. The result after iterative training rounds is a classifier for the problem inhand (refer Fig. 7.3).

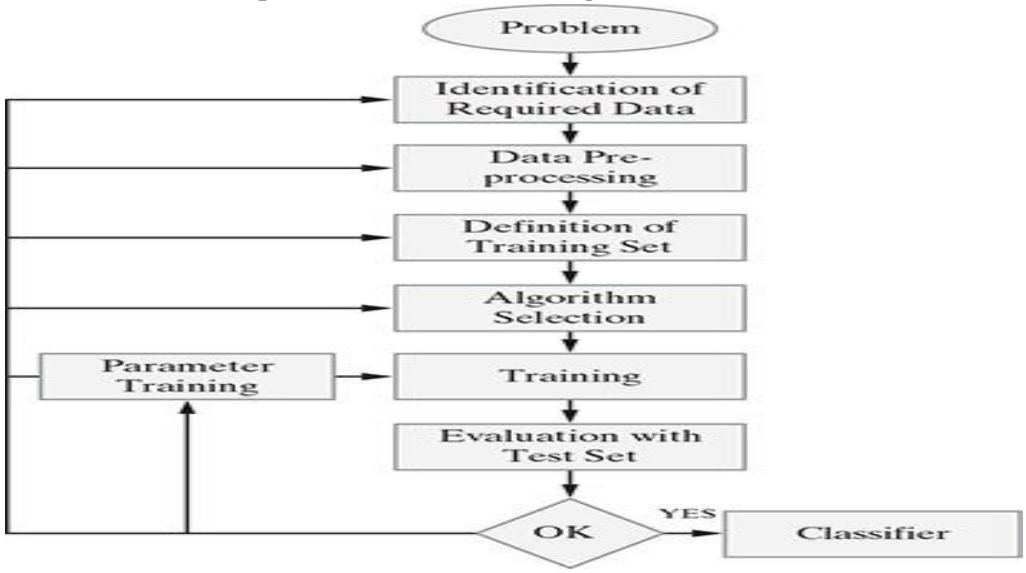


FIG. 7.3 Classification model steps

Problem Identification: Identifying the problem is the first step in the supervised learning model. The problem needs to be a well-formed problem,i.e. a problem with well-defined goals and benefit, which has along-term impact.

Identification of Required Data: On the basis of the problem identified above, the required data set that precisely represents the identified problem needs to be identified/evaluated. For example: If the problem is to predict whether a tumour is malignant or benign, then the corresponding patient data sets related to malignant tumour and benign tumours are to be identified.

Data Pre-processing: This is related to the cleaning/transforming the data set. This step ensures that all the unnecessary/irrelevant data elements are removed. Data pre-processing refers to the transformations applied to the identified data before feeding the same into the algorithm. Because the data isgathered from different sources, it is usually collected ina raw format and is not ready for immediate analysis.
This step ensures that the data is ready to be fed into themachine learning algorithm.

Definition of Training Data Set: Before starting the analysis, the user should decide what kind of data set is to be used as a training set. In the case of signature analysis, for example, the training data set might be a single handwritten alphabet, an entire handwritten word (i.e. a group of the alphabets) or an entire line of handwriting (i.e. sentences or a group of words). Thus, a set of 'input meta-objects' and corresponding 'output meta-objects' are also gathered. The training set needs to be actively representative of the real-world use of the given scenario. Thus, a set of data input (X) and corresponding outputs (Y) is gathered either from human experts or experiments.

Algorithm Selection: This involves determining the structure of the learning function and the correspondinglearning algorithm. This is the most critical step of supervised learning model. On the basis of various parameters, the best algorithm for a given problem is chosen.
Training: The learning algorithm identified in the previous step is run on the gathered training set for further fine tuning. Some supervised learning algorithmsrequire the user to determine specific control parameters(which are given as inputs to the algorithm). These parameters (inputs given to algorithm) may also be adjusted by optimizing performance on a subset (called as validation set) of the training set.

Evaluation with the Test Data Set: Training data isrun on the algorithm, and its performance is measured here. If a suitable result is not obtained, further training of parameters may be required.

7.4 COMMON CLASSIFICATION ALGORITHMS

- Following are the most common classification algorithms, out of which we have already learnt about the Naïve Bayes classifier in Chapter 6. We will cover details of the other algorithms in this chapter.
    - *k*-Nearest Neighbour (*k*NN)
    - Decision tree
    - Random forest
    - Support Vector Machine (SVM)
    - Naïve Bayes classifier

7.4.1 *k* -Nearest Neighbour (*k*NN)

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- Requires three things
    - The set of stored records
    - Distance Metric to compute distance between records

- The value of k, the number of nearest neighbors to retrieve

➢ To classify an unknown record:
- Compute distance to other training records
- Identify k nearest neighbors
- Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Algorithm:

Input: Training data set, test data set (or data points), value of '*k*' (i.e. number of nearest neighbours to be considered)

Steps:

Do for all test data points

Calculate the distance (usually Euclidean distance) of the test data point from the different training data points.

Find the closest '*k*' training data points, i.e. training data points whose distances are least from the test data point.

If $k = 1$

Then assign class label of the training data point to the test data point

Else

Whichever class label is mostly present in the training data points, assign that class label to the test data point y' = argmax $v\ I(\ v = yi\ )\ xi\ ,yi \in Dz$

End do

Compute distance between two points:

— Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

— Manhatten distance

$$d(p,q) = \sum_i |p_i - q_i|$$

— q norm distance

$$d(p,q) = (\Sigma_i |p_i - q_i|^q)^{1/q}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors

$$y' = \underset{v}{\text{argmax}} \sum_{(x_i,y_i) \in D_z} I(\ v = y_i\ )$$

where D_z is the set of k closest training examples to z.

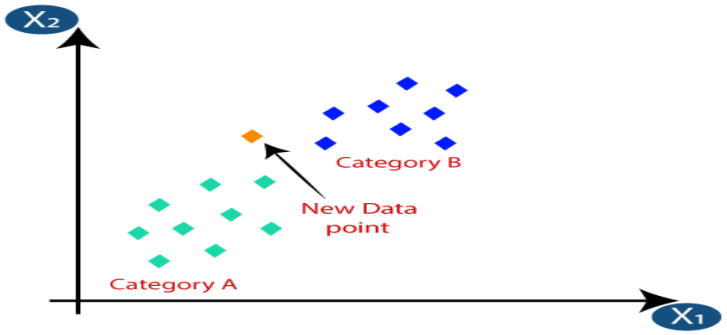
  - Weigh the vote according to distance

$$y' = \underset{v}{\text{argmax}} \sum_{(x_i,y_i) \in D_z} w_i \times I(\ v = y_i\ )$$

- weight factor, $w = 1/d^2$

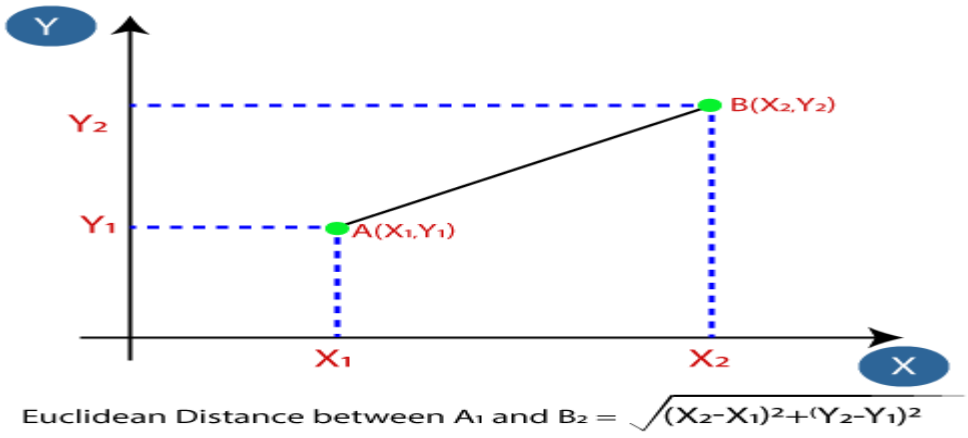
Example or How KNN Works:

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

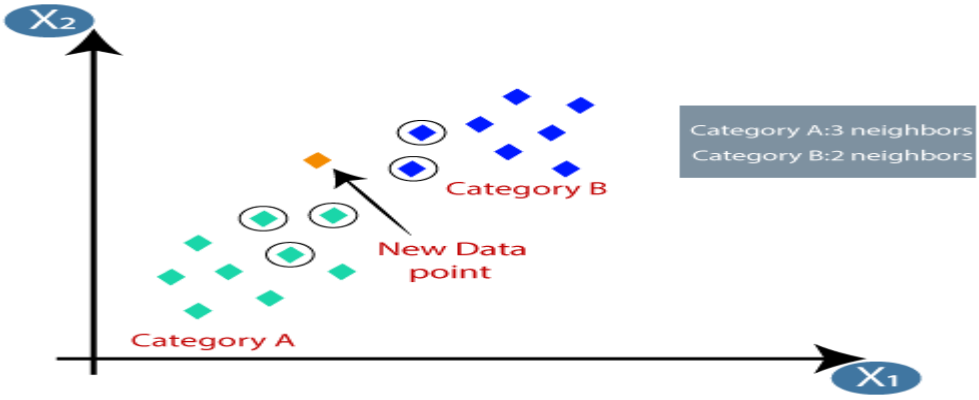


- Firstly, we will choose the number of neighbors, so we will choose the k=5.

- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

Why the kNN algorithm is called a lazy learner?

- Eager learners follow the general steps of machine learning, i.e. perform an abstraction of the information obtained from the input data and then follow it through by a generalization step.
- However, in the case of the kNN algorithm, these steps are completely skipped. It stores the training data and directly applies the philosophy of nearest neighbourhood finding to arrive at the classification. So, for kNN, there is no learning happening in the real sense. Therefore, kNN falls under the category of lazy learner.

Strengths of the kNN algorithm

- Extremely simple algorithm – easy to understand
- Very effective in certain situations, e.g. for recommender system design
- Very fast or almost no time required for the training phase

Weaknesses of the kNN algorithm

- Does not learn anything in the real sense. Classification is done completely on the basis of the training data. So, it has a heavy reliance on the training data. If the training data does not represent the problem domain comprehensively, the algorithm fails to make an effective classification.
- Because there is no model trained in real sense and the classification is done completely on the basis of the training data, the classification process is very slow.
- Also, a large amount of computational space is required to load the training data for classification.

Application of the kNN algorithm

- One of the most popular areas in machine learning where the kNN algorithm is widely adopted is recommender systems. As we know, recommender systems recommend users different items which are similar to a particular item that the user seems to like. The liking pattern may be revealed from past purchases or browsing history and the similar items are identified using the kNN algorithm.
- Another area where there is widespread adoption of kNN is searching documents/ contents similar to a given document/content. This is a core area under information retrieval and is known as concept search.

7.4.2 DECISION TREE

- Decision tree learning is one of the most widely adopted algorithms for classification. As the name indicates, it builds a model in the form of a tree structure.
- Its grouping exactness is focused with different strategies, and it is exceptionally productive. A decision tree is used for multi-dimensional analysis with multiple classes. It is characterized by fast execution time and ease in the interpretation of the rules. The goal of decision tree learning is to create a model predicts the value of the output variable based on the input variables in the feature vector.
- Each node of a decision tree corresponds to one of the feature vector. From every node, there are edges to children, wherein there is an edge for each of the possible values (or range of values) of the feature associated with the node. The tree terminates at different leaf nodes (or terminal nodes) where each leaf node represents a possible value for the output variable. The output variable is determined by following a path that starts at the root and is guided by the values of the input variables.

A decision tree is usually represented in the format depicted in Figure 7.8.

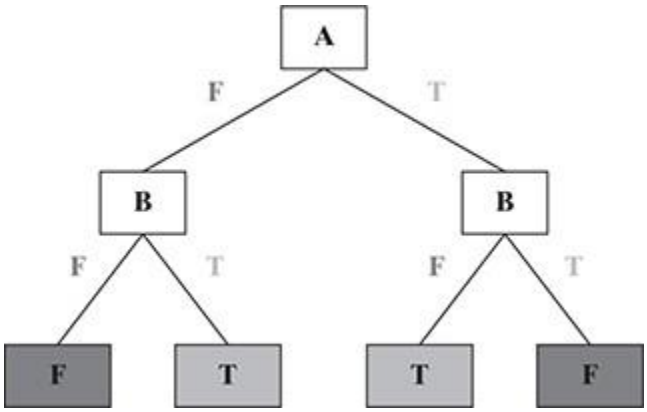


FIG. 7.8 Decision tree structure

- The first node is called as 'Root' Node. Branches from the root node are called as 'Leaf' Nodes where 'A' is the Root Node (first node). 'B' is the Branch Node. 'T' & 'F' are Leaf Nodes.
- Thus, a decision tree consists of three types of nodes:
  - Root Node
  - Branch Node
  - Leaf Node

- Figure 7.9 shows an example decision tree for a car driving – the decision to be taken is whether to 'Keep Going' or to 'Stop', which depends on various situations as depicted in the figure. If the signal is RED in colour, then the car should be stopped. If there is not enough gas(petrol) in the car, the car should be stopped at the next available gas station.

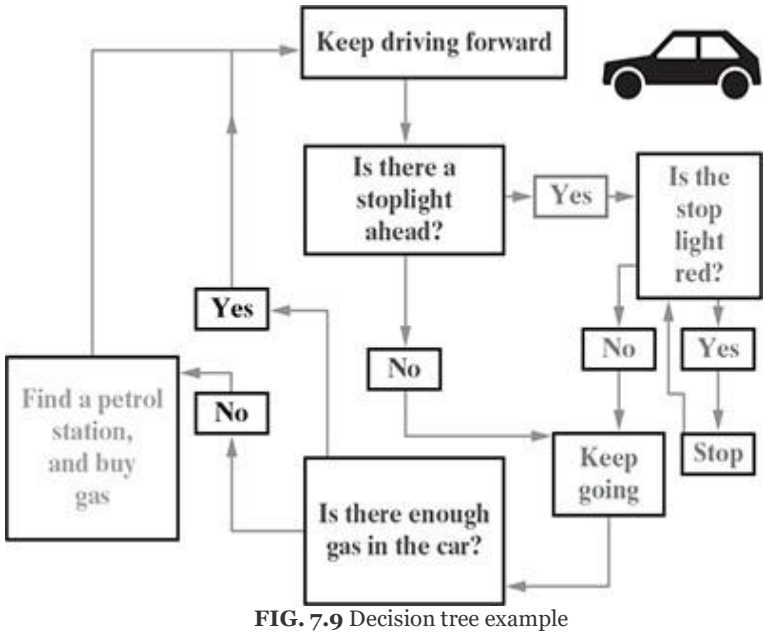


FIG. 7.9 Decision tree example

7.5.2.1 Building a decision tree

- Decision trees are built corresponding to the training data following an approach called recursive partitioning.
- The approach splits the data into multiple subsets on the basis of the feature values. It starts from the root node, which is nothing but the entire data set.
- It first selects the feature which predicts the target class in the strongest way. The decision tree splits the data set into multiple partitions, with data in each partition having a distinct value for the feature based on which the partitioning has happened. This is the first set of branches.
- Likewise, the algorithm continues splitting the nodes on the basis of the feature which helps in the best partition. This continues till a stopping criterion is reached. The usual stopping criteria are –
    - All or most of the examples at a particular node have the same class
    - All features have been used up in the partitioning
    - The tree has grown to a pre-defined threshold limit

Example:

- Let us try to understand this in the context of an example. Global Technology Solutions (GTS), a leading provider of IT solutions, is coming to College of Engineering and Management (CEM) for hiring B.Tech. students. Last year during campus recruitment, they had shortlisted 18 students for the final interview.
- Being a company of international repute, they follow a stringent interview process to select only the best of the students. The information related to the interview evaluation results of shortlisted students (hiding the names) on the basis of different evaluation parameters is available for reference in Figure 7.10.
- Chandra, a student of CEM, wants to find out if he may be offered a job in GTS. His CGPA is quite high. His self-evaluation on the other parameters is as follows: **Communication – Bad; Aptitude – High; Programming skills – Bad**

CGPA	Communication	Aptitude	Programming Skill	Job offered?
High	Good	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	Low	Good	No
Low	Good	Low	Bad	No
High	Good	High	Bad	Yes
High	Good	High	Good	Yes
Medium	Bad	Low	Bad	No
Medium	Bad	Low	Good	No
High	Bad	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	High	Bad	No
Low	Bad	High	Bad	No
Medium	Good	High	Bad	Yes
Low	Good	Low	Good	No
High	Bad	Low	Bad	No
Medium	Bad	High	Good	No
High	Bad	Low	Bad	No
Medium	Good	High	Bad	Yes

FIG. 7.10 Training data for GTS recruitment

- Let us try to solve this problem, i.e. predicting whether Chandra will get a job offer, by using the decision tree model. First, we need to draw the decision tree corresponding to the training data given in Figure 7.10. According to the table, job offer condition (i.e. the outcome) is FALSE for all the cases where **Aptitude = Low**, irrespective of other conditions. So, the feature Aptitude can be taken up as the first node of the decision tree.
- For **Aptitude = High**, job offer condition is TRUE for all the cases where **Communication = Good**. For cases where **Communication = Bad**, job offer condition is TRUE for all the cases where **CGPA = High**.

Figure 7.11 depicts the complete decision tree diagram for the table given in Figure 7.10.

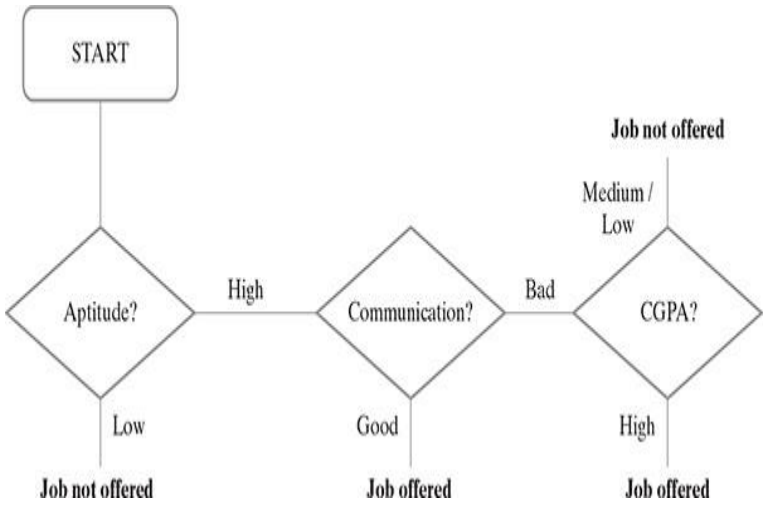


FIG. 7.11 Decision tree based on the training data

7.5.2.2 Searching a decision tree

- By using the above decision tree depicted in Figure 7.11, we need to predict whether Chandra might get a job offer for the given parameter values: CGPA = **High**, Communication = **Bad**, Aptitude = **High**, Programming skills = **Bad**. There are multiple ways to search through the trained decision tree for a solution to the given prediction problem.

Exhaustive search

1. Place the item in the first group (class). Recursively examine solutions with the item in the first group (class).
2. Place the item in the second group (class). Recursively examine solutions with the item in the second group (class).
3. Repeat the above steps until the solution is reached.

- Exhaustive search travels through the decision tree exhaustively, but it will take much time when the decision tree is big with multiple leaves and multiple attribute values.

Branch and bound search

- Branch and bound uses an existing best solution to sidestep searching of the entire decision tree in full.
- When the algorithm starts, the best solution is well defined to have the worst possible value; thus, any solution it finds out is an improvement. This makes the algorithm initially run down to the left-most branch of the tree, even though that is unlikely to produce a realistic result.
- In the partitioning problem, that solution corresponds to putting every item in one group, and it is an unacceptable solution. A programme can speed up the process by using a fast heuristic to find an initial solution. This can be used as an input for branch and bound. If the heuristic is right, the savings can be substantial.

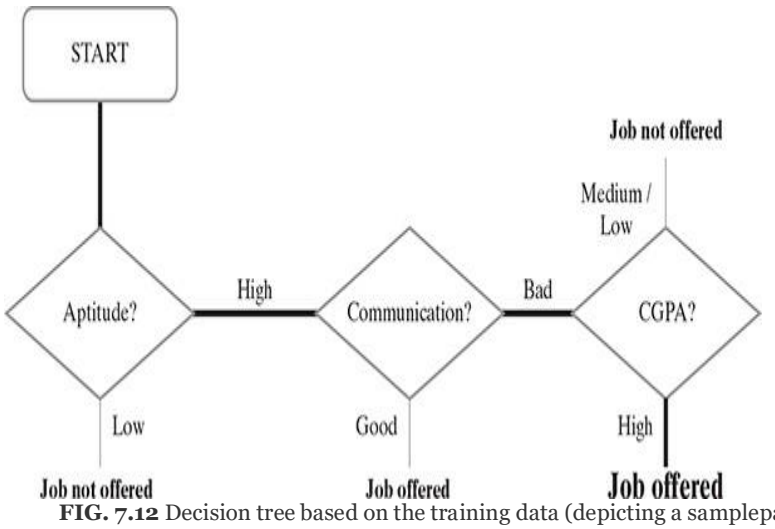


FIG. 7.12 Decision tree based on the training data (depicting a samplepath)

Figure 7.12 depicts a sample path (thick line) for theconditions CGPA = **High**, Communication = Bad, Aptitude = High and Programming skills = Bad. According to the above decision tree, the prediction canbe made as Chandra **will get the job offer**.

- There are many implementations of decision tree, the most prominent ones being C5.0, CART (Classification and Regression Tree), CHAID (Chi-square Automatic Interaction Detector) and ID3 (Iterative Dichotomiser 3)algorithms.
- The biggest challenge of a decision tree algorithm is to find out which feature to split upon. The main driver for identifying the feature is that the data should be split in such a way that the partitions created by the split should contain examples belonging to a single class. If that happens, the partitions are considered to be **pure**.
- **Entropy is a measure of impurity of an attribute or feature adopted by many algorithms such as ID3 and C5.0. The information gain is calculated on the basis of the decrease in entropy (S) after a data set is split according to a particular attribute (A).**
- Constructing a decision tree is all about finding an attribute that returns the highest information gain (i.e.the most homogeneous branches).

7.5.2.3 Entropy of a decision tree

- Let us say S is the sample set of training examples. Then, Entropy (S) measuring the impurity of S is defined as

$$\textbf{Entropy}(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

  - where c is the number of different class labels and prefers to the proportion of values falling into the *i*-th class label.
- For example, with respect to the training data in Figure 7.10, we have two values for the target class 'Job Offered?' – Yes and No. The value of p_i for class value 'Yes' is 0.44 (i.e. 8/18) and that for class value 'No' is 0.56 (i.e. 10/18). So, we can calculate the entropy as

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)
Entropy(S) = -0.44 log $_2$(0.44) - 0.56 log $_2$(0.56) = 0.99.

7.5.2.4 Information gain of a decision tree

- The information gain is created on the basis of the decrease in entropy (S) after a data set is split according to a particular attribute (A).
- Constructing a decision tree is all about finding an attribute that returns the highest information gain. If the information gain is 0, it means that there is no reduction in entropy due to split of the data set accordingto that particular feature.
- On the other hand, the maximum amount of information gain which may happen is the entropy of the data set before the split.
- Information gain for a particular feature A is calculated by the difference in entropy before a split (S_{bs}) with the entropy after the split (S_{as}).

Information Gain (S, A) = Entropy (S_{bs}) − Entropy (S_{as})

- For calculating the entropy after split, entropy for all partitions needs to be considered. Then, the weighted summation of the entropy for each partition can be taken as the total entropy after split. For performing weighted summation, the proportion of examples falling into each partition is used as weight.

$$\textbf{Entropy}(S_{as}) = \sum_{i=1}^{n} w_i \text{ Entropy } (p_i)$$

Let us examine the value of information gain for the training data set shown in Figure 7.10. We will find the value of entropy at the beginning before any split happens and then again after the split happens. We willcompare the values for all the cases –

1. when the feature 'CGPA' is used for the split
2. when the feature 'Communication' is used for the split
3. when the feature 'Aptitude' is used for the split
4. when the feature 'Programming Skills' is used for the split

Figure 7.13a gives the entropy values for the first levelsplit for each of the cases mentioned above.

As calculated, entropy of the data set before split (i.e.Entropy (S_{bs})) = 0.99, and entropy of the data set after split (i.e. Entropy (S_{as})) is

- 0.69 when the feature 'CGPA' is used for split
- 0.63 when the feature 'Communication' is used for split
- 0.52 when the feature 'Aptitude' is used for split
- 0.95 when the feature 'Programming skill' is used for split

(a) Original data set:

	Yes	No	Total
Count	8	10	18
pi	0.44	0.56	
-pi*log(pi)	0.52	0.47	0.99

Total Entropy = 0.99

(b) Splitted data set (based on the feature 'CGPA'):

CGPA = High

	Yes	No	Total
Count	4	2	6
pi	0.67	0.33	
-pi*log(pi)	0.39	0.53	0.92

Total Entropy = 0.69

CGPA = Medium

	Yes	No	Total
Count	4	3	7
pi	0.57	0.43	
-pi*log(pi)	0.46	0.52	0.99

Information Gain = 0.30

CGPA = Low

	Yes	No	Total
Count	0	5	5
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

(c) Splitted data set (based on the feature 'Communication'):

Communication = Good

	Yes	No	Total
Count	7	2	9
pi	0.78	0.22	
-pi*log(pi)	0.28	0.48	0.76

Total Entropy = 0.63

Communication = Bad

	Yes	No	Total
Count	1	8	9
pi	0.11	0.89	
-pi*log(pi)	0.35	0.15	0.50

Information Gain = 0.36

(d) Splitted data set (based on the feature 'Aptitude'):

Aptitude = High

	Yes	No	Total
Count	8	3	11
pi	0.73	0.27	
-pi*log(pi)	0.33	0.51	0.85

Total Entropy = 0.52

Aptitude = Low

	Yes	No	Total
Count	0	7	7
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Information Gain = 0.47

(e) Splitted data set (based on the feature 'Programming Skill'):

Programming Skill = Good

	Yes	No	Total
Count	5	4	9
pi	0.56	0.44	
-pi*log(pi)	0.47	0.52	0.99

Total Entropy = 0.95

Programming Skill = Bad

	Yes	No	Total
Count	3	6	9
pi	0.33	0.67	
-pi*log(pi)	0.53	0.39	092

Information Gain = 0.04

FIG. 7.13A Entropy and information gain calculation (Level 1)

Therefore, the information gain from the feature 'CGPA' = 0.99 – 0.69 = 0.3, whereas the information gain from the feature 'Communication' = 0.99 – 0.63 = 0.36. Likewise, the information gain for 'Aptitude' and 'Programming skills' is 0.47 and 0.04, respectively.

Hence, it is quite evident that among all the features, 'Aptitude' results in the best information gain when adopted for the split. So, at the first level, a split will be applied according to the value of 'Aptitude' or in other words, 'Aptitude' will be the first node of the decision tree formed. One important point to be noted here is that for **Aptitude = Low**, entropy is 0, which indicates that always the result will be the same irrespective of the values of the other features. Hence, the branch towards Aptitude = Low will not continue any further.

As a part of level 2, we will thus have only one branch to navigate in this case – the one for **Aptitude = High**. Figure 7.13b presents calculations for level 2. As can be seen from the figure, the entropy value is as follows:

- 0.85 before the split
- 0.33 when the feature 'CGPA' is used for split
- 0.30 when the feature 'Communication' is used for split
- 0.80 when the feature 'Programming skill' is used for split

Hence, the information gain after split with the features CGPA, Communication and Programming Skill is 0.52, 0.55 and 0.05, respectively. Hence, the feature Communication should be used for this split as it results in the highest information gain. So, at the second level, a split will be applied on the basis of the value of 'Communication'. Again, the point to be noted here is that for **Communication = Good**, entropy is 0, which indicates that always the result will be the same irrespective of the values of the other features. Hence, the branch towards Communication = Good

will notcontinue any further.

Aptitude = High

CGPA	Communication	Programming Skill	Job offered?
High	Good	Good	Yes
Medium	Good	Good	Yes
High	Good	Bad	Yes
High	Good	Good	Yes
High	Bad	Good	Yes
Medium	Good	Good	Yes
Low	Bad	Bad	No
Low	Bad	Bad	No
Medium	Good	Bad	Yes
Medium	Bad	Good	No
Medium	Good	Bad	Yes

(a) Level 2 starting set:

	Yes	No	Total
Count	8	3	11
pi	0.73	0.27	
-pi*log(pi)	0.33	0.51	0.85

Total Entropy = 0.85

(b) Splitted data set (based on the feature 'CGPA'):

CGPA = High

	Yes	No	Total
Count	4	0	4
pi	1.00	0.00	
-pi*log(pi)	0.00	0.00	0.00

Total Entropy = 0.33

CGPA = Medium

	Yes	No	Total
Count	4	1	5
pi	0.80	0.20	
-pi*log(pi)	0.26	0.46	0.72

Information Gain = 0.52

CGPA = Low

	Yes	No	Total
Count	0	2	2
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

(c) Splitted data set (based on the feature 'Communication'):

Communication = Good

	Yes	No	Total
Count	7	0	7
pi	1.00	0.00	
-pi*log(pi)	0.00	0.00	0.00

Total Entropy = 0.30

Communication = Bad

	Yes	No	Total
Count	1	3	4
pi	0.25	0.75	
-pi*log(pi)	0.50	0.31	0.81

Information Gain = 0.55

(d) Spitted data set (based on the feature 'Programming Skill'):

Programming Skill = Good

	Yes	No	Total
Count	5	1	6
pi	0.83	0.17	
-pi*log(pi)	0.22	0.43	0.65

Total Entropy = 0.80

Programming Skill = Bad

	Yes	No	Total
Count	3	2	5
pi	0.60	0.40	
-pi*log(pi)	0.44	0.53	0.97

Information Gain = 0.05

FIG. 7.13B Entropy and information gain calculation (Level 2)

As a part of level 3, we will thus have only one branch to navigate in this case – the one for **Communication = Bad**. Figure 7.13c presents calculations for level 3. As can be seen from the figure, the entropy value is as follows:

- 0.81 before the split
- 0 when the feature 'CGPA' is used for split
- 0.50 when the feature 'Programming Skill' is used for split

Aptitude = High & Communication = Bad

CGPA	Programming Skill	Job offered?
High	Good	Yes
Low	Bad	No
Low	Bad	No
Medium	Good	No

(a) Level 2 starting set:

	Yes	No	Total
Count	1	3	4
pi	0.25	0.75	
-pi*log(pi)	0.50	0.31	0.81

Total Entropy = 0.81

(b) Splitted data set (based on the feature 'CGPA'):

CGPA = High

	Yes	No	Total
Count	1	0	1
pi	1.00	0.00	
-pi*log(pi)	0.00	0.00	0.00

Total Entropy = 0.00

CGPA = Medium

	Yes	No	Total
Count	0	1	1
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Information Gain = 0.81

CGPA = Low

	Yes	No	Total
Count	0	2	2
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

(c) Splitted data set (based on the feature 'Programming Skill'):

Programming Skill = Good

	Yes	No	Total
Count	1	1	2
pi	0.50	0.50	
-pi*log(pi)	0.50	0.50	1.00

Total Entropy = 0.50

Programming Skill = Bad

	Yes	No	Total
Count	0	2	2
pi	0.00	1.00	
-pi*log(pi)	0.00	0.00	0.00

Information Gain = 0.31

FIG. 7.13C Entropy and information gain calculation (Level 3)

Hence, the information gain after split with the feature CGPA is 0.81, which is the maximum possible information gain (as the entropy before split was 0.81). Hence, as obvious, a split will be applied on the basis of the value of 'CGPA'. Because the maximum information gain is already achieved, the tree will not continue any further.

7.5.2.5 Algorithm for decision tree

Input: Training data set, test data set (or data points)

Steps:

Do for all attributes

Calculate the entropy E_i of the attribute F_i

if $E_i < E_{min}$

then $E_{min} = E_i$ and $F_{min} = F_i$

end if End do

Split the data set into subsets using the attribute F_{min}

Draw a decision tree node containing the attribute F_{min} and split the data set into subsets

Repeat the above steps until the full tree is drawn covering all the attributes of the original table.

7.5.2.6 Avoiding overfitting in decision tree – pruning

- The decision tree algorithm, unless a stopping criterion is applied, may keep growing indefinitely – splitting for every feature and dividing into smaller partitions till the point that

the data is perfectly classified. This, as is quiteevident, results in overfitting problem.

- To prevent a decision tree getting overfitted to the training data, pruning of the decision tree is essential.
- **Pruning a decision tree reduces the size of the tree such that the model is more generalized and can classify unknown andunlabelled data in a better way.**
- There are two approaches of pruning:
  - Pre-pruning: Stop growing the tree before it reaches perfection.
  - Post-pruning: Allow the tree to grow entirely and then post-prune some of the branches from it.
- In the case of pre-pruning, the tree is stopped from further growing once it reaches a certain number of decision nodes or decisions. Hence, in this strategy, the algorithm avoids overfitting as well as optimizes computational cost. However, it also stands a chance toignore important information contributed by a feature which was skipped, thereby resulting in miss out ofcertain patterns in the data.
- On the other hand, in the case of post-pruning, the tree is allowed to grow to the full extent. Then, by using certain pruning criterion, e.g. error rates at the nodes, the size of the tree is reduced. This is a more effective approach in terms of classification accuracy as it considers all minute information available from the training data. However, the computational cost is obviously more than that of pre-pruning.

7.5.2.7 Strengths of decision tree

- It produces very simple understandable rules. For smaller trees, not much mathematical and computational knowledge is required to understand this model.
- Works well for most of the problems.
- It can handle both numerical and categorical variables. Can work well both with small and large training data sets.
- Decision trees provide a definite clue of which features are moreuseful for classification.

7.5.2.8 Weaknesses of decision tree

- Decision tree models are often biased towards features having more number of possible values, i.e. levels.
- This model gets overfitted or underfitted quite easily.
- Decision trees are prone to errors in classification problems with many classes and relatively small number of training examples. A decision tree can be computationally expensive to train.
- Large trees are complex to understand.

7.5.2.9 Application of decision tree

- Decision tree can be applied in a data set in which there is a finite list of attributes and each data instance stores a value for that attribute (e.g. 'High' for the attribute CGPA). When each attribute has a small number of distinct values (e.g. 'High', 'Medium', 'Low'), it is easier/quicker for the decision tree to suggest (or choose) an effective solution. This algorithm can be extended to handle real-value attributes (e.g. a floating point temperature).
- The most straightforward case exists when there are only two possible values for an attribute (Boolean classification). Example: Communication has only two values as 'Good' or 'Bad'. It is also easy to extend the decision tree to create a target function with more than two possible output values. Example: CGPA can take one of the values from 'High', 'Medium', and 'Low'. Irrespective of whether it is a binary value/ multiple values, it is discrete in nature. For example, Aptitude cantake the value of either 'High' or 'Low'. It is not possible to assign the value of both 'High' and 'Low' to the attribute Aptitude to draw a decision tree.
- There should be no infinite loops on taking a decision. As we move from the root node to the next level node, it should move step-by-step towards the decision node.
- Otherwise, the algorithm may not give the final result fora given data. If a set of code goes in a loop, it would repeat itself forever, unless the system crashes.
- A decision tree can be used even for some instances with missing attributes and instances

with errors in the classification of examples or in the attribute values describing those examples; such instances are handled well by decision trees, thereby making them a robust learning method.

7.4.3 Random forest model

- Random forest is an ensemble classifier, i.e. a combining classifier that uses and combines many decision tree classifiers.
- Ensembling is usually done using the concept of bagging with different feature sets. The reason for using large number of trees in random forest is to train the trees enough such that contribution from each feature comes in a number of models.
- After the random forest is generated by combining the trees, majority vote is applied to combine the output of the different trees. A simplified random forest model is depicted in Figure 7.14. The result from the ensemble model is usually better than that from the individual decision tree models.

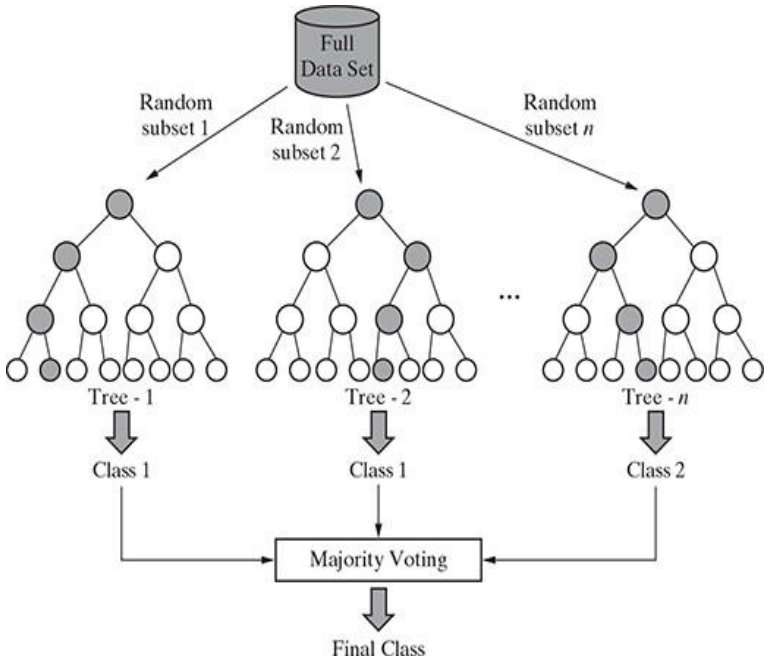


FIG. 7.14 Random forest model

How does random forest work?

The random forest algorithm works as follows:

1. If there are N variables or features in the input data set, select a subset of 'm' (m < N) features at random out of the N features. Also, the observations or data instances should be picked randomly.
2. Use the best split principle on these 'm' features to calculate the number of nodes 'd'.
3. Keep splitting the nodes to child nodes till the tree is grown to the maximum possible extent.
4. Select a different subset of the training data 'with replacement' to train another decision tree following steps (1) to (3). Repeat this to build and train 'n' decision trees.
5. Final class assignment is done on the basis of the majority votes from the 'n' trees

Out-of-bag (OOB) error in random forest

- In random forests, we have seen, that each tree is constructed using a different bootstrap sample from the original data. The samples left out of the bootstrap and not used in the construction of the i-th tree can be used to measure the performance of the model. At the end of the run, predictions for each such sample evaluated each time are tallied, and the final prediction for that sample is obtained by taking a vote. The total error rate of predictions for such samples is termed as out-of-bag (OOB) error rate.
- The error rate shown in the confusion matrix reflects the OOB error rate. Because of this

reason, the error ratedisplayed is often surprisingly high.

Strengths of random forest

- It runs efficiently on large and expansive data sets.
- It has a robust method for estimating missing data and maintains precision when a large proportion of the data is absent.
- It has powerful techniques for balancing errors in a class populationof unbalanced data sets.
- It gives estimates (or assessments) about which features are the most important ones in the overall classification.
- It generates an internal unbiased estimate (gauge) of the generalization error as the forest generation progresses. Generated forests can be saved for future use on other data. Lastly, the random forest algorithm can be used to solve bothclassification and regression problems.

Weaknesses of random forest

- This model, because it combines a number of decision tree models, isnot as easy to understand as a decision tree model.
- It is computationally much more expensive than a simple model likedecision tree.

Application of random forest

- Random forest is a very powerful classifier which combines the versatility of many decision tree models into a single model. Because of the superior results, this ensemble model is gaining wide adoption and popularity amongst the machine learning practitioners to solve a wide range of classification problems.

7.4.4 SUPPORT VECTOR MACHINES

- SVM is a model, which can do linear classification as wellas regression. SVM is based on the concept of a surface, called a hyperplane, which draws a boundary between data instances plotted in the multi-dimensional feature space.
- The output prediction of an SVM is one of two conceivable classes which are already defined in the training data. In summary, the SVM algorithm builds an N-dimensional hyperplane model that assigns future instances into one of the two possible output classes.

Classification using hyperplanes

- In SVM, a model is built to discriminate the data instances belonging to different classes. Let us assume for the sake of simplicity that the data instances are linearly separable.
- In this case, when mapped in a two- dimensional space, the data instances belonging to different classes fall in different sides of a straight line drawn in the two-dimensional space as depicted in Figure 7.15a. If the same concept is extended to a multi- dimensional feature space, the straight line dividing datainstances belonging to different classes transforms to a hyperplane as depicted in Figure 7.15b.

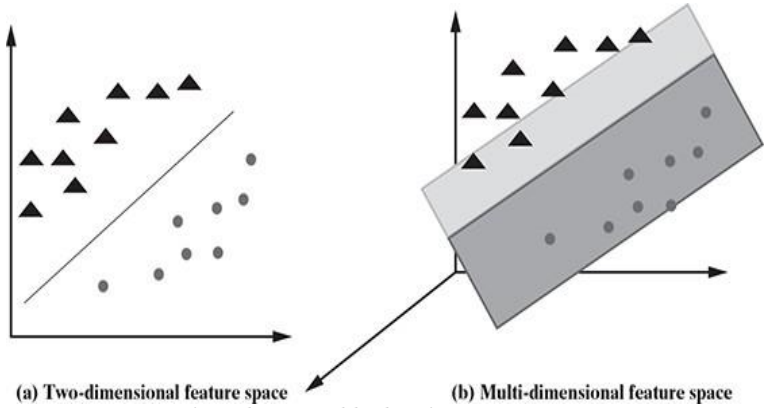


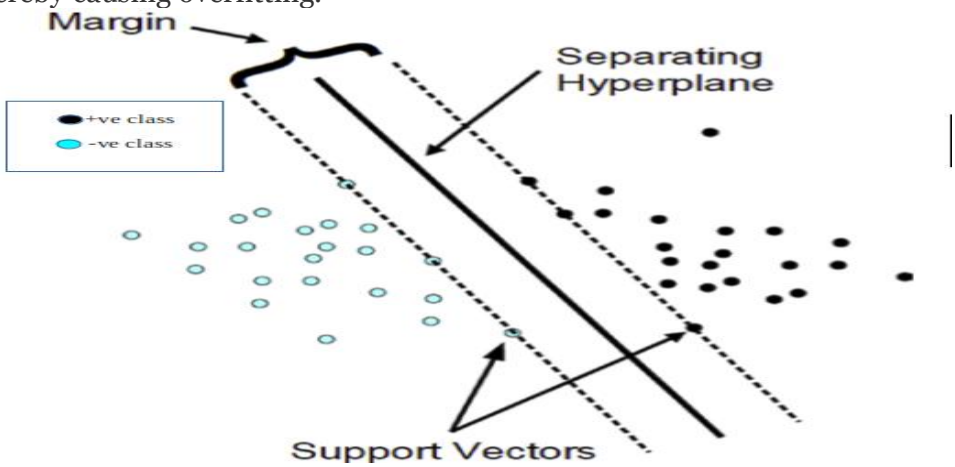
FIG. 7.15 Linearly separable data instances

- Thus, an SVM model is a representation of the input instances as points in the feature space, which are mapped so that an apparent gap between them divides the instances of the separate classes.
- In other words, the goal of the SVM analysis is to find a hyperplane, which separates the instances on the basis oftheir classes.
- New instances are then mapped into that same space and predicted to belong to a class on

the basis of which side of the gap the new instance will fall on.

- In summary, in the overall training process, the SVM algorithm analyses input data and identifies a surface in the multi-dimensional feature space called the hyperplane. There may be many possible hyperplanes, and **one of the challenges with the SVM model is to find the optimal hyperplane.**

Training data sets which have a substantial grouping periphery will function well with SVM. Generalization error in terms of SVM is the measure of how accurately and precisely this SVM model can predict values for previously unseen data (new data). A hard margin in terms of SVM means that an SVM model is inflexible in classification and tries to work exceptionally fit in the training set, thereby causing overfitting.



- **Support Vectors:** Support vectors are the data points (representing classes), the critical component in a data set, which are near the identified set of lines (hyperplane). If support vectors are removed, they will alter the position of the dividing hyperplane.
- **Hyperplane and Margin:**
  - SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. And the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the optimal hyperplane.

7.5.4.1 Identifying the correct hyperplane in SVM

- As we have already discussed, there may be multiple options for hyperplanes dividing the data instances belonging to the different classes. We need to identify which one will result in the best classification.
- Let us examine a few scenarios before arriving to that conclusion. For the sake of simplicity of visualization, thehyperplanes have been shown as straight lines in most of the diagrams.

Scenario 1

- As depicted in Figure 7.16, in this scenario, we have threehyperplanes: A, B, and C. Now, we need to identify the correct hyperplane which better segregates the two classes represented by the triangles and circles. As we can see, hyperplane 'A' has performed this task quite well.

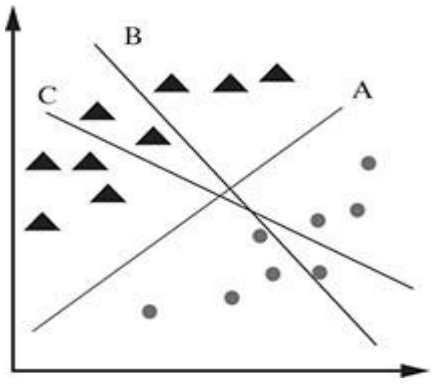


FIG. 7.16 Support vector machine: Scenario 1

Scenario 2

- As depicted in Figure 7.17, we have three hyperplanes: A, B, and C. We have to identify the correct hyperplane which classifies the triangles and circles in the best possible way. Here, maximizing the distances between the nearest data points of both the classes and hyperplane will help us decide the correct hyperplane. This distance is called as **margin**.
- In Figure 7.17b, you can see that the margin for hyperplane A is high as compared to those for both B and C. Hence, hyperplane A is the correct hyperplane. Another quick reason for selecting the hyperplane with higher margin (distance) is robustness. **If we select a hyperplane having a lower margin (distance), then thereis a high probability of misclassification.**

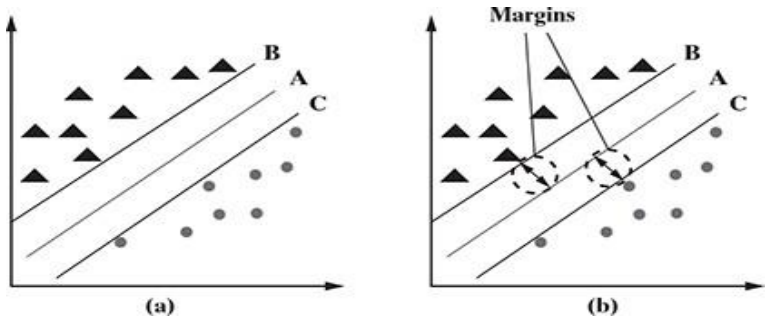


FIG. 7.17 Support vector machine: Scenario 2

Scenario 3

- Use the rules as discussed in the previous section to identify the correct hyperplane in the scenario shown in Figure 7.18. Some of you might have selected hyperplane B as it has a higher margin (distance from the class) than A. But, here is the catch; SVM selects the hyperplane which classifies the classes accurately before maximizing the margin. Here, hyperplane B has a classification error, and A has classified all data instances correctly. Therefore, A is the correct hyperplane.

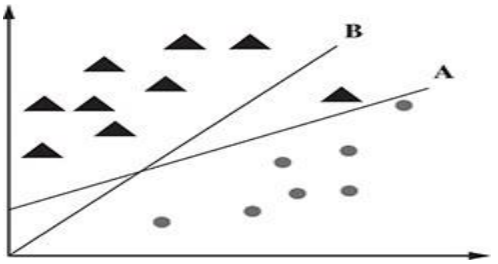


FIG. 7.18 Support vector machine: Scenario 3

Scenario 4

- In this scenario, as shown in Figure 7.19a, it is not possible to distinctly segregate the two classes by using astraight line, as one data instance belonging to one of the classes (triangle) lies in the territory of theother class (circle) as an outlier.
- One triangle at the other end is like an outlier for thetriangle class. SVM has a feature to ignore outliers and find the hyperplane that has the maximum margin (hyperplane A, as shown in Fig. 7.19b). Hence, we can say that SVM is robust to outliers.

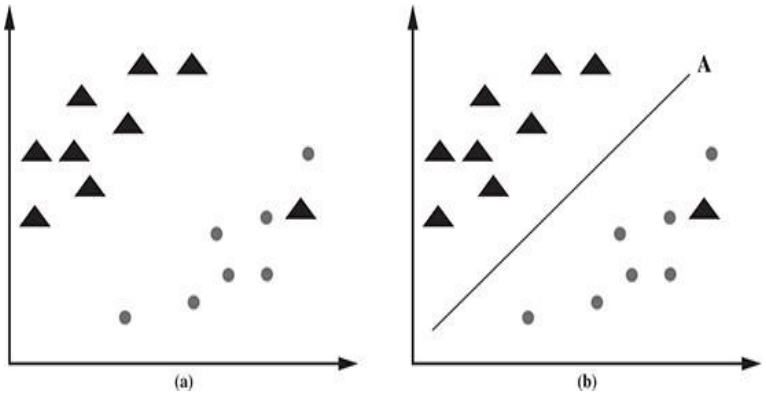


FIG. 7.19 Support vector machine: Scenario 4

- So, by summarizing the observations from thedifferent scenarios, we can say that
  a. The hyperplane should segregate the data instances belonging to the two classes in the best possible way.
  b. It should maximize the distances between the nearest data points of both the classes, i.e. maximize the margin.
  c. If there is a need to prioritize between higher margin and lesser misclassification, the hyperplane should try to reduce misclassifications.

7.5.4.2 Maximum margin hyperplane

- Finding the Maximum Margin Hyperplane (MMH) is nothing but identifying the hyperplane which has the largest separation with the data instances of the two classes.
- Though any set of three hyperplanes can do thecorrect classification, why do we need to search for the set of hyperplanes causing the largest separation? The answer is that doing so helps us in achieving more generalization and hence less number of issues in the classification of unknown data.

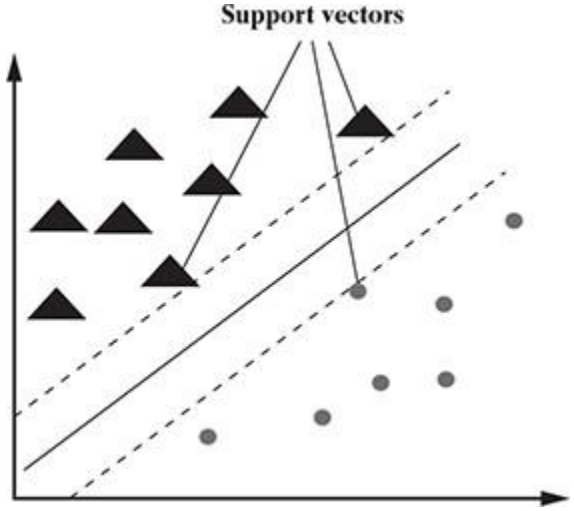


FIG. 7.20 Support vectors

- **Support vectors**, as can be observed in Figure 7.20, are data instances from the two classes which are closest to the MMH. Quite understandably, there should be at least one support vector from each class.

Identifying the MMH for linearly separable data

- Finding out the MMH is relatively straightforward for the data that is linearly separable. In this case, an outer boundary needs to be drawn for the data instances belonging to the different classes. These outer boundaries are known as convex hull, as depicted in Figure 7.21. The MMH can be drawn as the perpendicular bisector of the shortest line (i.e. the connecting line having the shortest length) between theconvex hulls.

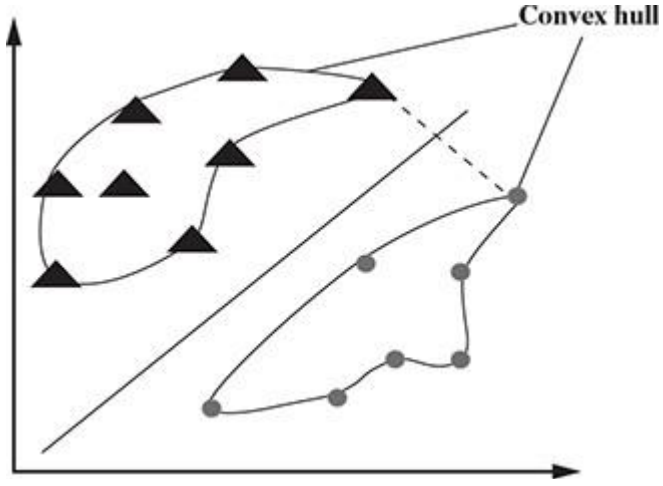


FIG. 7.21 Drawing the MMH for linearly separable data

- N-dimensional feature space can be represented by the equation: $\vec{c}.\vec{X} + c_0 = 0$
- Using this equation, the objective is to find a set of values for the vector $\vec{c}$ such that two hyperplanes, represented by the equations below, can be specified.

$$\vec{c}.\vec{X} + c_0 \geq +1$$
$$\vec{c}.\vec{X} + c_0 \leq -1$$

This is to ensure that all the data instances that belong to one class falls above one hyperplane and all the data instances belonging to the other class falls below another hyperplane.

Identifying the MMH for non-linearly separable data

Now that we have a clear understanding of how to identify the MMH for a linearly separable data set, let us do some study about how non-linearly separable data needs to be handled by SVM. For this, we have to use a slack variable ξ, which provides some soft margin for data instances in one class that fall on the wrong side of the hyperplane. As depicted in Figure 7.22, a data instance belonging to the circle class falls on the side of the hyperplane designated for the data instances belonging to the triangle class. The same issue also happens for a data instance belonging to the triangle class.

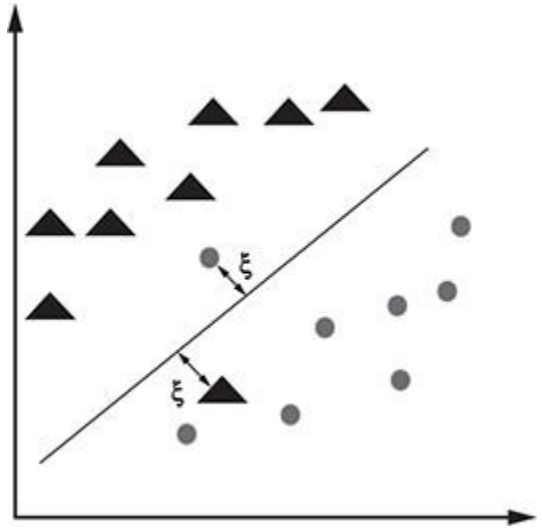


FIG. 7.22 Drawing the MMH for non-linearly separable data

A cost value '*C*' is imposed on all such data instances that fall on the wrong side of the hyperplane. The task of SVM is now to minimize the total cost due to such data instances in order to solve the revised optimization problem:

$$\min\left(\frac{1}{2}\vec{c}^2\right) + C\sum_{i=1}^{N}\xi_i$$

7.5.4.3 Kernel trick

- SVM has a technique called the **kernel trick** to deal with non-linearly separable data. As shown in Figure 7.23, these are functions which can transform lower dimensional input space to a higher dimensional space. In the process, it converts linearly non-separable data to a linearly separable data. These functions are called **kernels**.

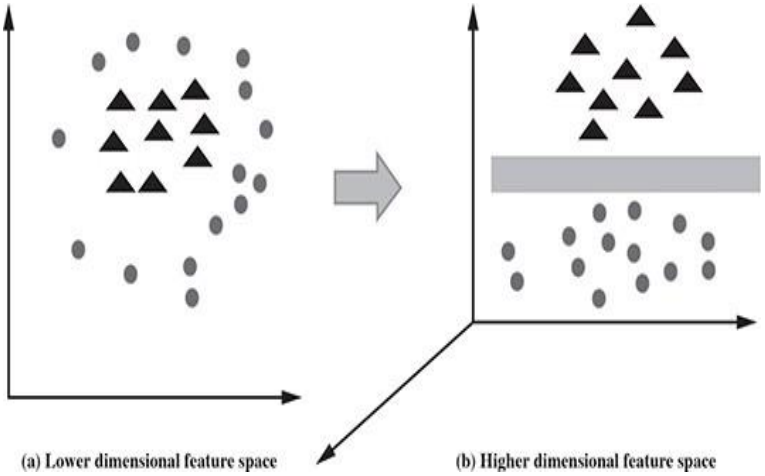


FIG. 7.23 Kernel trick in SVM

- Some of the common kernel functions for transforming from a lower dimension '*i*' to a higher dimension '*j*' used by different SVM implementations areas follows:
  - Linear kernel: It is in the form $K(\vec{x_i}, \vec{x_j}) = \vec{x_i}.\vec{x_j}$
  - Polynomial kernel: It is in the form
$$K(\vec{x_i}, \vec{x_j}) = (\vec{x_i}.\vec{x_j} + 1)^d$$
  - Sigmoid kernel: It is in the form
$$K(\vec{x_i}, \vec{x_j}) = \tan h(k\vec{x_i}.\vec{x_j} - \partial)$$
  - Gaussian RBF kernel: It is in the form
$$K(\vec{x_i}, \vec{x_j}) = e\frac{-\vec{x_i} - \vec{x_j}^2}{2\sigma^2}$$
- When data instances of the classes are closer to each other, this method can be used. The effectiveness of SVMdepends both on the
  - selection of the kernel function
  - adoption of values for the kernel parameters

7.5.4.4 Strengths of SVM

SVM can be used for both classification and regression.

- It is robust, i.e. not much impacted by data with noise or outliers. The prediction results using this model are very promising.

7.5.4.5 Weaknesses of SVM

- SVM is applicable only for binary classification, i.e. when there are only two classes in the problem domain.
- The SVM model is very complex – almost like a black box when it deals with a high-dimensional data set. Hence, it is very difficult and close to impossible to understand the model in such cases.
- It is slow for a large dataset, i.e. a data set with either a large number of features or a large number of instances.
- It is quite memory-intensive.

7.5.4.6 Application of SVM

- SVM is most effective when it is used for binary classification, i.e. for solving a machine

learning problemwith two classes. One common problem on which SVM can be applied is in the field of bioinformatics – more specifically, in detecting cancer and other genetic disorders. It can also be used in detecting the image of a face by binary classification of images into face and non- face components. More such applications can be described.

Summary:

1. Common classification algorithms are *k*NN, decision tree, random forest, SVM, and Naïve Bayes.
2. The *k*NN algorithm is among the best and the simplest of machine learning algorithms. A new instance is classified by a majority vote of its neighbours, with the instance being allocated the class that is predominant among its *k*NN.
3. Decision tree learning is the most broadly utilized classifier. It is characterized by fast execution time and ease in the interpretation of the rules.
4. The decision tree algorithm needs to find out the attribute splitting by which results in highest information gain. For a sample of training examples S, entropy measures the impurity of S.

$$\textbf{Entropy}(S) = \sum_{i=1}^{C} - p_i \log_2 p_i$$

where *c* is the number of different class labels and *p* refers to the proportion of values falling into the *i*-th class label.

5. Information gain is created on the basis of the decrease in entropy (*S*) after a dataset is split based on a particular attribute (*A*). Information gain for a particular feature A is calculated by the difference in entropy before a split (S_{bs}) with the entropy after the split (S_{as}).

   Information Gain (S, A) = Entropy (S_{bs}) - Entropy (S_{as})
6. Random forest is an ensemble classifier (combining classifier) which uses and combines many decision tree models. The result from an ensemble model is usually better than that from one of the individual models.
7. An SVM is a binary linear classifier. The output prediction of an SVM is one of the two conceivable classes which are already defined in the training data. SVM is also an example of a linear classifier and a maximum margin classifier.
8. SVM has a new technique called the kernel trick. These are functions, which take a lower dimensional input space and transform it to a higher dimensional space and in the process converts a non-linearly separable problem to a linearly separable problem. These functions are called kernels. When all the classes are closer to each other, this method can be used.

SHORT ANSWER-TYPE QUESTIONS (5 MARKS EACH)

1. What is supervised learning? Why it is called so?
2. Give an example of supervised learning in a hospital industry.
3. Give any three examples of supervised learning.
4. What is classification and regression in a supervised learning?
5. Give some examples of common classification algorithms.
6. Explain, in brief, the SVM model.
7. What is cost of misclassification in SVM?
8. Define Support Vectors in the SVM model.
9. Define kernel in the SVM model.
10. What are the factors determining the effectiveness of SVM?
11. What are the advantages of the SVM model?
12. What are the disadvantages of the SVM model?
13. Write notes on
    1. validation error in the *k*NN algorithm
    2. choosing *k* value in the *k*NN algorithm
    3. inductive bias in a decision tree

14. What are the advantages of the *k*NN algorithm?
15. What are the disadvantages of the *k*NN algorithm?
16. Explain, in brief, the decision tree algorithm.
17. What is node and leaf in decision tree?
18. What is entropy of a decision tree?
19. Define information gain in a decision tree.
20. Write any three strengths of the decision tree method.
21. Write any three weaknesses of the decision tree method.
22. Explain, in brief, the random forest model?

LONG ANSWER-TYPE QUESTIONS (10 MARKS EACH)

1. Distinguish between supervised learning, semi-supervised learning, and unsupervised learning.
2. Explain any five examples of classification problems in detail.
3. Explain classification steps in detail.
4. Discuss the SVM model in detail with different scenarios.
5. What are the advantages and disadvantages associated with SVM?
6. Discuss the *k*NN model in detail.
7. Discuss the error rate and validation error in the *k*NN algorithm.
8. Discuss how to calculate the distance between the test data and the training data for *k*NN.
9. Write the algorithm for *k*NN.
10. What is decision tree? What are the different types of nodes? Explain in detail
11. Explain various options of searching a decision tree.
12. Discuss the decision tree algorithm in detail.
13. What is inductive bias in a decision tree? How to avoid overfitting?
14. What are the strengths and weaknesses of the decision tree method?
15. Discuss appropriate problems for decision tree learning in detail.
16. Discuss the random forest model in detail. What are the features of random forest?
17. Discuss OOB error and variable importance in random forest.