UNIT 3:

BAYESIAN CONCEPT LEARNING & SUPERVISED LEARNING: CLASSIFICATION

UNIT – III

Bayesian Concept Learning

Introduction, Bayes Theorem: Prior, Posterior , Likelihood, Bayes Theorem Concept Learning : Concept of Consistent Learners , Bayes Optimal Classifier, Naïve Bayes Classifier , Applications of Naïve Bayes Classifier.

Supervised Learning: Classification: Introduction, Classification Model, Classification Learning Steps, Common Classification Algorithms-k-Nearest Neighbor (k-NN), Decision tree, Random forest model.

Introduction

- ✓ Bayes 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.

Hypothesis

- ✓ A supposition or proposed explanation made on the basis of limited evidence as a starting point for further investigation.
- ✓ A hypothesis is a proposed explanation for a phenomenon

Features of Bayesian learning methods

Some of the features of Bayesian learning methods that have made them popular are as follows:

1. Text-based classification such as spam or junk mail filtering, author identification, or topic categorization
2. 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.
3. Prior knowledge of the candidate hypothesis is combined with the observed data for arriving at the final probability of a hypothesis
4. 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.

5. Bayesian methods can perform better than the other methods while validating the hypotheses that make probabilistic predictions.
6. 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.
7. 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.

BAYES' THEOREM

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

Bayes' probability rule

where A and B are conditionally related events and p(A|B) denotes the probability of event A occurring when event B has already occurred.

Let us assume that we have a training data set D where we have noted some observed data. Our task is to determine the best hypothesis in space H by using the knowledge of D.

Prior

The prior knowledge or belief about the probabilities of various hypotheses in H is called Prior in context of Bayes' theorem.

P(h) is the initial probability of a hypothesis 'h' that the patient has a malignant tumor based only on the malignancy test, without considering the prior knowledge of the correctness of the test process or the so-called training data.

P(T) is the prior probability that the training data will be observed or, in this case, the probability of positive malignancy test results.

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 tumor is actually malignant

Posterior

The probability that a particular hypothesis holds for a data set based on the Prior is called the posterior probability or simply Posterior.

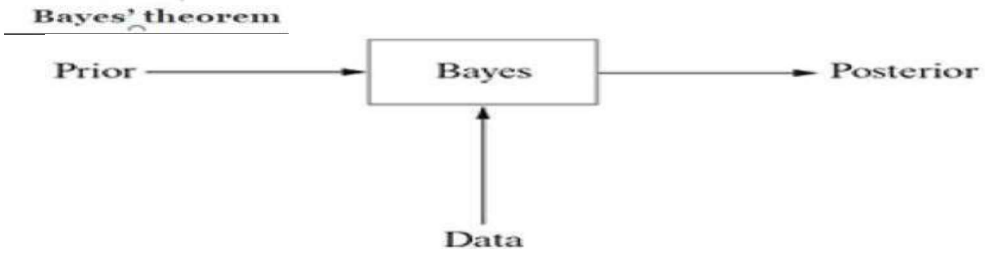
In our notation, we will say that we are interested in finding out P(H|T), which means whether the hypothesis holds true given the observed training data T.

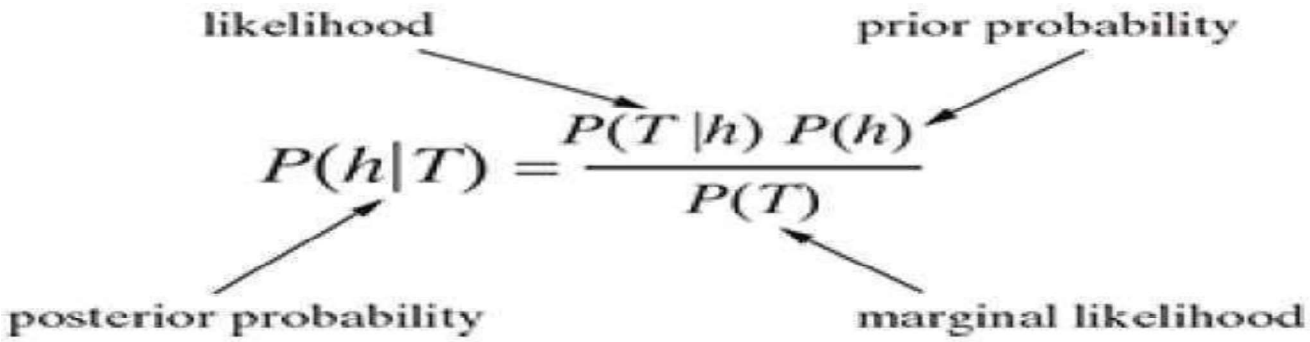
This is called the posterior probability or simply Posterior in machine learning language. According to

Bayes' theorem, combines the prior and posterior probabilities together.

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

Likelihood:-





Bayes optimal classifier

we will discuss the use of the MAP hypothesis to answer the question what is the most probable

So, extending the above example,

The set of possible outcomes for the new instance x is within the set $C = \{\text{True}, \text{False}\}$ and

$$P(h_1 \mid T) = 0.4,\ P(\text{False} \mid h_1) = 0,\ P(\text{True} \mid h_1) = 1$$
$$P(h_2 \mid T) = 0.3,\ P(\text{False} \mid h_2) = 1,\ P(\text{True} \mid h_2) = 0$$
$$P(h_3 \mid T) = 0.3,\ P(\text{False} \mid h_3) = 1,\ P(\text{True} \mid h_3) = 0$$

Then,

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

classification of the new instance given the training data. To illustrate the concept, let us assume three hypotheses h1, h2, and h3 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 h1 , but false by

h2 and h3. 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 $P(c_i|T)$ is maximum is

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

Naïve Bayes classifier

- ✓ A Naïve Bayes classifier is a primary probabilistic classifier based on a view of applying Bayes' theorem (from Bayesian inference with strong naive) independence assumptions.
- ✓ The prior probabilities in Bayes' theorem that are changed with the help of newly available information are classified as posterior probabilities.

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

- ✓ A key benefit of the naive Bayes classifier is that it requires only a little bit of training information (data) to gauge the parameters (mean and differences of the variables) essential for the classification (arrangement).
- ✓ In the Naïve Bayes classifier, independent variables are always assumed, and only the changes (variances) of the factors/variables for each class should be determined and not the whole covariance matrix.
- ✓ Because of the rather naïve assumption that all features of the dataset are equally important and

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

and

independent, this is called Naïve Bayes classifier.

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

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

Posterior Probability is of the format 'What is the probability that a particular object belongs to class i given its observed feature values?'

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

of some evidence (E) that can be observed.

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	

Naive 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, 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 problem with Naïve Bayes Classifier

Refer below table for the training dataset.

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.

Training dataset

	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

	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

Frequency Table :

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

Step 2:

To predict whether the team will win for given weather conditions (a1) = Rainy, Wins in last three matches (a2) = 2 wins, Humidity (a3) = Normal and Win toss (a4) = True, we need to choose 'Yes' from the above table for the given conditions.

From Bayes' theorem, we get

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

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

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

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

$$= 0.014109347$$

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.

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

$$= \frac{0.014109347}{0.014109347 + 0.010285714}$$

$$= 0.578368999$$

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

$$= \frac{0.010285714}{0.014109347 + 0.010285714}$$

$$= 0.421631001$$

Applications of Naïve Bayes classifier

1. **Text classification**: Naïve Bayes classifier is among the most successful known algorithms for learning to classify text documents.
2. **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'.
3. **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**.**
4. **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.

Supervised Learning: Classification

Some examples of supervised learning are as follows

a) Prediction of results of a game based on the past analysis of results
b) Predicting whether a tumor is malignant or benign on the basis of the analysis of data
c) Price prediction in domains such as real estate, stocks, etc.

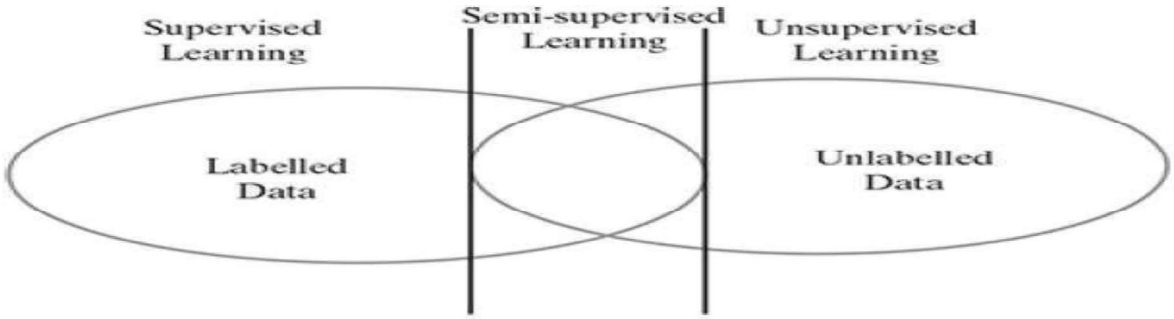
CLASSIFICATION MODEL

We can observe that in classification, the whole problem centres around assigning a label or category or class to a test data on the basis of the label or category or 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.

Classification is a type of supervised learning where a target feature, which is of categorical type, is predicted for test data on the basis of the information imparted by the training data. The target categorical feature is known as class.

Some typical classification problems include the following:

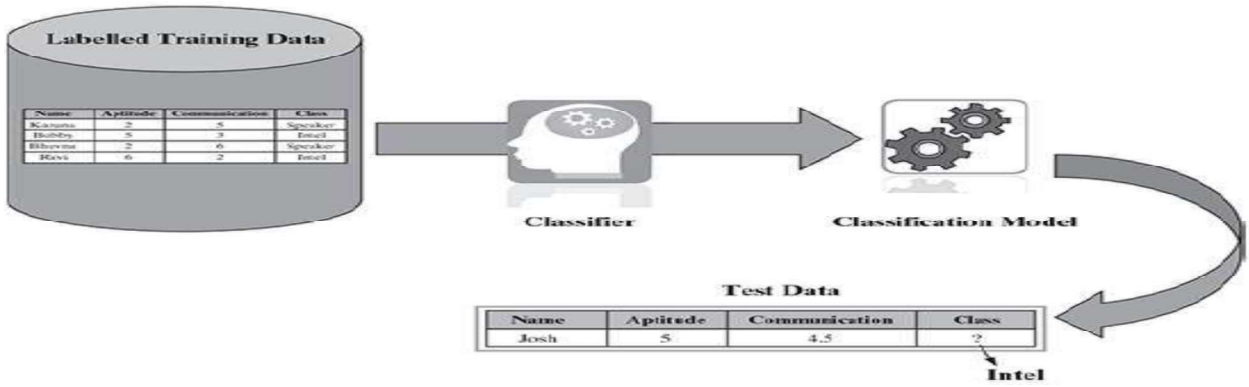
- ✓ Image classification
- ✓ Disease prediction



Supervised learning vs. unsupervised learning

- ✓ Win–loss prediction of games
- ✓ Prediction of natural calamity such as earthquake, flood, etc.
- ✓ Handwriting recognition

Classification Model



CLASSIFICATION LEARNING STEPS

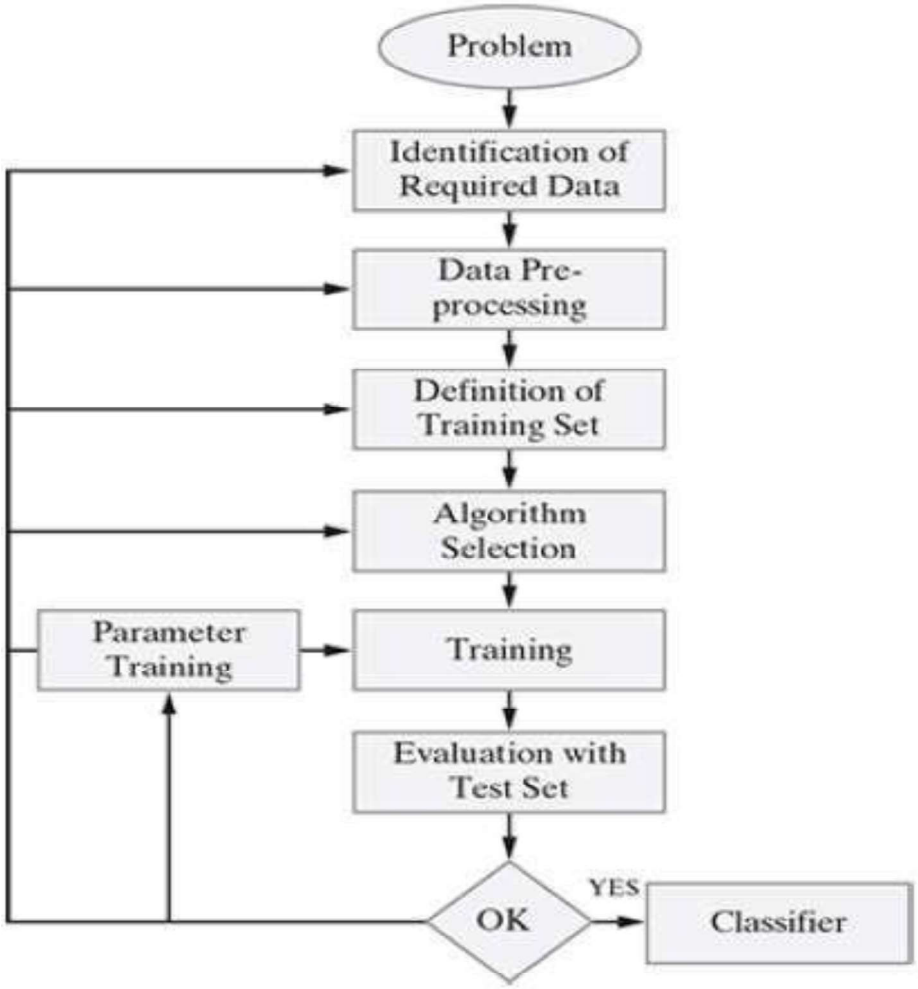
Following steps are followed in classification learning model.

1) **Problem Identification:** Identifying the problem is the first step in the supervised learning model. The problem needs to be a well-formed problem.
2) **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.
3) **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.

4) **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.

5) Algorithm Selection: This involves determining the structure of the learning function and the corresponding learning 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.

6) Training: The learning algorithm identified in the previous step is run on the gathered training set for further fine tuning. Some supervised learning algorithms require 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.



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

COMMON CLASSIFICATION ALGORITHMS

Following are the common classification algorithms –

1. k-Nearest Neighbor (k-NN)
2. Decision tree
3. Random forest
4. Support Vector Machine (SVM)
5. Naïve Bayes classifier

a) k-Nearest Neighbor (k-NN)

- The k-NN algorithm is a simple but extremely powerful classification algorithm.
- The name of the algorithm originates from the underlying philosophy of k-NN — i.e. people having similar background or mindset tend to stay close to each other. In other words, neighbors in a locality have a similar background.
- In the same way, as a part of the k-NN algorithm, the unknown and unlabeled data which comes for a prediction problem is judged on the basis of the training data set elements which are similar to the unknown element.

How KNN Works?

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 predominantly present in the training data points, assign that class label to the test data point

End do

Working factor of choose the Factor K

- It is often a tricky decision to decide the value of k. The reasons are as follows:
    - If the value of k is very large (in the extreme case equal to the total number of records in the training data), the class label of the majority class of the training data set will be assigned to the test data regardless of the class labels of the neighbors nearest to the test data.
    - If the value of k is very small (in the extreme case equal to 1), the class value of a noisy data or outlier in the training data set which is the nearest neighbor to the test data will be assigned to the test data.
- The best k value is somewhere between these two extremes.

Few strategies, highlighted below, are adopted by machine learning practitioners to arrive at a value for k.

1) One common practice is to set k equal to the square root of the number of training records.
2) An alternative approach is to test several k values on a variety of test data sets and choose the one that delivers the best performance.
3) Another interesting approach is to choose a larger value of k, but apply a weighted voting process in which the vote of close neighbors is considered more influential than the vote of distant neighbors.

Example

Student dataset

Name	Aptitude	Communication	Class
Karuna	2	5	Speaker
Bhuvna	2	6	Speaker
Gaurav	7	6	Leader
Parul	7	2.5	Intel
Dinesh	8	6	Leader
Jani	4	7	Speaker
Bobby	5	3	Intel
Parimal	3	5.5	Speaker
Govind	8	3	Intel
Susant	6	5.5	Leader
Gouri	6	4	Intel
Bharat	6	7	Leader
Ravi	6	2	Intel
Pradeep	9	7	Leader
Josh	5	4.5	Intel

Segregated student data set

	Name	Aptitude	Communication	Class
Training Data	Karuna	2	5	Speaker
	Bhuvna	2	6	Speaker
	Gaurav	7	6	Leader
	Parul	7	2.5	Intel
	Dinesh	8	6	Leader
	Jani	4	7	Speaker
	Bobby	5	3	Intel
	Parimal	3	5.5	Speaker
	Govind	8	3	Intel
	Susant	6	5.5	Leader
	Gouri	6	4	Intel
	Bharat	6	7	Leader
	Ravi	6	2	Intel
	Pradeep	9	7	Leader
Test Data →	Josh	5	4.5	Intel

Modeling

- In the *k*-NN algorithm, the class label of the test data elements is decided by the class label of the training data elements which are neighboring, i.e. similar in nature.

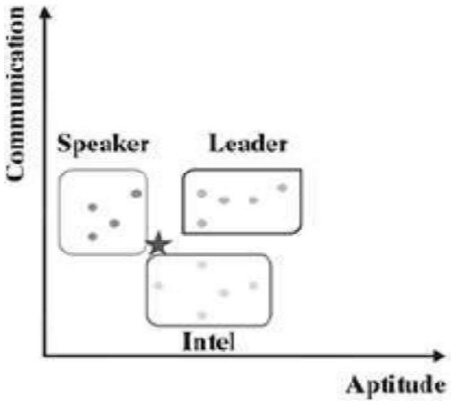
But there are two challenges:

1. What is the basis of this similarity or when can we say that two data elements are similar?
2. How many similar elements should be considered for deciding the class label of each test data element?

- The most common approach adopted by *k*-NN to measure similarity between two data elements is **Euclidean distance**.

Euclidean Distance assumes that an object can travel freely in space and reach point $B(x_1, y_1)$ from point $A(x_2, y_2)$ in a straight line.

The formula for the euclidian distance is : $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$



Name	Aptitude	Communication	Class
Karuna	2	5	Speaker
Bhuvna	2	6	Speaker
Gaurav	7	6	Leader
Parul	7	2.5	Intel
Dinesh	8	6	Leader
Jani	4	7	Speaker
Bobby	5	3	Intel
Parimal	3	5.5	Speaker
Govind	8	3	Intel
Susant	6	5.5	Leader
Gouri	6	4	Intel
Bharat	6	7	Leader
Ravi	6	2	Intel
Pradeep	9	7	Leader
Josh	5	4.5	???

- ✓ In the *k*NN algorithm, the value of '*k*' indicates the number of neighbors that need to be considered.

Name	Aptitude	Communication	Class	Distance	*k* = 1	*k* = 2	*k* = 3
Karuna	2	5	Speaker	3.041			
Bhuvna	2	6	Speaker	3.354			
Parimal	3	5.5	Speaker	2.236			
Jani	4	7	Speaker	2.693			
Bobby	5	3	Intel	1.500			1.500
Ravi	6	2	Intel	2.693			
Gouri	6	4	Intel	1.118	1.118	1.118	1.118
Parul	7	2.5	Intel	2.828			
Govind	8	3	Intel	3.354			
Susant	6	5.5	Leader	1.414			
Bharat	6	7	Leader	2.693			
Gaurav	7	6	Leader	2.500			
Dinesh	8	6	Leader	3.354			
Pradeep	9	7	Leader	4.717			
Josh	5	4.5	???				

- ✓ For example, if the value of *k* is 3, only three nearest neighbors or three training data elements closest to the test data element are considered. Out of the three data elements, the class which is predominant is considered as the class label to be assigned to the test data.
- ✓ In case the value of k is 1, only the closest training data element is considered. The class label of that data element is directly assigned to the test data element.

Why KNN algorithm is called as Lazy Learner?

- ✓ It only stores a training dataset versus undergoing a training stage.
- ✓ This also means that all the computation occurs when a classification or prediction is being made.
- ✓ Since it heavily relies on memory to store all its training data, it is also referred to as an instance-based or memory-based learning method.

Strength and Weaknesses of KNN algorithm

Strengths

- ✓ 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

Weakness

- ✓ Does not learn anything in the real sense. Classification is done completely on the basis of the training data.
- ✓ Does not scale well: Since KNN is a lazy algorithm, it takes up more memory and data storage compared to other classifiers. This can be costly from both a time and money perspective.
- ✓ Curse of dimensionality: The KNN algorithm tends to fall victim to the curse of dimensionality, which means that it doesn't perform well with high-dimensional data inputs.

Applications

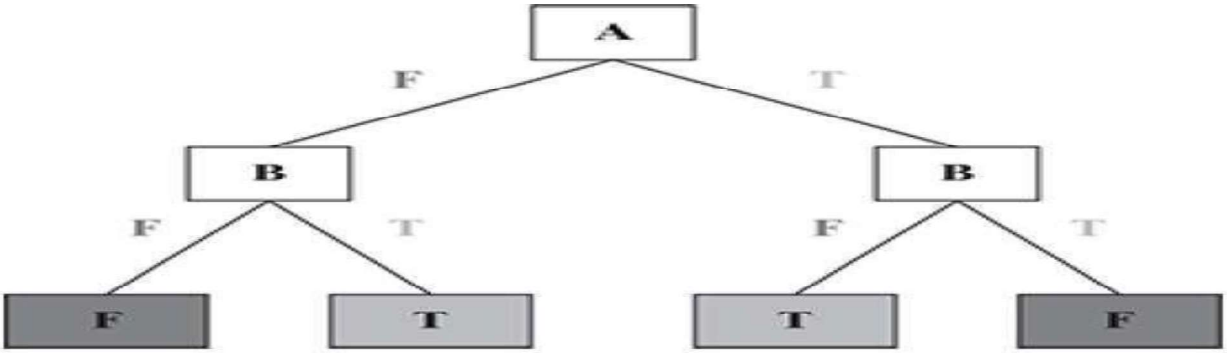
1) Recommendation Engines
2) Healthcare
3) Searching documents/ contents similar to a given document/content.

b) Decision tree algorithm

- ✓ 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.
- ✓ 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.
- ✓ Each node (or decision 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.



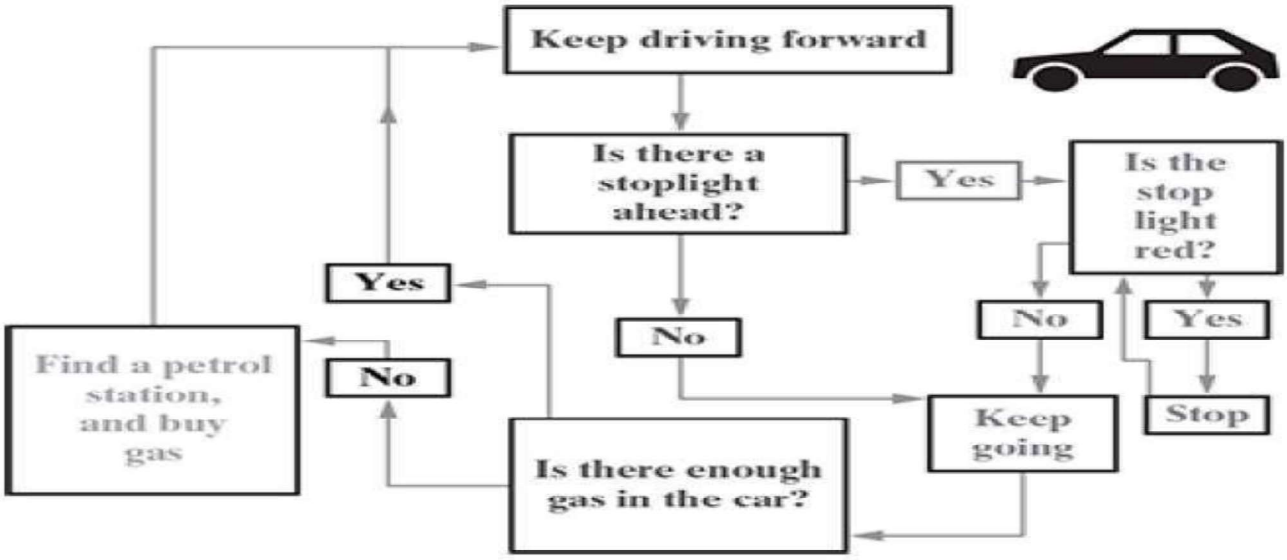
Decision tree structure

A decision tree consists of three types of nodes:

- Root Node
- Branch Node
- Leaf Node

Example

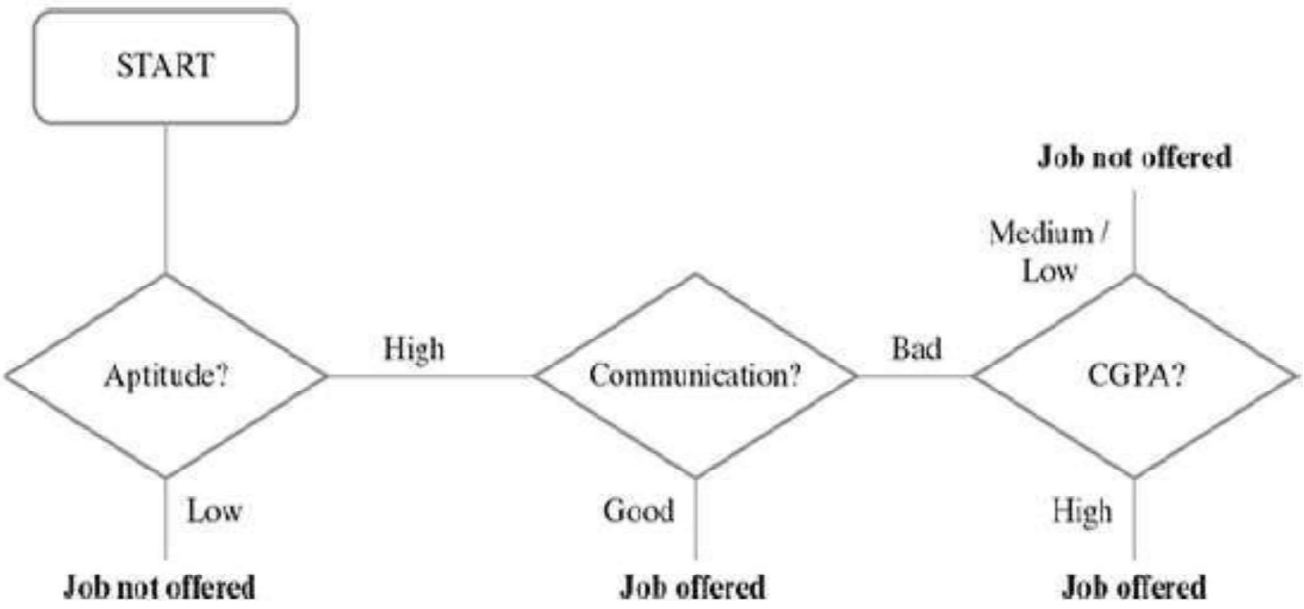
An example decision tree for a car driving – the decision to be taken is whether to 'Keep Going' or to 'Stop', which depends on various situations as depicted in the figure. If the signal is RED in colour, then the car should be stopped. If there is not enough gas (petrol) in the car, the car should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



Building a Decision Tree

Training data set

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



Decision tree based on the training data

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.

Implementations of Decision tree

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***.

A Decision tree is built based on two properties

1. Entropy
2. Information gain

Entropy

Entropy is an information theory metric that measures the impurity or uncertainty in a group of observations. It determines how a decision tree chooses to split data.

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 p refers to the proportion of values falling into the i-th class label.

Example:

Let's have a dataset made up of three colors; red, purple, and yellow. If we have one red, three purple, and four yellow observations in our set, our equation becomes:

$$E = -(p_r log_2 p_r + p_p log_2 p_p + p_y log_2 p_y)$$

Where p_r, p_p and p_y are the probabilities of choosing a red, purple and yellow example respectively. We have $p_r = \frac{1}{8}$ because only $\frac{1}{8}$ of the dataset represents red. $\frac{3}{8}$ of the dataset is purple hence $p_p = \frac{3}{8}$. Finally, $p_y = \frac{4}{8}$ since half the dataset is yellow. As such, we can represent p_y as $p_y = \frac{1}{2}$. Our equation now becomes:

$$E = -(\frac{1}{8} log_2(\frac{1}{8}) + \frac{3}{8} log_2(\frac{3}{8}) + \frac{4}{8} log_2(\frac{4}{8}))$$

Our entropy would be: 1.41

- ✓ what happens when all observations belong to the same class? In such a case, the entropy will always be zero.

$$E = -(1 log_2 1)$$

$$= 0$$

- ✓ Such a dataset has no impurity. This implies that such a dataset would not be useful for learning.
- ✓ However, if we have a dataset with say, two classes, half made up of yellow and the other half being purple, the entropy will be one.

$$E = -((0.5 log_2 0.5) + (0.5 log_2 0.5))$$

$$= 1$$

- ✓ This kind of dataset is good for learning.

Information gain

- ✓ 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 (i.e. the most homogeneous branches).

Information gain for a particular feature A is calculated by the difference in entropy before a split (or S_{bs}) with the entropy after the split (S_{as}).

$$\textbf{Information Gain (S, A) = Entropy } (S_{bs}) - \textbf{Entropy } (S_{as})$$

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.

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

Example

Suppose we have a dataset with two classes. This dataset has 5 purple and 5 yellow examples. The initial value of entropy will be given by the equation below. Since the dataset is balanced, we expect the answer to be 1.

$$E_{initial} = -((0.5log_20.5) + (0.5log_20.5))$$

$$= 1$$

Say we split the dataset into two branches. One branch ends up having four values while the other has six. The left branch has four purples while the right one has five yellows and one purple.

We mentioned that when all the observations belong to the same class, the entropy is zero since the dataset is pure. As such, the entropy of the left branch $E_{left} = 0$. On the other hand, the right branch has five yellows and one purple. Thus:

$$E_{right} = -(\frac{5}{6}log_2(\frac{5}{6}) + \frac{1}{6}log_2(\frac{1}{6}))$$

A perfect split would have five examples on each branch. This is clearly not a perfect split, but we can determine how good the split is. We know the entropy of each of the two branches. We weight the entropy of each branch by the number of elements each contains.

The entropy before the split, which we referred to as initial entropy $E_{initial} = 1$. After splitting, the current value is 0.39. We can now get our information gain, which is the entropy we "lost" after splitting.

$$Gain = 1 - 0.39$$

$$= 0.61$$

The more the entropy removed, the greater the information gain. The higher the information gain, the better the split.

This helps us calculate the quality of the split. The one on the left has 4, while the other has 6 out of a total of 10. Therefore, the weighting goes as shown below:

$$E_{split} = 0.6 * 0.65 + 0.4 * 0$$

$$= 0.39$$

Avoiding overfitting in decision tree – pruning

There are two approaches of pruning:

a) Pre-pruning: Stop growing the tree before it reaches perfection.

b) 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 over fitting as well as optimizes computational cost. However, it also stands a chance to ignore important information contributed by a feature which was skipped, thereby resulting in miss out of certain patterns in the data.
- ✓ 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.

Strengths and weaknesses of Decision trees

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 more useful for classification.

Weaknesses of decision tree

- ✓ Decision tree models are often biased towards features having more number of possible values, i.e. levels.
- ✓ This model gets over fitted or under fitted 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.

Application of decision tree

Here are some applications of decision trees:

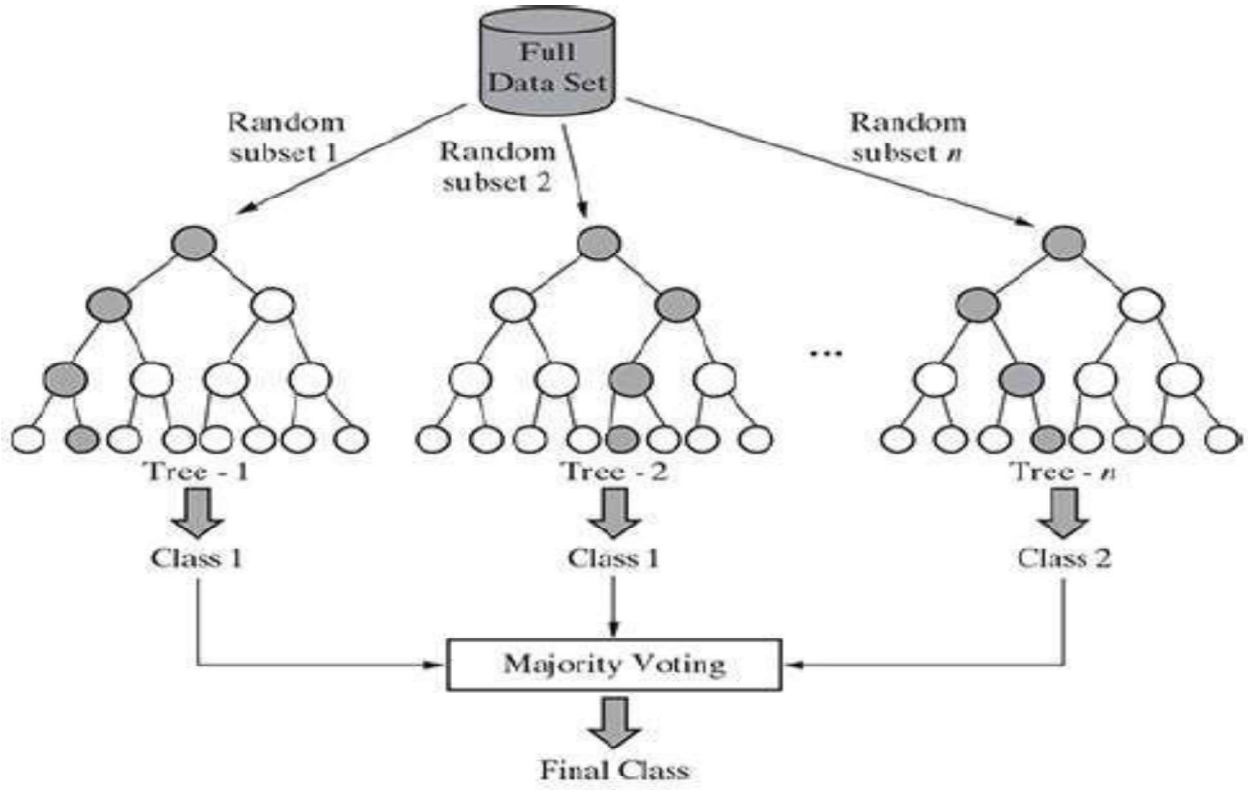
1. Marketing
2. Retention of Customers
3. Diagnosis of Diseases and Ailments
4. Detection of Frauds

1) 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.



Random forest working Process

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 rate displayed is often surprisingly high.

Strengths and Weaknesses of Random forest

Strengths of random forest

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

Weaknesses of random forest

1) This model, because it combines a number of decision tree models, is not as easy to understand as a decision tree model.
2) It is computationally much more expensive than a simple model like decision 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.

TEXT BOOKS:

1. Saikat Dutt, Subramanian Chandramouli, Amit Kumar Das, Machine Learning, Pearson, 2019.

REFERENCE BOOKS:

1. Ethern Alpaydin, ― Introduction to Machine Learning, MIT Press, 2004.
2. Stephen Marsland, ― Machine Learning - An Algorithmic Perspective, Second Edition, Chapman and Hall / CRC Machine Learning and Pattern Recognition Series, 2014