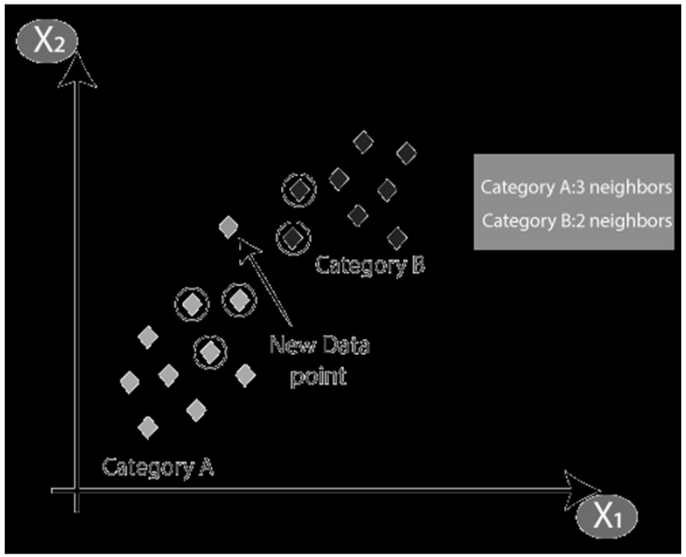
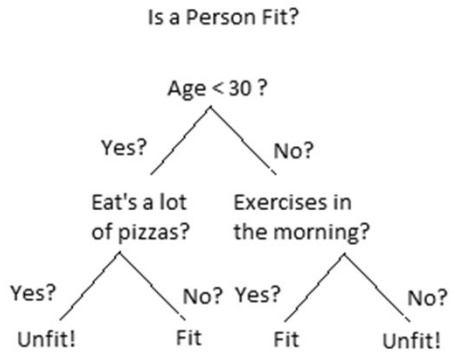
UNIT – 4 Supervised Learning – Classification, Regression

By : Prof. Nootan Padia, Marwadi University, Rajkot

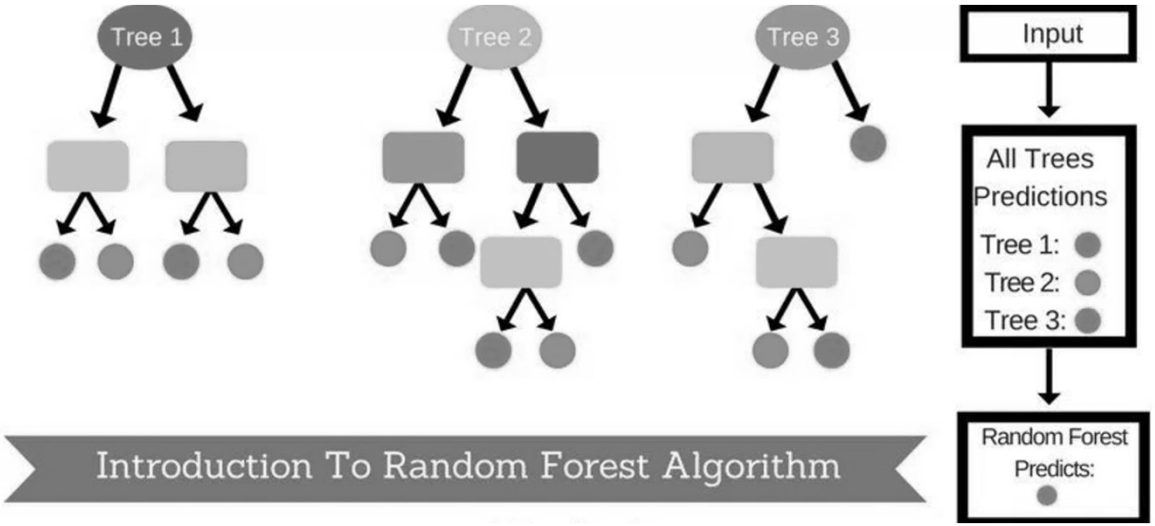
Introduction

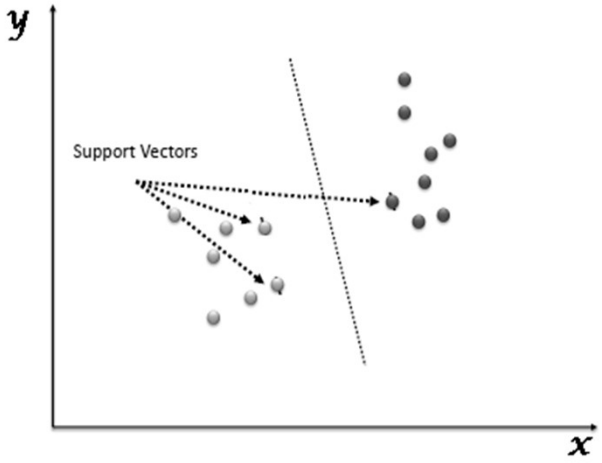
- KNN (K – Nearest Neighbour) – which tries to classify unlabelled data instances based on the similarity with the labelled instances in the training data.
- Decision Tree – , as the name suggests, is a classifier based on a series of logical decisions, which resembles a tree with branches.





- Random Forest Classifier – can be thought as a collection of many decision trees.
- SVM (Support Vector Machine) - In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well





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.

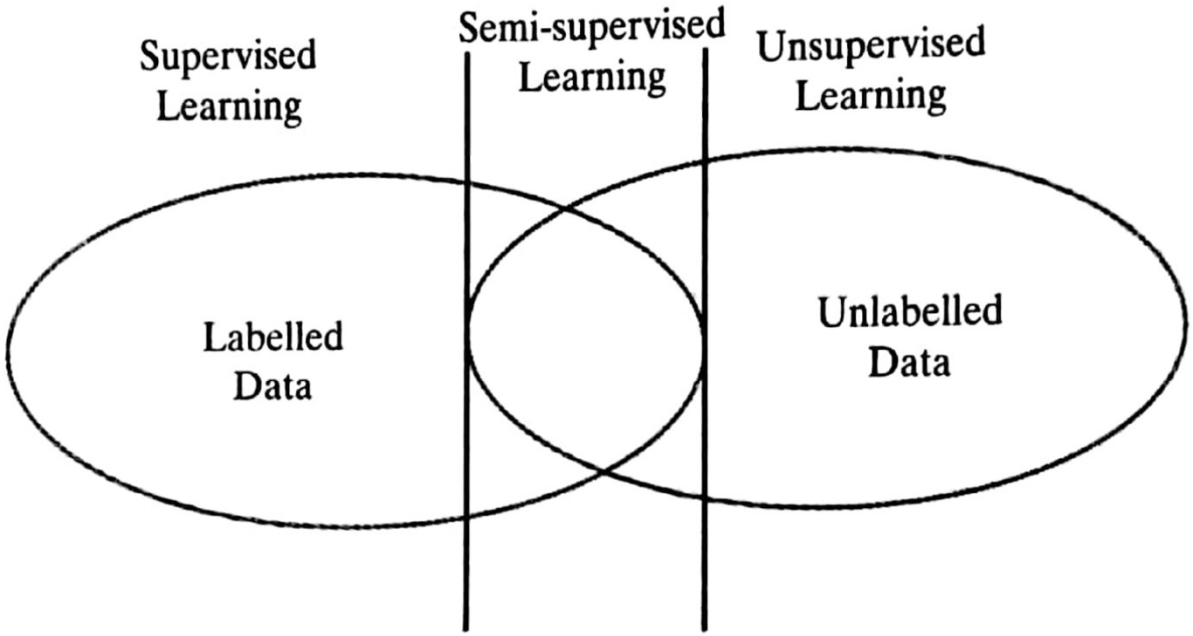


FIG. 7.1
Supervised learning vs. unsupervised learning

- 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 worsen 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 worsening and thus need to be moved to ICU.
- This kind of prediction problem comes under the pureview of supervised learning or, more specifically, under classification.
- The hospital already has all past patient records.

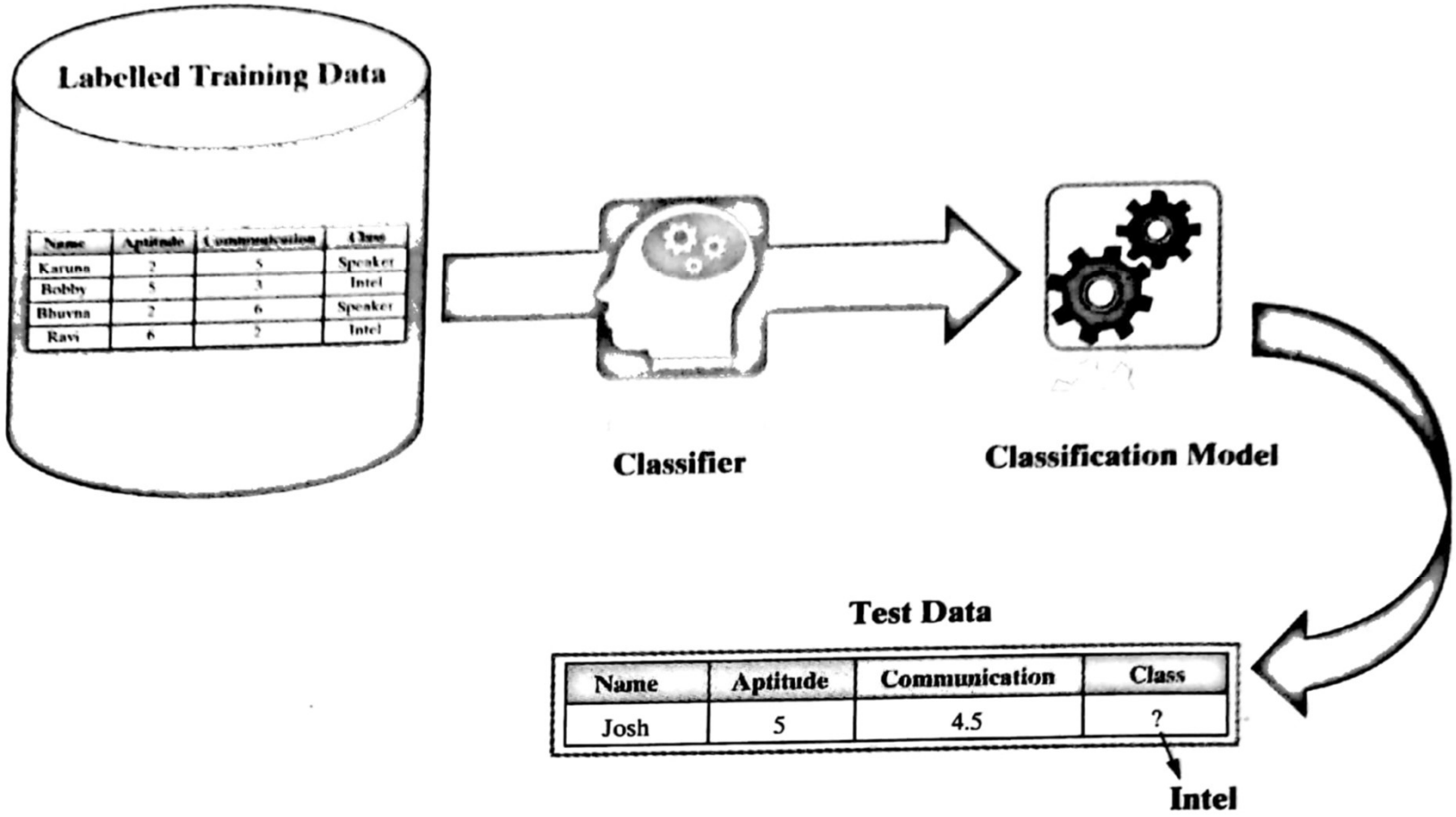
- The records of the patients whose health condition aggravated in the past and had to be moved to ICU can form the training data for this prediction problem.
- Test results of newly admitted patients are used to classify them as high risk or low risk patients.
- Some more examples of supervised learning 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.
- *'IBM Watson Health' developed by IBM software provides evidence-hacked cancer care to each patient by understanding millions of data points.*
- *'Watson for Oncology' helps physicians quickly identify vital information in a patient's medical record, surface relevant evidence, and explore various treatment options for patients*

Classification

CLASSIFICATION MODEL

- Let us consider two examples, say 'predicting whether a tumour is malignant or benign' and 'price prediction in the domain of real estate' .
- Are these two problems same in nature?
- The answer is 'no'.
- It is true that both of them are problems related to prediction.
- However, for tumour prediction, we are trying to predict which category or class. i.e. 'malignant' or 'benign', an unknown input data related to tumour belongs to.
- In the other case, that is, for price prediction, we are trying to predict an absolute value and not a class.
- 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 centers 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.
- Figure 7.2 depicts the typical process of classification where a classification model is obtained from the labelled training data by a classifier algorithm.
- On the basis of the model, a class label (e.g.'lntel' as in the case of the test data referred in Fig. 7.2) is assigned to the test data.


Labelled Training Data
Name
Aptitude
Communication
Class
Karuna
2
5
Speaker
Bobby
5
3
Intel
Bhuvna
2
6
Speaker
Ravi
6
2
Intel
Classifier
Classification Model
Test Data
Name
Aptitude
Communication
Class
Josh
5
4.5
?
Intel

- A critical classification problem in the context of the banking domain is identifying potentially fraudulent transactions.
- Because there are millions of transactions which 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 or labelled as usual or suspicious.
- The suspicious transactions are subsequently segregated for a closer review.
- In summary, 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
    - Win-loss prediction of games
    - Prediction of natural calamity such as earthquake, flood, etc.
    - Handwriting recognition

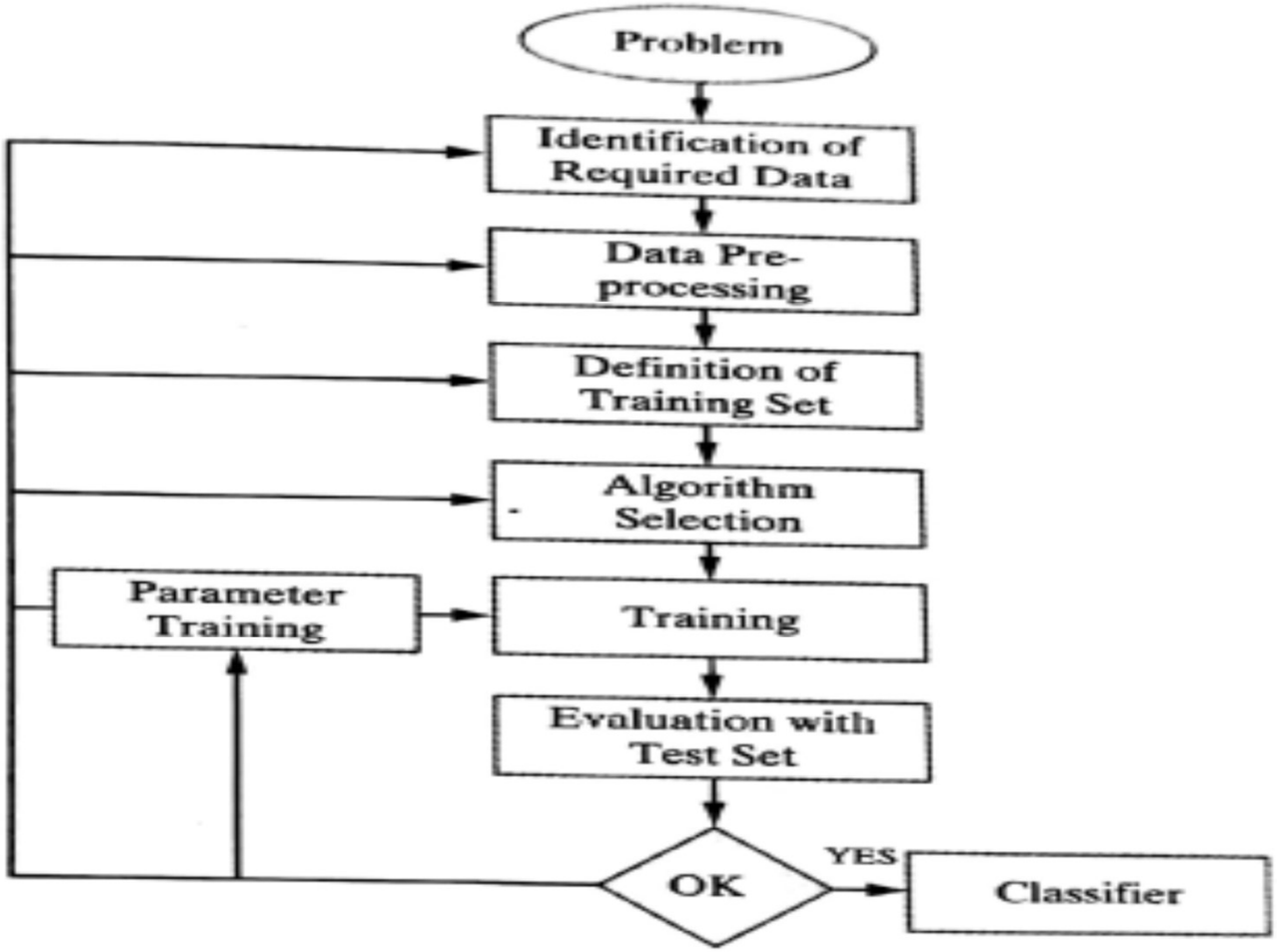
CLASSIFICATION LEARNING STEPS

1. 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 a long-term impact.

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.
- 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 tumors are to be identified.


Problem
Identification of Required Data
Data Pre-processing
Definition of Training Set
Algorithm Selection
Parameter Training
Training
Evaluation with Test Set
OK
YES
Classifier

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.
- Because the data is gathered from different sources, it is usually collected in a raw format and is not ready for immediate analysis.
- This step ensures that the data is ready to be fed into the machine learning 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.
- 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.

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

KNN (K – Nearest Neighbours)

k-Nearest Neighbour (kNN)

- The kNN algorithm is a simple but extremely powerful classification algorithm.
- The name of the algorithm originates from the underlying philosophy of kNN - i.e. people having similar background or mindset tend to stay close to each other.
- In other words, neighbours in a locality have a similar background.
- In the same way, as a part of the kNN algorithm, the unknown and unlabelled 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.
- So, the class label of the unknown element is assigned on the basis of the class labels of the similar training data set elements.

How kNN works

- Let us consider a very simple Student data set as depicted in Figure 7.4.
- It consists of 15 students studying in a class.
- Each of the students has been assigned a score on a scale of 10 on two performance parameters - 'Aptitude' and 'Communication'.
- Also, a class value is assigned to each student based on the following criteria:
  - 1. Students having good communication skills as well as a good level of aptitude have been classified as 'Leader'
  - 2. Students having good communication skills but not so good level of aptitude have been classified as 'Speaker'
  - 3. Students having not so good communication skill but a good level of aptitude have been classified as 'Intel'

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

FIG. 7.4
Student data set

- While building a classification model, a part of the labelled input data is retained as test data.
- The remaining portion of the input data is used to train the model - hence known as training data.
- The motivation to retain a part of the data as test data is to evaluate the performance of the model.
- As we have seen, the performance of the classification model is measured by the number of correct classifications made by the model when applied to an unknown data set.
- However, it is not possible during model testing to know the actual label value of an unknown data.
- Therefore, the test data, which is a part of the labelled input data, is used for this purpose.

- If the class value predicted for most of the test data elements matches with the actual class value that they have, then we say that the classification model possesses a good accuracy.
- In context of the Student data set, to keep the things simple, we assume one data element of the input data set as the test data.
- As depicted in Figure 7.5, the record of the student named Josh is assumed to be the test data.
- Now that we have the training data and test data identified, we can start with the modeling.
- As we have already discussed, in the kNN 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?

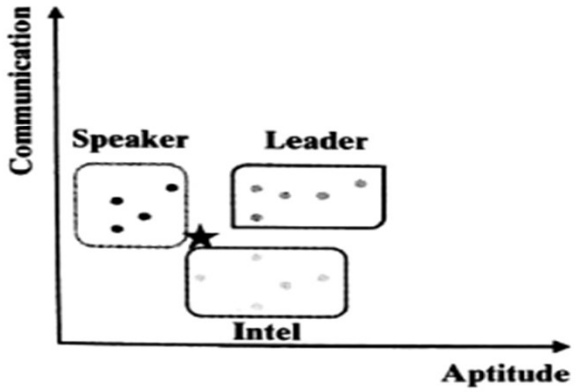
- 2. How many similar elements should be considered for deciding the class label of each test data element?

- To answer the first question, though there are many measures of similarity, the most common approach adopted by kNN to measure similarity between two data elements is Euclidean distance.
- Considering a very simple data set having two features (say f_1 and f_2), Euclidean distance between two data elements d_1 and d_2 can be measured by

$$\text{Euclidean distance} = \sqrt{(f_{11} - f_{12})^2 + (f_{21} - f_{22})^2}$$

- where
- f_{11} = value of feature f_1 for data element d_1,
- f_{12} = value of feature f_1 for data element d_2
- f_{21}= value of feature f_2 for data element d_1,
- f_{22} = value of feature f_2 for data element d_2

- So, as depicted in Figure 7.6, the training data points of the Student data set considering only the features 'Aptitude' and 'Communication' can be represented as dots in a two-dimensional feature space.



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	?

FIG. 7.6
2-D representation of the student data set

- As shown in the figure, the training data points having the same class value are coming close to each other.
- The reason for considering two-dimensional data space is that we are considering just the two features of the Student data set, i.e. 'Aptitude' and 'Communication', for doing the classification.
- The feature 'Name' is ignored because, as we can understand, it has no role to play in deciding the class value.
- The test data point for student Josh is represented as an asterisk in the same space.
- To find out the closest or nearest neighbours of the test data point, Euclidean distance of the different dots need to be calculated from the asterisk.
- Then, the class value of the closest neighbours helps in assigning the class value of the test data element.

- Now, let us try to find the answer to the second question, i.e. how many similar elements should be considered.
- The answer lies in the value of which is a user-defined parameter given as an input to the algorithm.
- In the kNN algorithm, the value of ‘k’ indicates the number of neighbours that need to be considered.
- For example, if the value of k is 3, only three nearest neighbours 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.

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	???				

FIG. 7.7
Distance calculation between test and training points

- Let us now try to find out the outcome of the algorithm for the Student data set we have.
- In other words, we want to see what class value kNN will assign for the test data for student Josh.
- Again, let us refer back to Figure 7.7.
- As is evident, when the value of k is taken as 1, only one training data point needs to be considered.
- The training record for student Gouri comes as the closest one to test record of Josh. with a distance value of 1.118.
- Gouri has class value 'Intel'.
- So, the test data point is also assigned a class label value 'Intel'.
- When the value of k is assumed as 3, the closest neighbours of Josh in the training data set are Gouri, Susant, and Bobby with distances being 1.118, 1.414, and 1.5, respectively.
- Gouri and Bobby have class value 'Intel', while Susant has class value 'Leader'.
- In this case, the class value of Josh is decided by majority voting.
- Because the class value of 'Intel' is formed by the majority of the neighbours, the class value of Josh is assigned as 'Intel'.

- This same process can be extended for any value of k.
- But 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 neighbours 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 neighbour 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.
  - One common practice is to set k equal to the square root of the number of training records.
  - 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.
  - Another interesting approach is to choose a larger value of k. but apply a weighted voting process in which the vote of close neighbours is considered more influential than the vote of distant neighbours.

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

End do

Why the kNN algorithm is called a lazy learner?

- We have already discussed in Chapter 3 that 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, as we have seen 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 and weaknesses of the 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
- **Weaknesses**
  - 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.
  - 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.

Decision Tree

Decision tree

- 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 (based on the past data called past vector) that predicts the value of the output variable based on the input variables in the feature vector.
- 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.
- A decision tree is usually represented in the format depicted here:

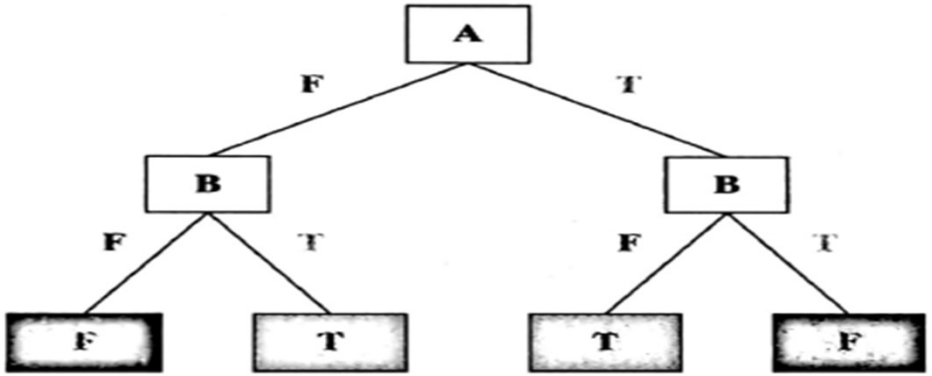


FIG. 7.8
Decision tree structure

- Each internal node (represented by boxes) tests an attribute (represented as 'A'/'B' within the boxes).
- Each branch corresponds to an attribute value (T/F) in the above case.
- Each leaf node assigns a classification.
- 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 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 color, 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.

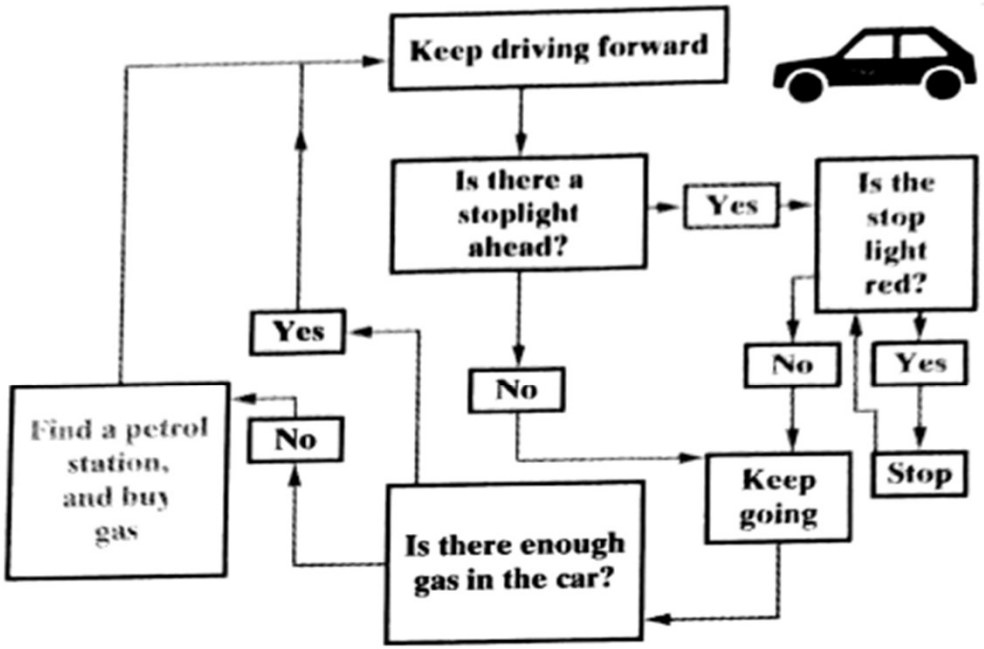


FIG. 7.9
Decision tree example

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 —
    1. All or most of the examples at a particular node have the same class
    2. All features have been used up in the partitioning
    3. The tree has grown to a pre-defined threshold limit
- Let us 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.

- 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
- 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 in next slide.
- 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.

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

- 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 in next slide depicts the complete decision tree diagram for the table given in following Figure.

FIG. 7.11
Decision tree based on the training data

Searching a decision tree

- By using the decision tree depicted in Figure in previous slide, 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**

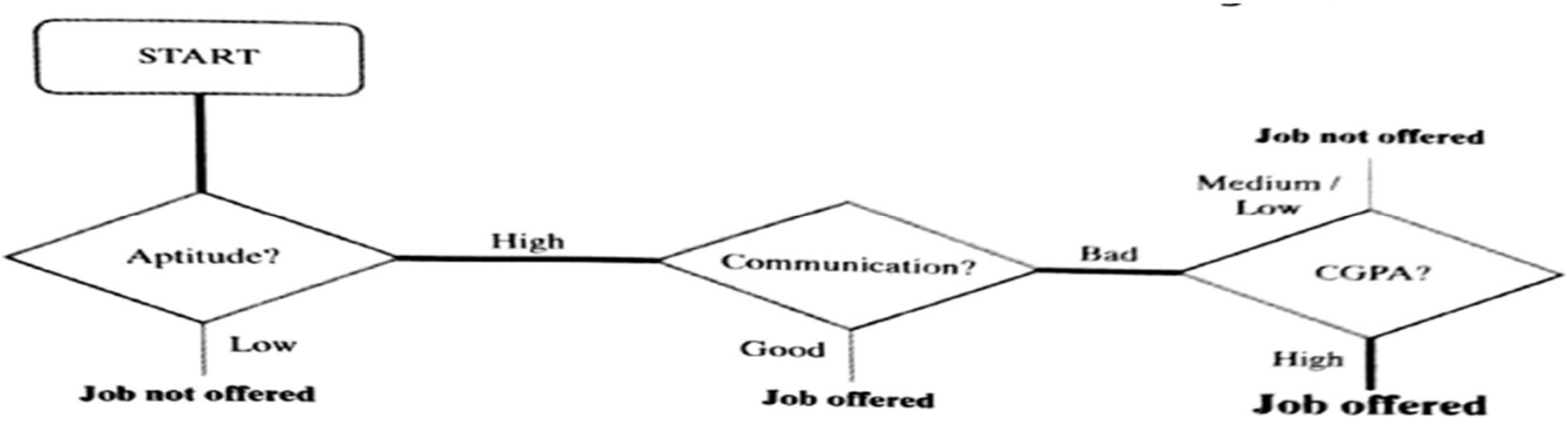
    1. Place the item in the first group (class). Recursively examine solutions with the item in the first group (class).

    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.



- Figure 7.12 depicts a sample path (thick line) for the conditions CGPA = High, Communication = Bad, Aptitude = High and Programming skills = Bad.
- According to the above decision tree, the prediction can be 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).

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 $E_i < E_{\text{min}}$

then $E_{\text{min}} = E_{\text{i}}$ and $F_{\text{min}} = F_i$

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.

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 quite evident, 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 and unlabelled 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 to ignore important information contributed by a feature which was skipped, thereby resulting in miss out of certain 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.

Strengths and weaknesses of Decision Tree

- **Strengths**
  - 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**
  - Decision tree models are often biased towards features having more number of possible values, i.e. levels.
  - This model gets overfitted or underlined 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

- 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.
- The most straightforward case exists when there are only two possible values for an attribute (Boolean classification). E.g. Communication is having two values – Good or Bad.
- It is also easy to extend the decision tree to create a target function with more than two possible output values.
- 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 for a 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.

Entropy

- 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 p refers to the proportion of values falling into the i-th class label.
- For example, with respect to the training data in Figure in next slide, we have two values for the target class 'Job Offered?' - Yes and No.
- The value of p, 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) = -0.44 log x(0.44) - 0.56 log 2(0.56) = 0.99.

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

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 (i.e. the most homogeneous branches).
- If the information gain is 0, it means that there is no reduction in entropy due to split of the data set according to 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 (or 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.

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

- Let us examine the value of information gain for the training data set shown in Figure in previous slide.
- We will find the value of entropy at the beginning before any split happens and then again after the split happens.

- We will compare the values for all the cases –
    1. when the feature 'COPA' 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 level split 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

- Figure 7.13(A) Entropy and information gain calculation (Level 1)

(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

- 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 not continue 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

FIG. 7.13B (*Continued*)

- Fig. 7.13(B) Entropy and Information gain calculation (Level 2)

(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

- 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.

Random Forest

Random forest

- Random forest is an ensemble(small group) 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.

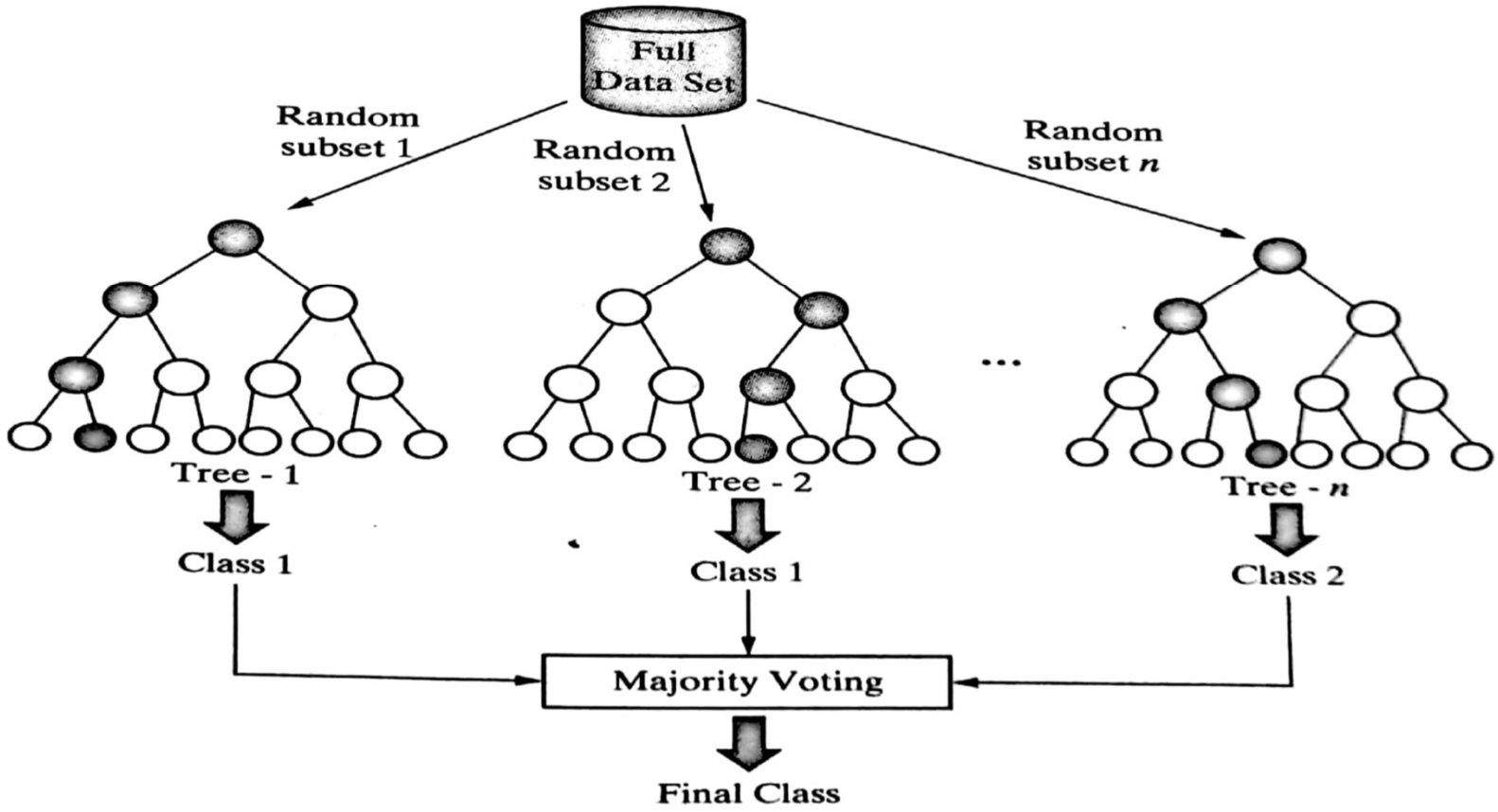


FIG. 7.14
Random forest model

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 of data instances should be picked randomly.
  - **2.** Use the best split principle on these 'm' features to calculate the numb, at nodes
  - **3.** Keep splitting the nodes to child nodes till the tree is grown to the maxima, 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 boot-strap 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 (matched), 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.
- Became of this reason, the error rate displayed is often surprisingly high.

Strength and Weakness

- **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 population of 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 using on other data.
  - Lastly, the random forest algorithm can he used to solve both classification and regression problems

- **Weaknesses of random forest**
  - This model, because it combines a number of decision tree models, is not as easy to understand as a decision tree model.
  - 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.

SVM (Support Vector Machines)

Support vector machines (SVM)

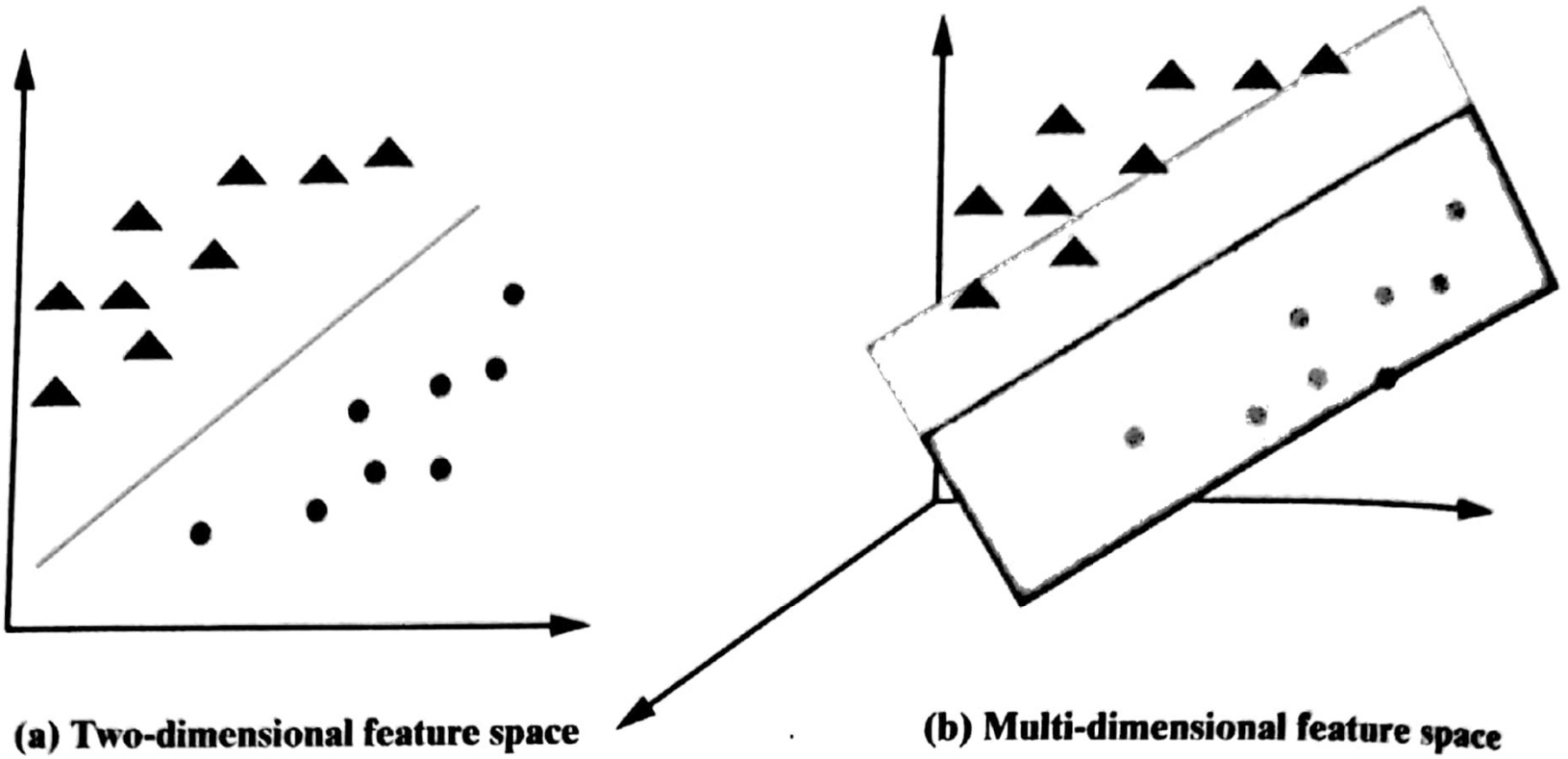
- SVM is a model, which can do linear classification as well as regression.
- SVM is based on the concept of a surface, called a hyperplane, which draws a boundary between data instances plotted in the multi-dimensional feature space.
- The output prediction of an SVM is one of two possible classes which 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.

- **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.
- **Weakness 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.

- **Application of SVM**
  - SVM is most effective when it is used for binary classification, i.e. for solving a machine learning problem with 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.

Classification using hyperplanes

- In SVM, a model is built to discriminate the data instances belonging to different classes.
- Let as 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 data instances belonging to different classes transform to a hyperplane as depicted in Figure 7.15b.
- 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.



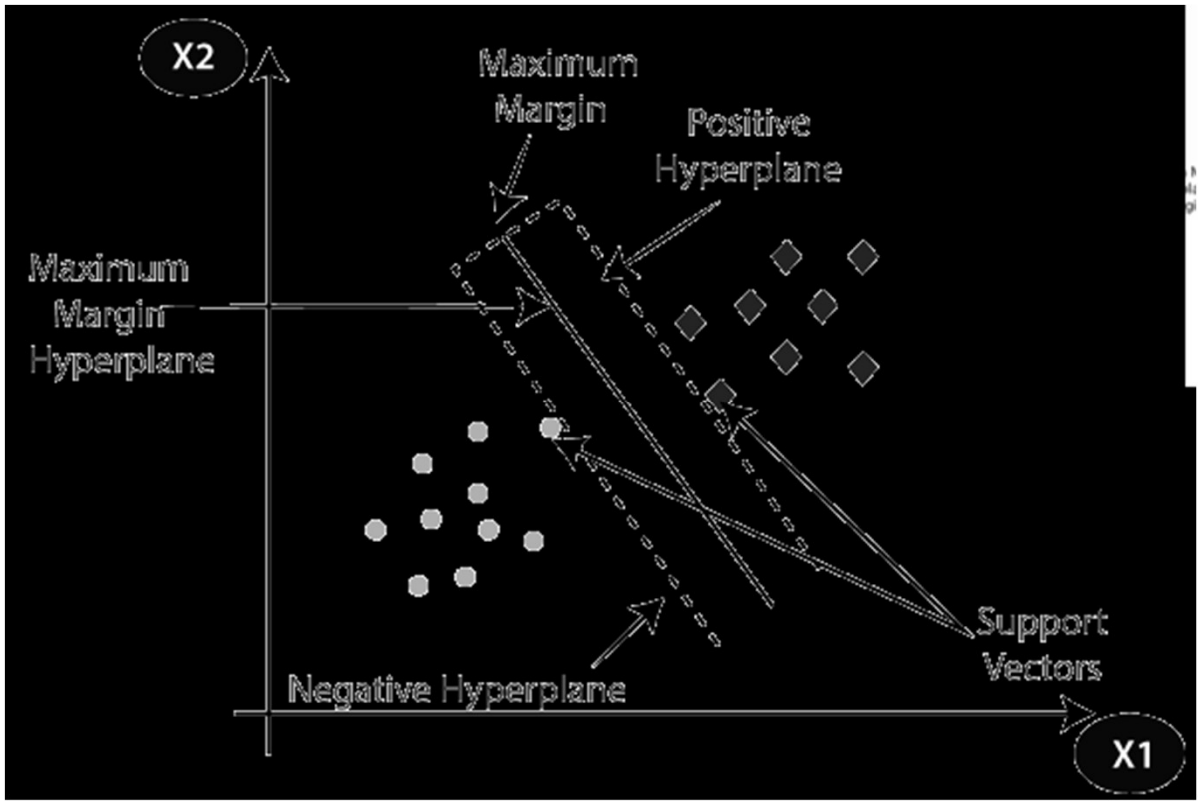
(a) Two-dimensional feature space

(b) Multi-dimensional feature space

FIG. 7.15
Linearly separable data instances

- In other words, the goal of the SVM analysis is to find a plane, or rather a hyperplane, which separates the instances on the basis of their classes.
- New examples (i.e. 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 (hyper-plane).
  - If support vectors are removed, they will alter the position of the dividing hyperplane.
- **Hyperplane and Margin**
  - For an N-dimensional feature space, hyperplane is a flat subspace of dimension (N-1) that separates and classifies a set of data.
  - For example, if we consider a two-dimensional feature space (which is nothing but a data set having two features and a class variable), a hyperplane will be a one-dimensional subspace or a straight line.



- Source : TPoint

- In the same way, for a three-dimensional feature space (data set having three features and a class variable), hyperplane is a two-dimension) subspace or a simple plane.
- However, quite understandably, it is difficult to visualize a feature space greater than three dimensions, much like for a subspace or hyperplane having more than three dimensions
- Mathematically, in a two-dimensional space, hyperplane can be defined by the equation:
- $C_0+c_1x_1+c_2x_2= 0$, which is nothing but an equation of a straight line.
- Extending this concept to an N-dimensional space, hyperplane can be defined by the equation:
- $C_0+C_1X_1+C_2X_2+...+C_NX_N=0$
- which, in short, can be represented as follows:

$$\vec{c}.\vec{X} + c_0 = 0$$

- Spontaneously, the farther (or more distance) from the hyperplane the data points lie, the more confident we can be about correct categorization.
- So, when a new testing data point/data set is added, the side of the hyperplane it lands on will decide the class that we assign to it.
- The distance between hyperplane and data points is known as margin.

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, the hyperplanes have been shown as straight lines in most of the diagrams.

- **Scenario 1**
  - As depicted in Figure 7.16, in this scenario, we have three hyperplanes: 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.

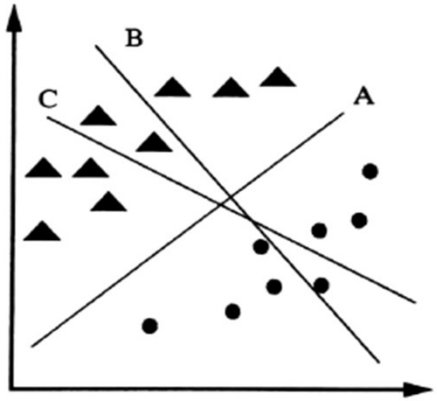


FIG. 7.16
Support vector machine: Scenario 1

- **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.

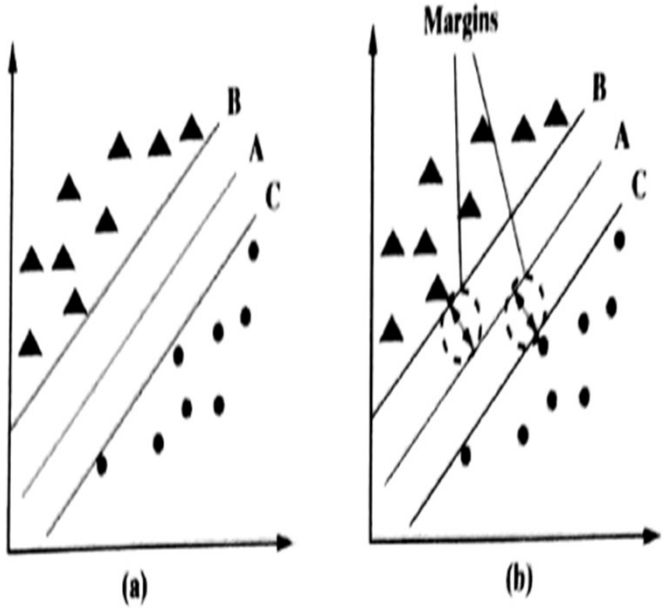


FIG. 7.17
Support vector machine: Scenario 2

- 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 there is a high probability of misclassification.

- **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.

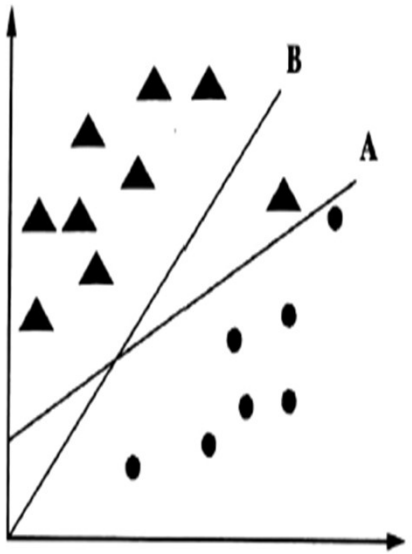


FIG. 7.18
Support vector machine: Scenario 3

- **Scenario 4**
- In this scenario, as shown in Figure 7.19a, it is not possible to distinctly segregate the two classes by using a straight line, as one data instance belonging to one of the classes (triangle) lies in the territory of the other class (circle) as an outlier.
- One triangle at the other end is like an outlier for the triangle 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.

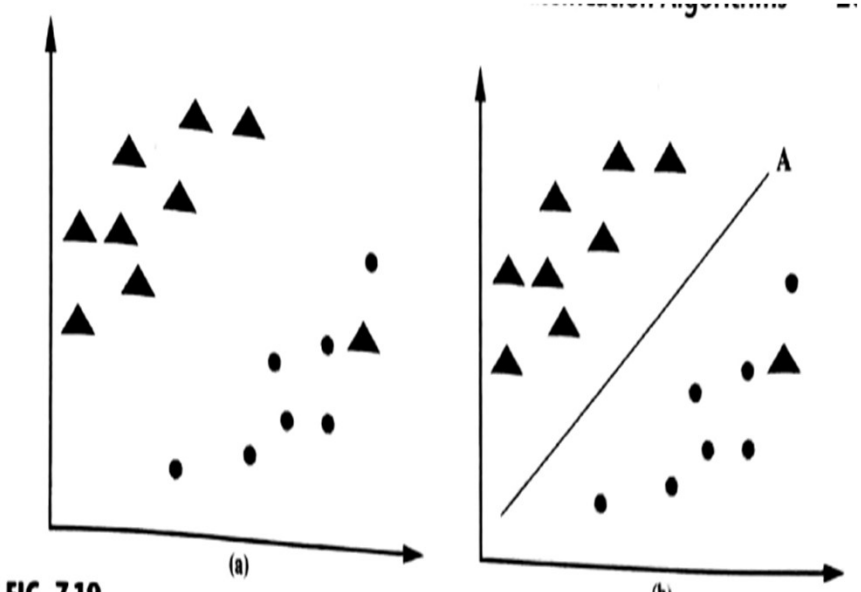


FIG. 7.19
Support vector machine: Scenario 4

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 the correct 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.

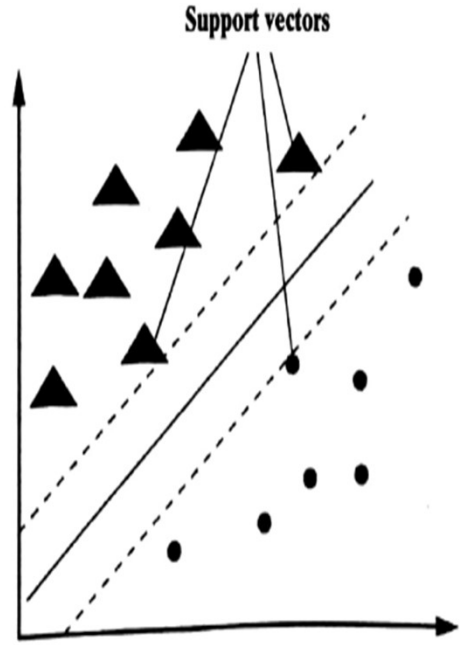


FIG. 7.20
Support vectors

- 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.
- The identification of support vectors requires intense mathematical formulation, which is out of scope of this hook.
- However, it is fairly intuitive to understand that modeling a problem using SVM is nothing but identifying the support vectors and MMH corresponding to the problem space.
- **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 the convex hulls.

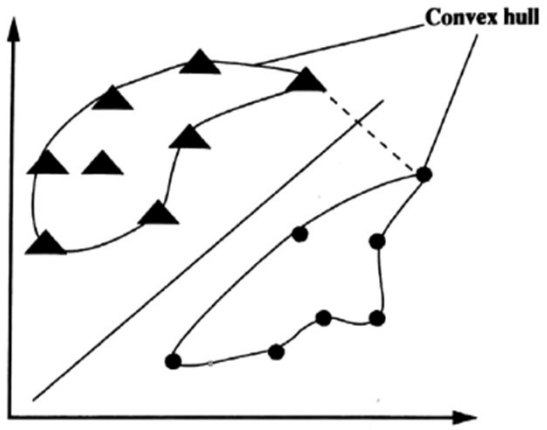


FIG. 7.21
Drawing the MMH for linearly separable data

- We have already seen earlier that a hyperplane in the 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 vectors 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 2 hyperplane.

- According to vector geometry, the distance of these planes should be 2/C.
- It is quite obvious that in order to maximize the distance between hyperplanes, the value of C should be minimized.
- So, in summary, the task of SVM is to solve the optimization problem:

$$\min\left(\frac{1}{2}\vec{c}\right)$$

- **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 E, 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.
- 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:

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

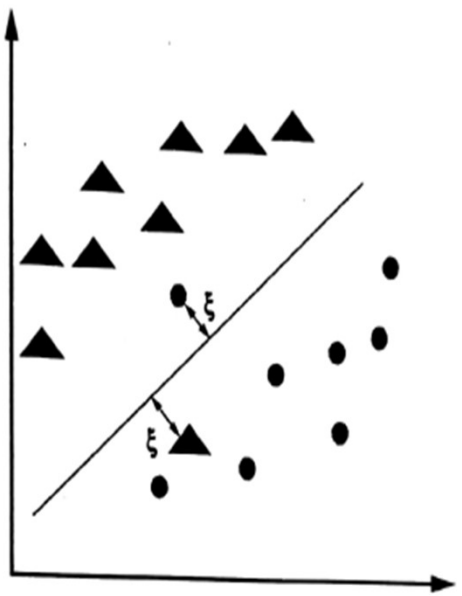


FIG. 7.22
Drawing the MMH for non-linearly separable data

- **Kernel trick**
  - As we have seen in the last section, one way to deal with non-linearly separable data is by using a slack variable and an optimization function to minimize the cost value.
  - However, this is not the only way to use SVM to solve machine learning problems involving non-linearly separable data sets.
  - 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.

- Some of the common kernel functions for transforming from a lower dimension 1' to a higher dimension used by different SVM implementations are as follows:
- Linear
- Polynomial
- Sigmoid
- Gaussian RBF kernel

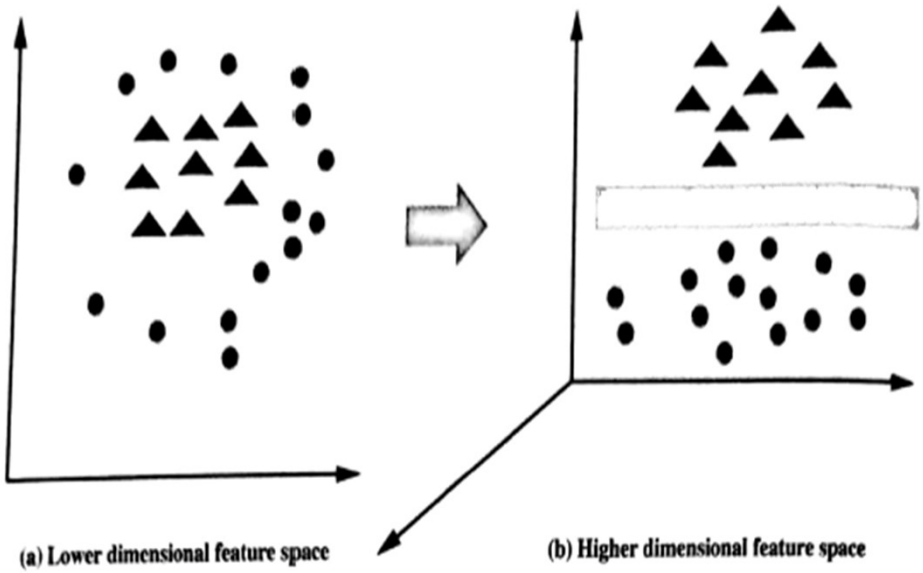


FIG. 7.23
Kernel trick in SVM

Bayes' Theorem

Bayesian theorem Introduction

- The technique 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.

WHY BAYESIAN METHODS ARE IMPORTANT?

- Bayesian learning algorithms, like the naive Bayes classifier, 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 it uses training dataset 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 for the new feature values.

- 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 in computer 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.

- Features of Bayesian learning methods :
  - 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, 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 he measured.

- 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.

BAYES' THEOREM

- Let’s be clear about what is concept learning.
- Let us take an example of how a child starts to learn meaning of new words, e.g. 'ball’.
- The child is provided with positive examples of 'objects' which are 'ball'.
- At first, the child may be confused with many different colors, shapes and sizes of the balls and may also get confused with some objects which look similar to ball, like a balloon or a globe.
- The child's parent continuously feeds her positive examples like 'that is a ball’, 'this is a green ball’, 'bring me that small ball’, etc.
- Rarely there are negative examples used for such concept teaching, like 'this is a non-ball’, but the parent may clear the confusion of the child when it points to a balloon and says it is a ball by saying 'that is not a ball'.

- But it is observed that the learning is most influenced through positive examples rather than through negative examples, and the expectation is that the child will be able to identify the object 'ball' from a wide variety of objects and different types of balls kept together once the concept of a ball is clear to her.
- We can extend this example to explain how we can expect machines to learn through the feeding of positive examples, which forms the basis for concept learning.
- To relate the above-mentioned learning concept with the mathematical model of Bayes, we can correlate the learning process of 'meaning of a word' as equivalent to learning, a concept using binary classification.
- Let us define a concept set C and a corresponding function *f(k).*
- We also define *f(k)* = 1, when k is within the set C and *f(k)* = 0 otherwise.

- So, by using the function *f*, we will be able to classify the element either inside or outside our concept set.
- In Bayes' theorem, we will learn how to use standard probability calculus to determine the uncertainty about the function *f*, and we can validate the classification by feeding positive examples.
- Bayes' probability rule can be mentioned as given below:

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

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 (set of) H by using the knowledge of D.

- **Prior**
  - The prior knowledge or belief about the probabilities of various hypotheses in *H* is called Prior .
  - It means probability of event before evidence is seen. (Before observing training data).
  - Let us introduce few notations to explain the concepts.
    - We will assume that *P(h)* is the initial probability of a hypothesis *'h'*, 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.
    - We will denote *P(T|h)* as the probability of observing data *T* in a space where *'h'* holds true.

- **Posterior**
  - The probability that a particular hypothesis holds for a data set based on the Prior is called the posterior probability or simply Posterior.
  - It means probability of event after evidence is seen.
  - After observing the training data.
  - 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*.
  - According to Bayes' theorem

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

combines the prior and posterior probabilities together.

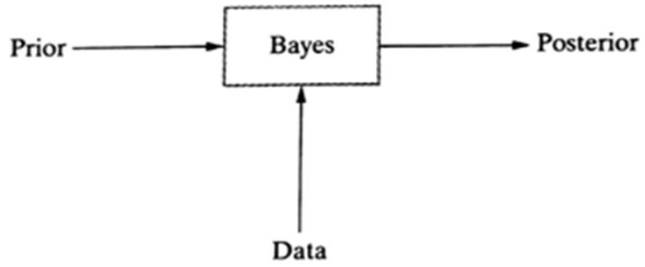


FIG. 6.1
Bayes' theorem

- From the above equation, we can deduce that *P(h|T)* increases as *P(h)* and *P(T|h)* increases and also decreases as *P(T)* decreases.
- It is a common question in machine learning problems to find out the maximum probable hypothesis *h* from a set of hypotheses *H(h∈H)* given the observed training data *T*.
- This maximally probable hypothesis is called the maximum a posteriori (MAP) hypothesis.

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

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

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

- **<u>Likelihood</u>**
  - 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 maximum likelihood (ML) hypothesis. h_{ML}.

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

  - See figure 6.2 for the conceptual and mathematical representation of Bayes theorem and the relationship of Prior, Posterior and Likelihood.

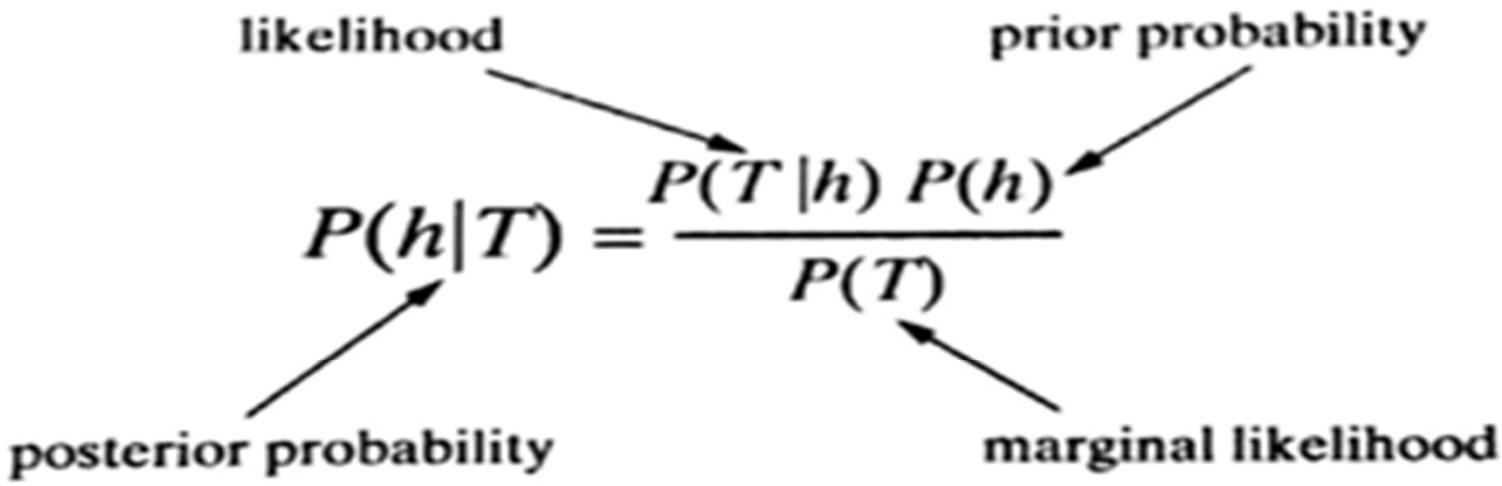


FIG. 6.2
Concept of prior, posterior, and likelihood

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.
- Here, two alternative hypotheses:
  - (1) a particular tumour is of malignant type and
  - (2) a particular tumour is non-malignant type.

- The priori available are –
  - 1. Only 0.005% 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, and Negative Lab Test = NT
  - h_1 = the particular tumour is of malignant type = MT
  - h_2 = the particular tumour is not malignant type = !MT

P(MT) = 0.005	P(!MT) = 0.995
P(PT|MT) = 0.98	P(PT|!MT) = 0.02
P(NT|!MT) = 0.97	P(NT|MT) = 0.03

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

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

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

H/w

- Predict that if the new person is having cold and flue is corona positive or negative as per the following data?
- Use Bayes' theorem for that.

Symptoms (cold/flu) ?	Covid – 19 ?	
	Yes	No
Yes	1000	20
No	10	300

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.

- **Brute-force Bayesian 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 -> {0,1} where X is the instance space corresponding to H.
  - The sequence of training examples is $\{(x_1, t_1), (x_2, t_2),..., (x_m,t_m)\}$, where xi 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 $(x_1,...,x_m)$ is held fixed, and then, the sequence of target values becomes $T = \{t_1,...,t_m\}$.
  - 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 posterior probability

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

  - 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 of the 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 each other
- 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 can write

Consistent hypothesis: a concept c that evaluates to T on all positive examples and to F on all negatives.

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

- P(T|h) is the probability of observing the target values t, in the fixed set of instances (x_i,...,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 posterior probability

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

- For the cases when h is inconsistent with the training data T, using 6.5 we get

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

when h is inconsistent with T, 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 is consistent with T as H_D, then by using the total probability equation, we get

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

- This makes 6.5 as

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

- 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 the training 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 $1/|H_D|$

Concept of consistent learners

- 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 hypotheses where 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.

Bayes optimal classifier

- Here, we will discuss the use of the MAP hypothesis to answer the question what is the most probable 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 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 $P(c_i|T)$ is maximum is
- $$\text{Bayes optimal classifier} = \underset{c_i \in C}{\text{argmax}} \sum_{h_i \in H} P(c_i|h_i)P(h_i|T)$$

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

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

- Then

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

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

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

- 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.

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 he predicted on the basis of some evidence (E) that can be observed.
- From Bayes rule, we observed that

  1. A prior probability of hypothesis h or P(h): This is the probability of an event or hypothesis before the evidence is observed.

  2. 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 cm and 68 kg, respectively.
- What is the probability that this person belongs to the class 'basketball player'?
- This can be predicted using the Naive Bayes classifier. This is known as probabilistic classifications.
- In machine learning, a probabilistic classifier is a classifier that can be predicted, given a perception or information (input), a likelihood calculation over a set of classes, instead of just yielding (outputting) the most likely class that the perception (observation) should belong to.
- Parameter estimation for Naïve Bayes models uses the method of ML.

- Bayes' theorem is used when new information can be used to revise previously determined probabilities.
- Depending on the particular nature of the probability model, Naive Bayes classifiers can be trained very professionally in a supervised learning setting.
- 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 {a1,a2..,an} 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 CMAP on the basis of the attribute values of the new instance {a1,a2..,an}.
- So,

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

- which can be rewritten using Bayes theorem as

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

- As combined probability of the attributes defining the new instance fully is always 1:

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

- So, to get the most probable classifier, we have to evaluate the two terms $P(a_1,a_2,\ldots,a_n|c_i)$ and $P(c_i)$.
- In a practical scenario, it is possible to calculate $P(c_i)$ by calculating the frequency of each target value c_i in the training data set.
- But the $P(a_1,a_2,\ldots,a_n|c_i)$ cannot he estimated easily and needs a very high effort of calculation.
- The reason is that the number of these terms is equal to the product of number of possible instances and the number of possible target values, and thus, each instance in the instance space needs to he visited many times to arrive at the estimate of the occurrence.

- Thus, the Naive 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 $a_1,a_2,..,a_n$ is the product of probabilities of individual attributes $P(a_i|c_j)$.

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

- Then, from equation 6.7, we get the approach for the Naive Bayes classifier as

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

- Here, we will be able to compute $P(a_i|c_j)$ as we have to calculate this only for the number of distinct attributes values (a_i) times the number of distinct target values (c_j), which is much smaller set than the product of both the sets.
- The most important reason for the popularity of the Naive Bayes classifier approach is that it is not required to search the whole hypothesis space for this algorithm, but rather we can arrive at the target classifier by counting the frequencies of various data combinations within the training example.
- To summarize, 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.

- 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 Naive 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 independent, this is called Naïve Bayes classifier.
- Naïve Bayes classifiers are direct linear classifiers that are known for being the straightforward, yet extremely proficient result.
- The modified version of Naive Bayes classifier originates from the assumption that information collection (data set) is commonly autonomous (mutually independent).

Table 6.1
Strengths and Weaknesses of Bayes Classifiers

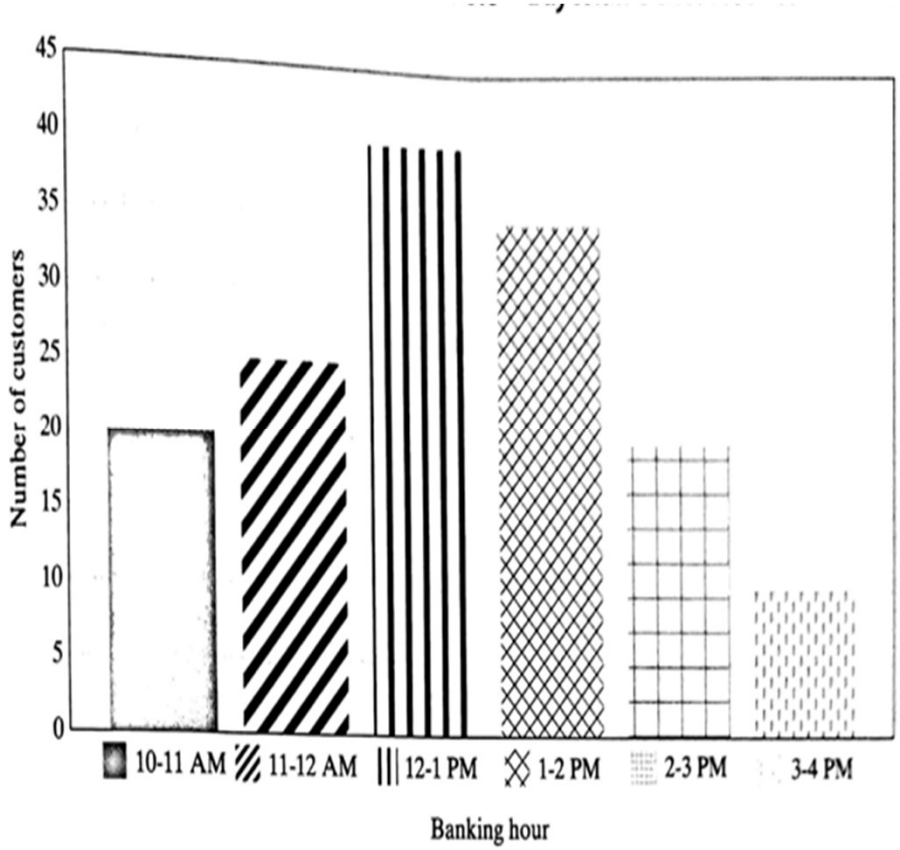
Strengths	Weakness
Simple and fast in calculation but yet effective in result	The basis assumption of equal importance and independence often does not hold true
In situations where there are noisy and missing data, it performs well	If the target dataset contains large numbers of numeric features, then 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	

Handling Continuous Numeric Features in Naive Bayes Classifier

- In the previous 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.
- In this case, the customers visit the bank continuously during banking hours and have different values for the attributes we want to evaluate before classifying them into the interested/non-interested categories.
- 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 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 shown below .
- 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.



Regression

Introduction of Regression

- The area, known as regression, focuses on solving problems such as predicting value of real estate, demand forecast in retail, weather forecast, etc.

Example of regression

- Example of regression with Karen and real estate boom in the city
- Digital market agency for selling and buying homes
- Her long experience in real estate and intuition about correct price
- Challenge of not managing customers personally and giving price quotes
- Hired assistant – but struggling with correct price quote
- Karen's friend frank – data scientist – gave solution to karen's problem for estimating price based on various parameters
- He used regression model
- So, we just discussed about one problem which can be solved using regression.
- In the same way, a bunch of other problems related to prediction of numerical value can be solved using the regression model.

- In the context of regression, dependent variable (Y) is the one whose value is to be predicted, e.g. the price quotes of the real estate in the context of Karen's problem.
- This variable is presumed to be functionally related to one (say, X) or more independent variables called predictors.
- In the context of Karen's problem, Frank used area of the property, location, floor, etc. as predictors of the model that he built.
- In other words, the dependent variable depends on independent variable(s) or predictor(s).
- Regression is essentially finding a relationship (or) association between the dependent variable (Y) and the independent variable(s) (X), i.e. to find the function 'f' for the association Y= f (X).

Common regression algorithms

- The most common regression algorithms are
  - Simple linear regression
  - Multiple linear regression
  - Polynomial regression
  - Multivariate adaptive regression splines
  - Logistic regression
  - Maximum likelihood estimation (least squares)

Simple Linear Regression

- As the name indicates, simple linear regression is the simplest regression model which involves only one predictor.
- This model assumes a linear relationship between the dependent variable and the predictor variable as shown in Figure 8.1.

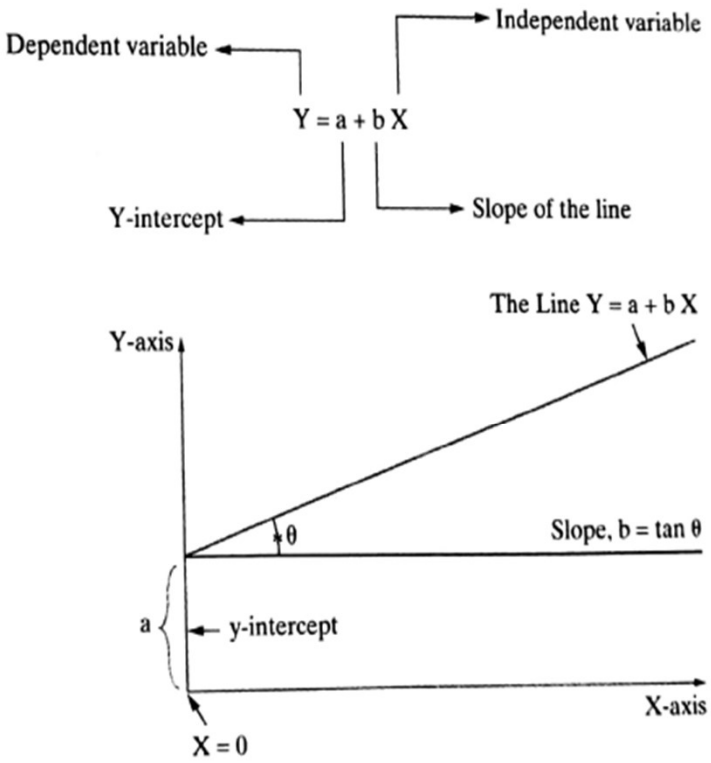


FIG. 8.1
Simple linear regression

- In the context of Karen's problem, if we take Price of a Property as the dependent variable and the Area of the Property (in sq. m.) as the predictor variable, we can build a model using simple linear regression.

$$\text{Price}_{\text{Property}} = f(\text{Area}_{\text{Property}})$$

- Assuming a linear association, we can reformulate the model as

$$\text{Price}_{\text{Property}} = a + b.\,\text{Area}_{\text{Property}}$$

- where 'a' and 'b' are intercept and slope of the straight line, respectively.
- Just to recall, straight lines can be defined in a slope-intercept form Y = a+bX, where a = intercept and b = slope of the straight line.
- The value of intercept indicates the value of Y when X = 0.

- It is known as 'the intercept or Y intercept' because it specifies where the straight line crosses the vertical or Y-axis (refer to Fig. 8.1).
- **Slope of the simple linear regression model**
  - Slope of a straight line represents how much the line in a graph changes in the vertical direction (Y-axis) over a change in the horizontal direction (X-axis) as shown in Figure 8.2.
  - Slope = Change in Y/Change in X
  - Rise is the change in Y-axis (Y2 — Y1) and Run is the change in X-axis (X2-X1). So, slope is represented as given below:

$$\text{Slope} = \text{Rise} / \text{Run}$$

$$= (Y2 — Y1) / (X2 — X1)$$

- **Example of slope**
- Let us find the slope of the graph where the lower point on the line is represented as (-3, -2) and the higher point on the line is represented as (2,2).

  (X1, Y1) = (-3, -2) and (X2, Y2) = (2,2)
  Rise = (Y2 - Y1) = (2 - (-2)) = 2 + 2 = 4
  Run = (X2 – X1) = (2 - (-3)) = 2 + 3 = 5
  Slope = Rise/Run = 4/5 = 0.8
- There can be two types of slopes in a linear regression model: positive slope and negative slope.
  - Linear positive slope
  - Curve linear positive slope
  - Linear negative slope
  - Curve linear negative slope

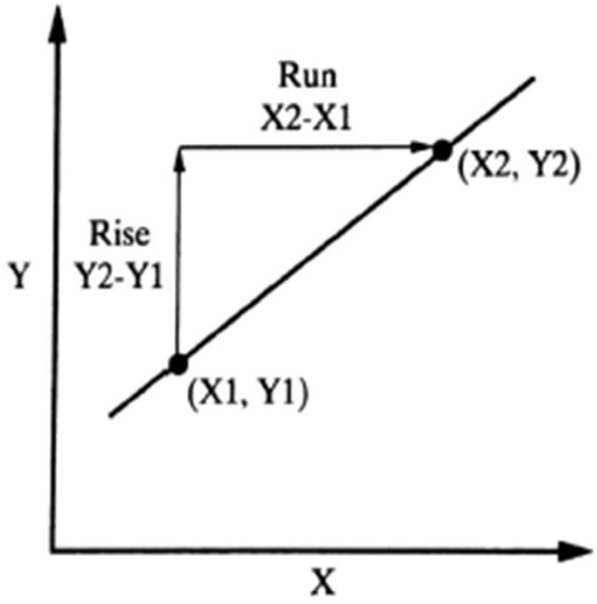


FIG. 8.2
Rise and run representation

- **Linear positive slope**
  - A positive slope always moves upward on a graph from left to right (refer to Fig.8.3).
  - Slope = Rise/Run = (Y2 — Y1) / (X2 — X1) = Delta (Y) / Delta(X)
  - Scenario 1 for positive slope: Delta (Y) is positive and Delta (X) is positive
  - Scenario 2 for positive slope: Delta (Y) is negative and Delta (X) is negative
- **Curve linear positive slope**
  - Curves in these graphs (refer to Fig. 8.4) slope upward from left to right.
  - Slope = (Y2 – Y1) / (X2 – X1)

  = Delta (Y) / Delta(X)
  - Slope for a variable (X) may vary between two graphs, but it will always be positive; hence, the graphs in next slide are called as graphs with curve linear positive slope.

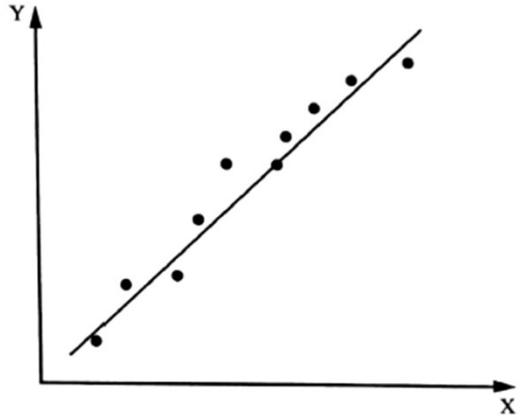


FIG. 8.3
Linear positive slope

Curve linear positive slope

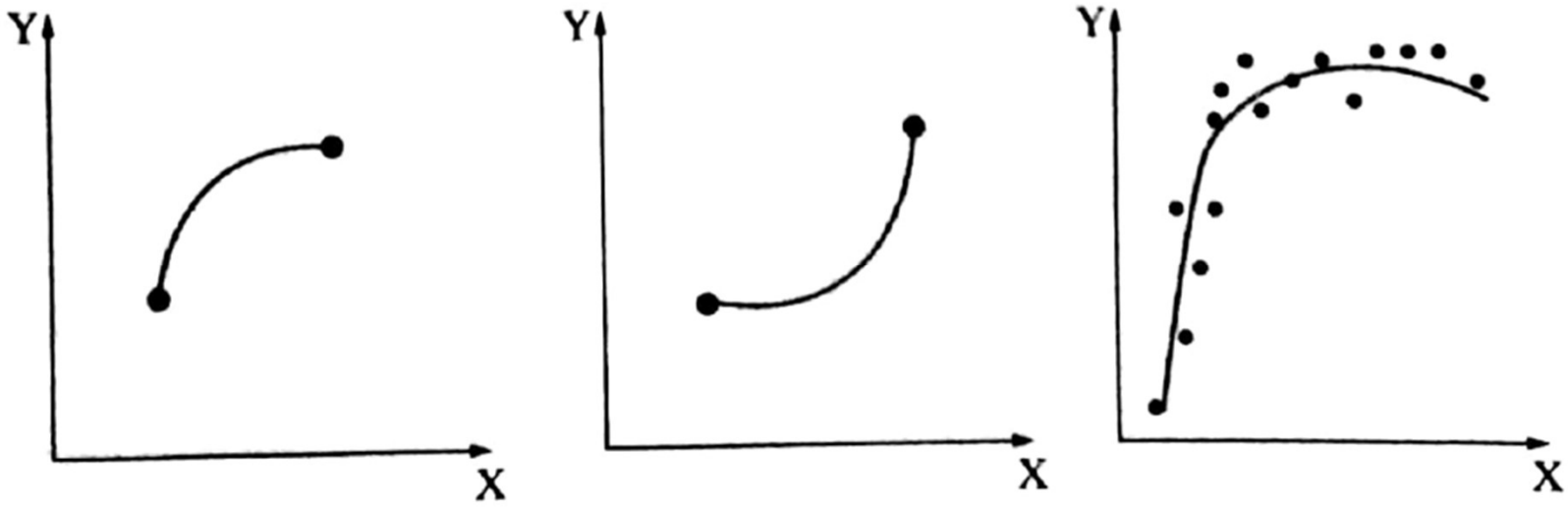


FIG. 8.4
Curve linear positive slope

- **Linear negative slope**
  - A negative slope always moves downward on a graph from left to right.
  - As a value (on X-axis) increases, Y value decreases (refer to Fig. 8.5).
  - Slope = Rise/Run = (Y2 — Y1) /(X2 — X1) = Delta (Y) / Delta(X)
  - Scenario 1 for negative slope: Delta (Y) is positive and Delta (X) is negative
  - Scenario 2 for negative slope: Delta (Y) is negative and Delta (X) is positive

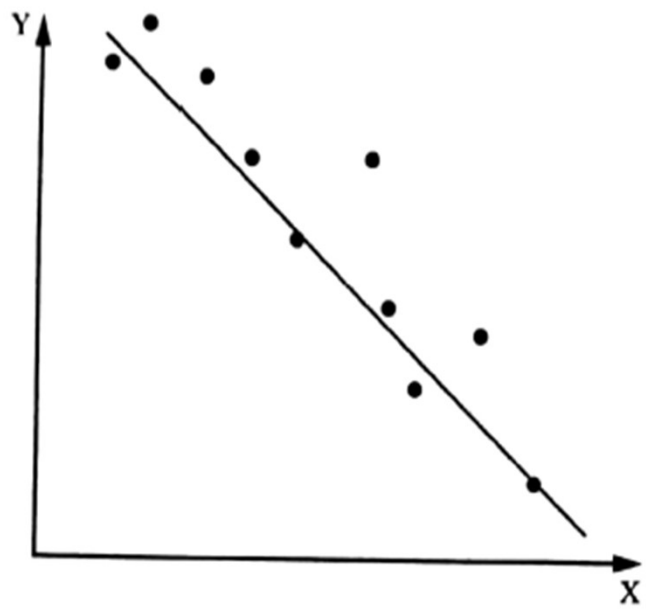


FIG. 8.5
Linear negative slope

- **Curve linear negative slope**
  - Curves in these graphs (refer to Fig. 8.6) slope downward from left to right.
  - Slope = (Y2 – Y1) / (X2 - X1) = Delta (Y) / Delta(X)
  - Slope for a variable (X) may vary between two graphs, but it will always be negative; hence, the above graphs are called as graphs with curve linear negative slope.

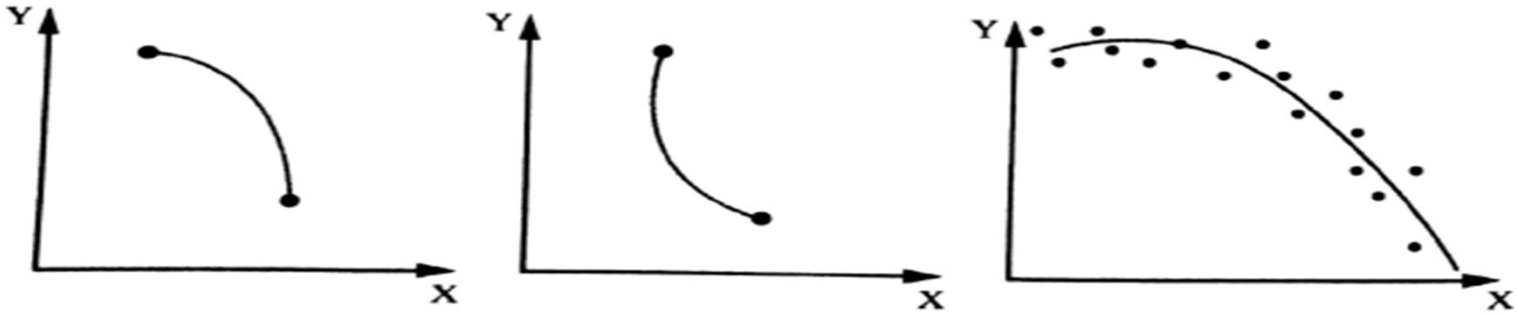


FIG. 8.6
Curve linear negative slope

- **No relationship graph**
  - Scatter graph shown in Figure 8.7 indicates 'no relationship' curve as it is very difficult to conclude whether the relationship between X and Y is positive or negative.

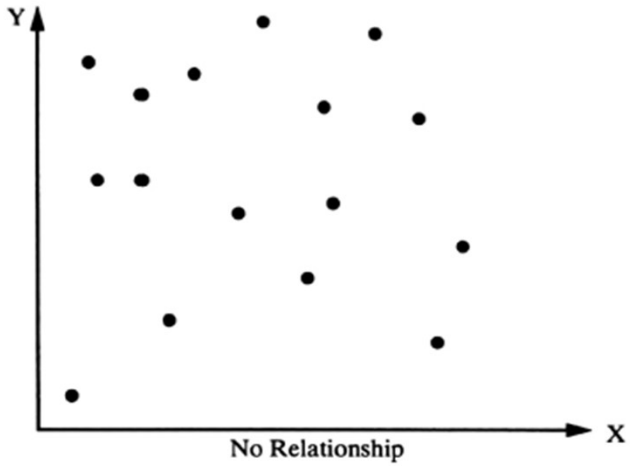


FIG. 8.7
No relationship graph

Error in simple regression

- The regression equation model in machine learning uses the above slope—intercept format in algorithm.
- X and Y values are provided to the machine, and it identifies the values of a (intercept) and b (slope) by relating the values of X and Y.
- However, identifying the exact match of values for a and b is not always possible.
- There will be some error value (E) associated with it.
- This error is called marginal or residual error.
- $Y = (a + bX) + E$
- Now that we have some context of the simple regression model, let us try to explore an example to understand clearly how to decide the parameters of the model (i.e. values of a and b) for a given problem.

Example of simple regression

- A college professor believes that if the grade for internal examination is high in a class, the grade for external examination will also be high.
- A random sample of 15 students in that class was selected, and the data is given below:

Internal Exam	15	23	18	23	24	22	22	19	19	16	24	11	24	16	23
External Exam	49	63	58	60	58	61	60	63	60	52	62	30	59	49	68

- A scatter plot was drawn to explore the relationship between the independent variable (internal marks) mapped to X-axis and dependent variable (external marks) mapped to Y-axis as depicted in Figure 8.8.

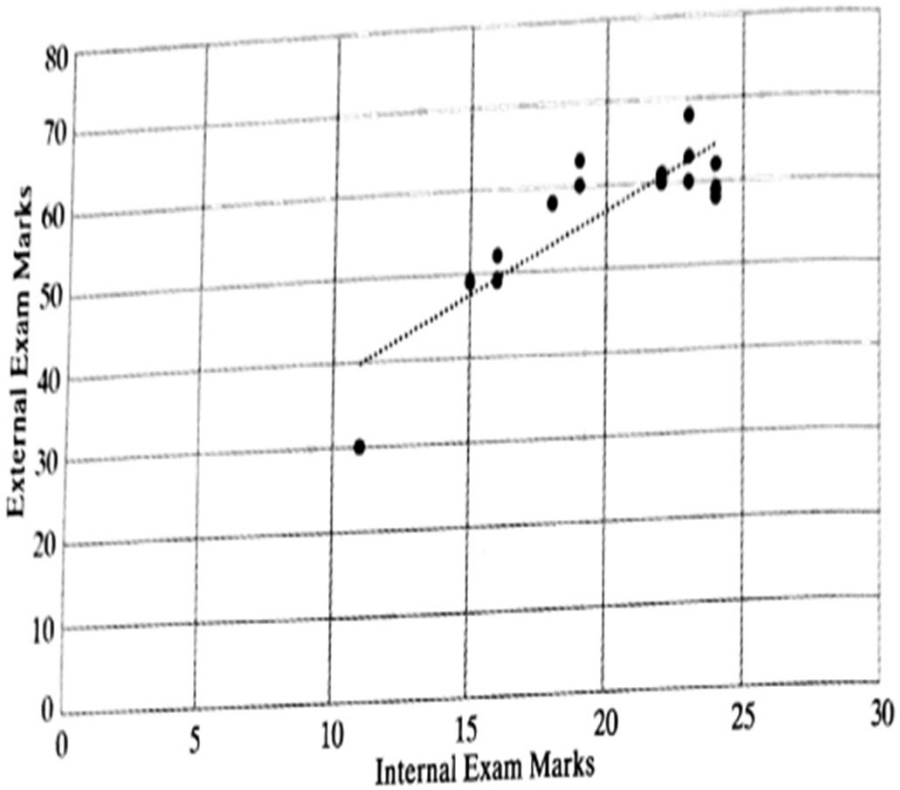


FIG. 8.8
Scatter plot and regression line

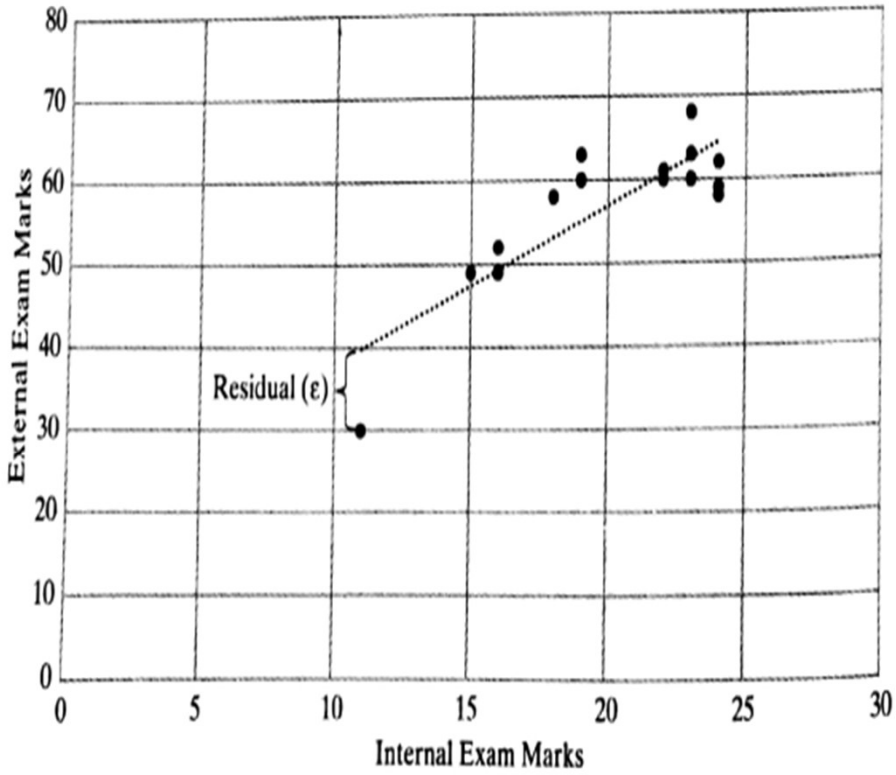


FIG. 8.9
Residual error

- As you can observe from the above graph, the line (i.e. the regression line) does not predict the data exactly (refer to Fig. 8.8).
- Instead, it just cuts through the data. Some predictions are lower than expected, while some others are higher than expected.
- Residual is the distance between the predicted point (on the regression line) and the actual point as depicted in Figure 8.9.
- As we know, in simple linear regression, the line is drawn using the regression formula.
- Y = (a + b X) + E
- If we know the values of 'a' and ‘b’, then it is easy to predict the value of Y for any given X by using the above formula.
- But the question is how to calculate the values of 'a' and ‘b' for a given set of X and Y values?

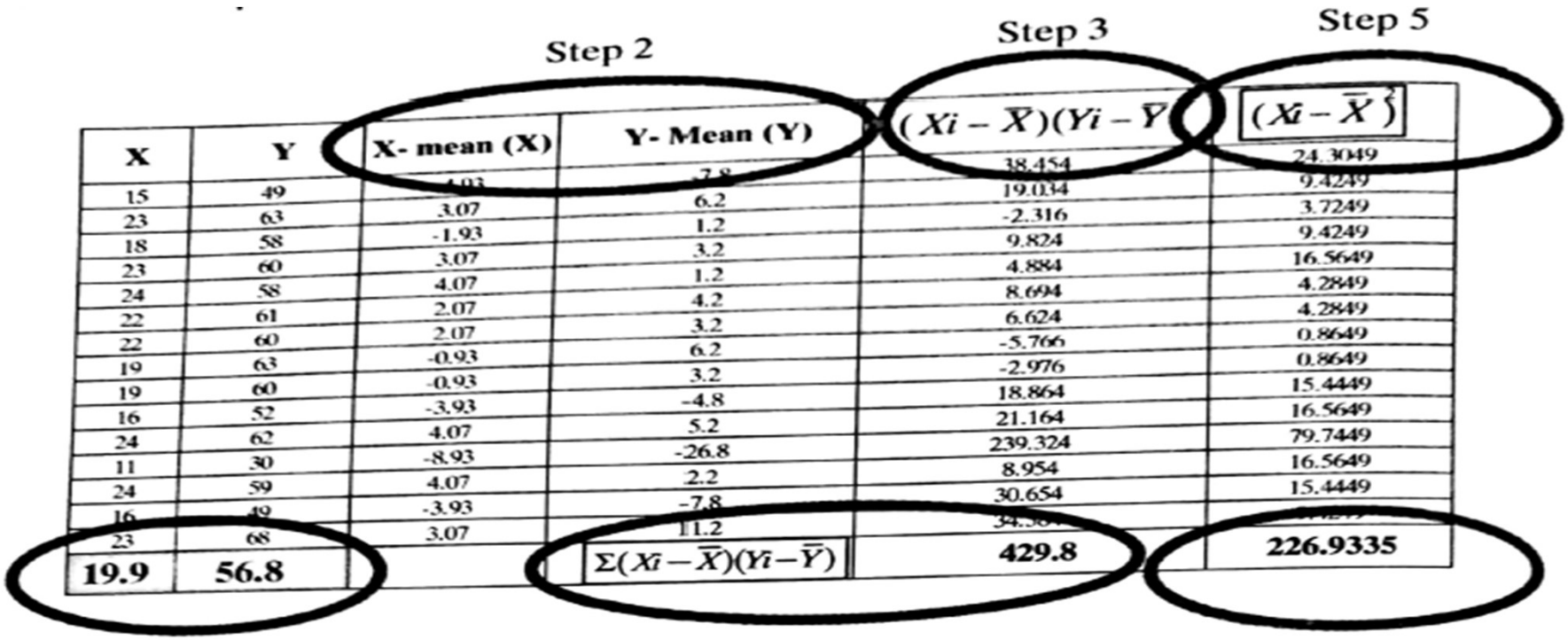
- A straight line is drawn as close as possible over the points on the scatter plot.
- Ordinary Least Squares (OLS) is the technique used to estimate a line that will minimize the error (e), which is the difference between the predicted and the actual values of Y.
- This means summing the errors of each prediction or, more appropriately, the

$$\text{Sum of the Squares of the Errors (SSE)} \left(\text{i.e. } \sum_i \varepsilon_i^2 \right).$$

- It is observed that the SSE is least when b takes the value

$$b = \frac{\Sigma_i (X_i - \overline{X})(Y_i - \overline{Y})}{\Sigma_i (X_i - \overline{X})^2} = \frac{\text{cov}(X, Y)}{\text{Var}(X)}$$

- So, let us calculate the value of a and b for the given example. For detailed calculation, refer to Figure 8.10.
-



Step 7: Divide (step4 / step6)

b = 429.28 / 226.93 = 1.89

Step 8: Calculate a using the value of b

$a = \overline{Y} - b\overline{X}$

a = 56.8 – 1.89 × 19.9

a = 19.05

Sum of $X = 299$

Sum of $Y = 852$

Mean X, $M_X = 19.93$

Mean Y, $M_Y = 56.8$

Sum of squares (SS_X) = 226.9333

Sum of products (SP) = 429.8

Regression equation = $\hat{y} = bX + a$

$$b = \frac{SP}{SS_X} = \frac{429.8}{226.93} = 1.89395$$

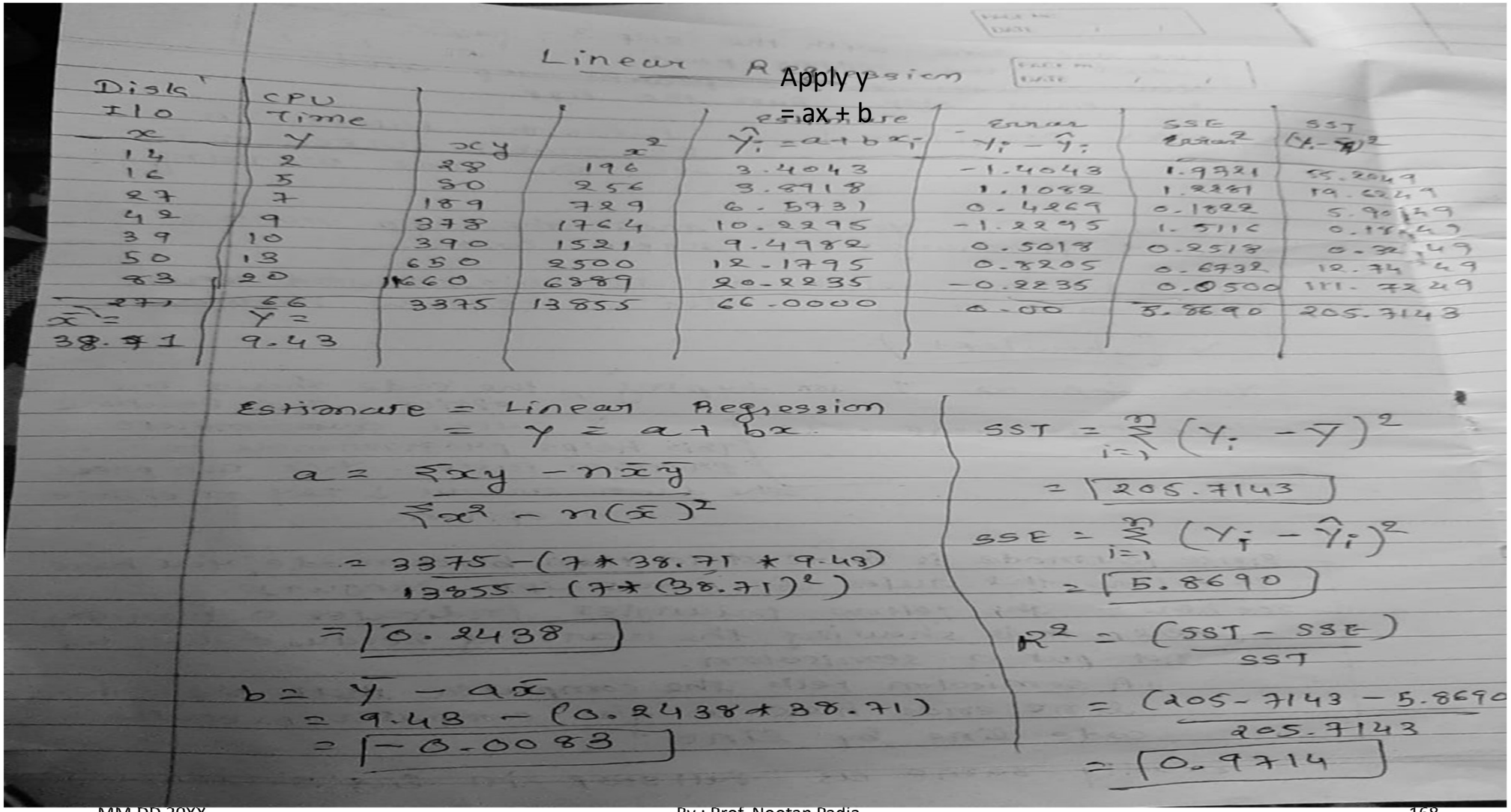
$$a = M_Y - bM_X = 56.8 - (1.89 \times 19.93) = 19.0473$$

$$\hat{y} = 1.89395X + 19.0473$$

- Hence, for the above example, the estimated regression equation is constructed on the basis of the estimated values of a and b:
- = 1.89395X + 19.0473
- So, in the context of the given problem, we can say
- Marks in external exam = 19.04 + 1.89 x (Marks in internal exam)
- or, $M_{Ext} = 19.04 + 1.89 \times M_{int}$

8.3.1.5 OLS algorithm

- Step 1: Calculate the mean of X and Y
- Step 2: Calculate the errors of X and Y
- Step 3: Get the product
- Step 4: Get the summation of the products
- Step 5: Square the difference of X
- Step 6: Get the sum of the squared difference
- Step 7: Divide output of step 4 by output of step 6 to calculate 'b'
- Step 8: Calculate 'a' using the value of 'b'



Linear Regression

Apply y
= ax + b

Disk I/O x	CPU Time y	xy	x^2	Estimate $\hat{y}_i = a + bx_i$	Error $y_i - \hat{y}_i$	SSE Error^2	SST $(y_i - \bar{y})^2$
14	2	28	196	3.4043	-1.4043	1.9721	55.2049
16	5	80	256	3.8918	1.1082	1.2281	19.6249
27	7	189	729	6.5731	0.4269	0.1822	5.9049
42	9	378	1764	10.2295	-1.2295	1.5116	0.1849
39	10	390	1521	9.4982	0.5018	0.2518	0.3249
50	13	650	2500	12.1795	0.8205	0.6732	12.7449
83	20	1660	6889	20.2235	-0.2235	0.0500	111.7249
271	66	3375	13855	66.0000	0.00	5.8690	205.7143
$\bar{x}$ = 38.71	$\bar{y}$ = 9.43						

Estimate = Linear Regression

$= y = a + bx$

$$a = \frac{\sum xy - n\bar{x}\bar{y}}{\sum x^2 - n(\bar{x})^2}$$

$$= \frac{3375 - (7 * 38.71 * 9.43)}{13855 - (7 * (38.71)^2)}$$

$$= \boxed{0.2438}$$

$$b = \bar{y} - a\bar{x}$$

$$= 9.43 - (0.2438 * 38.71)$$

$$= \boxed{-0.0083}$$

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

$$= \boxed{205.7143}$$

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$= \boxed{5.8690}$$

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

$$= \frac{(205.7143 - 5.8690)}{205.7143}$$

$$= \boxed{0.9714}$$

Multiple linear regression

- In a multiple regression model, two or more independent variables, i.e. predictors are involved in the model.
- If we think in the context of Karen's problem, in the last section, we came up with a simple linear regression by considering Price of a Property as the dependent variable and the Area of the Property (in sq. m.) as the predictor variable.
- However, location, floor, number of years since purchase, amenities available etc. are also important predictors which should not be ignored.
- Thus, if we consider Price of a Property (in $) as the dependent variable and Area of the Property (in sq. m.), location, floor, number of years since purchase and amenities available as the independent variables, we can form a multiple regression equation as shown below:
- **$Price_{property} = f(Area_{property}, location, floor, Ageing, Amenities)$**

- The simple linear regression model and the multiple regression model assume that the dependent variable is continuous.
- The following expression describes the equation involving the relationship with two predictor variables, namely X_1 and X_2.
- $Y = a + b_1X_1 + b_2X_2$
- The model describes a plane in the three-dimensional space of Y, X_1, and X_2.
- Parameter 'a' is the intercept of this plane.
- Parameters 'b_1' and 'b_2' are referred to as partial regression coefficients.
- Parameter b_1 represents the change in the mean response corresponding to a unit change in X_1, when X_2 is held constant.
- Parameter b_2 represents the change in the mean response corresponding to a unit change in X_2 when X_1 is held constant.

- Consider the following example of a multiple linear regression model with two predictor variables, namely X_1, and X_2 (refer to Fig. 8.15).

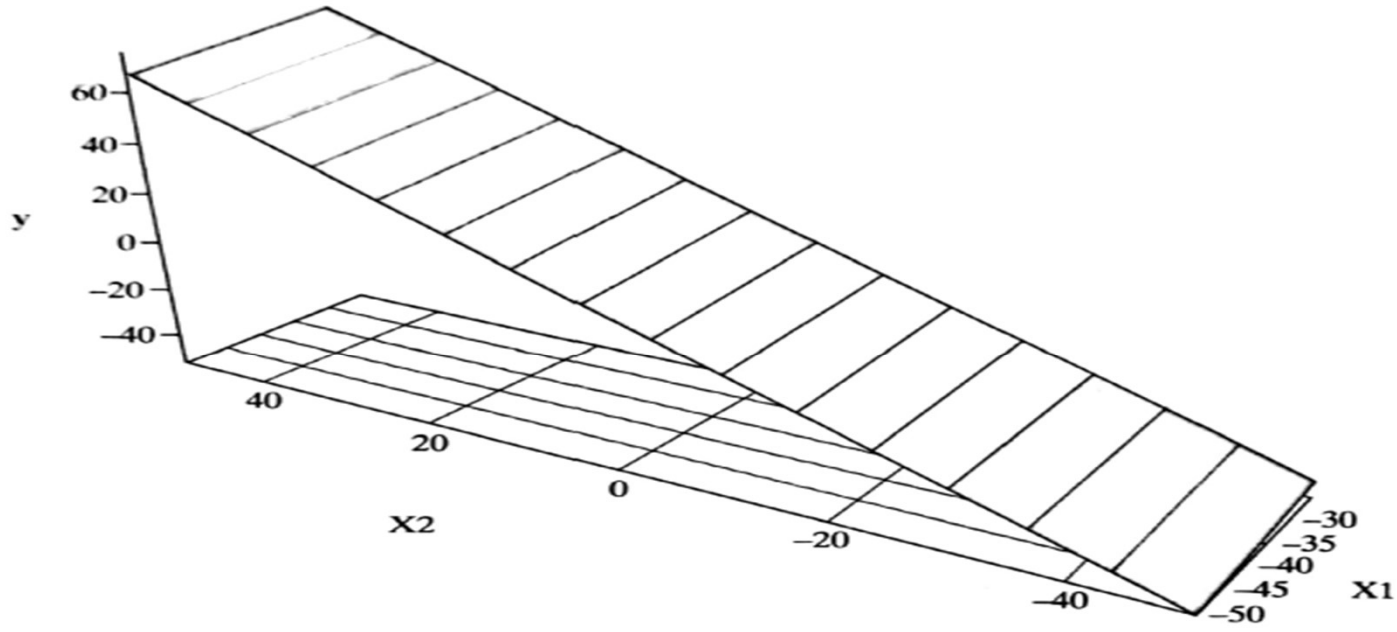


FIG. 8.15
Multiple regression plane

- Multiple regression for estimating equation when there are 'n' predictor variables is as follows:

$$\hat{Y} = a + b_1X_1 + b_2X_2 + b_3X_3 + \ \ldots \ + b_nX_n$$

- While finding the best fit line, we can fit either a polynomial or curvilinear regression.
- These are known as polynomial or curvilinear regression, respectively.

Assumptions in Regression Analysis

- The dependent variable (Y) can be calculated / predicated as a linear function of a specific set of independent variables (X's) plus an error term (e).
- The number of observations (n) is greater than the number of parameters (k) to be estimated, i.e. n > k.
- Relationships determined by regression are only relationships of association based on the data set and not necessarily of cause and effect of the defined class
- Regression line can be valid only over a limited range of data. If the line is extended (outside the range of extrapolation), it may only lead to wrong predictions.
- If the business conditions change and the business assumptions underlying the regression model are no longer valid, then the past data set will no longer be able to predict future trends.

- Variance is the same for all values of X (homoskedasticity).
- The error term (e) is normally distributed. This also means that the mean of the error (e) has an expected value of 0.
- The values of the error (e) are independent and are not related to any values of X. This means that there are no relationships between a particular X. Y that are related to another specific value of X, Y.

- Given the above assumptions, the OLS estimator is the Best Linear Unbiased Estimator (BLUE), and this is called as Gauss-Markov Theorem.

Main Problems in Regression Analysis

- In multiple regressions, there are two primary problems: multicollinearity and heteroskedasticity.
- **Multicollinearity**
  - Two variables are perfectly collinear if there is an exact linear relationship between them.
  - Multicollinearity is the situation in which the degree of correlation is not only between the dependent variable and the independent variable, but there is also a strong correlation within (among) the independent variables themselves.
  - A multiple regression equation can make good predictions when there is multicollinearity, but it is difficult for us to determine how the dependent variable will change if each independent variable is changed one at a time.
  - When multicollinearity is present, it increases the standard errors of the coefficients.

- By overinflating the standard errors, multicollinearity tries to make some variables statistically insignificant when they actually should be significant (with lower standard errors).
- One way to measure multicollinearity is to calculate the Variance Inflation Factor (VIF), which assesses how much the variance of an estimated regression coefficient increases if the predictors are correlated.
- If no factors are correlated, the VIFs will be equal to 1.
- The assumption of no perfect collinearity states that there is no exact linear relationship among the independent variables.
- This assumption implies two aspects of the data on the independent variables.
- First, none of the independent variables, other than the variable associated with the intercept term, can be a constant.
- Second, variation in the X's is necessary. In general, the more variation in the independent variables, the better will be the OLS estimates in terms of identifying the impacts of the different independent variables on the dependent variable.

- **Heteroskedasticity**
  - Heteroskedasticity refers to the changing variance of the error term.
  - If the variance of the error term is not constant across data sets, there will be erroneous predictions.
  - In general, for a regression equation to make accurate predictions, the error term should be independent, identically (normally) distributed (iid).
  - Mathematically, this assumption is written as

$$\text{var}(u_i|X) = \sigma^2 \quad \text{and}$$
$$\text{cov}(u_i u_j|X) = 0 \quad \text{for } i \neq j$$

  - Here ‘var’ represents the variance, 'cov' represents the covariance, 'u' represents the error terms, and 'X’ represents the independent variables.
  - This assumption is more commonly written as

$$\text{var}(u_i) = \sigma^2 \quad \text{and}$$
$$\text{cov}(u_i u_j) = 0 \quad \text{for } i \neq j.$$

Improving Accuracy of the Linear Regression Model

- Let us understand bias and variance in the regression model before exploring how to improve the same.
- The concept of bias and variance is similar to accuracy and prediction.
- Accuracy refers to how close the estimation is near the actual value, whereas prediction refers to continuous estimation of the value.
- High bias = low accuracy (not close to real value)
- High variance = low prediction (values are scattered)
- Low bias = high accuracy (close to real value)
- Low variance = high prediction (values are close to each other)
- Let us say we have a regression model which is highly accurate and highly predictive; therefore, the overall error of our model will be low, implying a low bias (high accuracy) and low variance (high prediction).

- This is highly preferable. Similarly, we can say that if the variance increases (low prediction), the spread of our data points increases, which results in less accurate prediction.
- As the bias increases (low accuracy), the error between our predicted value and the observed values increases.
- Therefore, balancing out bias and accuracy is essential in a regression model.
- In the linear regression model, it is assumed that the number of observations (n) is greater than the number of parameters (k) to be estimated, i.e. n > k, and in that case, the least squares estimates tend to have low variance and hence will perform well on test observations.
- However, if observations (n) is not much larger than parameters (k), then there can be high variability in the least squares fit, resulting in overfitting and leading to poor predictions.

- If k > n, then linear regression is not usable.
- This also indicates infinite variance, and so, the method cannot be used at all.
- Accuracy of linear regression can be improved using the following three methods.
    1. Shrinkage Approach
    2. Subset Selection
    3. Dimensionality (Variable) Reduction

- **Shrinkage (Regularization) approach**
  - By limiting (shrinking) the estimated coefficients, we can try to reduce the variance at the cost of a negligible increase in bias.
  - This can in turn lead to substantial improvements in the accuracy of the model.
  - Few variables used in the multiple regression model are in fact not associated with the overall response and are called as irrelevant variables: this may lead to unnecessary complexity in the regression model.
  - This approach involves fitting a model involving all predictors.
  - However, the estimated coefficients are shrunken towards zero relative to the least squares estimates.
  - This shrinkage (also known as regularization) has the effect of reducing the overall variance.

- Some of the coefficients may also be estimated to be exactly zero, thereby indirectly performing variable selection.
- The two best-known techniques for shrinking the regression coefficients towards zero are
  - ridge regression
  - lasso (Least Absolute Shrinkage Selector Operator)
- Ridge regression performs L2 regularization, i.e. it adds penalty equivalent to square of the magnitude of coefficients
- Minimization objective of ridge = LS Obj + a * (sum of square of coefficients)
- Ridge regression (include all k predictors in the final model) is very similar to least squares, except that the coefficients are estimated by minimizing a slightly different quantity.
- If k > n, then the least squares estimates do not even have a unique solution, whereas ridge regression can still perform well by trading off a small increase in bias for a large decrease in variance.

- Thus, ridge regression works best in situations where the least squares estimates have high variance.
- One disadvantage with ridge regression is that it will include all k predictors in the final model.
- This may not be a problem for prediction accuracy, but it can create a challenge in model interpretation in settings in which the number of variables k is quite large.
- Ridge regression will perform better when the response is a function of many predictors, all with coefficients of roughly equal size.
- Lasso regression performs L1 regularization, i.e. it adds penalty equivalent to the absolute value of the magnitude of coefficients.
- Minimization objective of ridge = LS Obj + a * (absolute value of the magnitude of coefficients)
- The lasso overcomes this disadvantage by forcing some of the coefficients to zero value.
- We can say that the lasso yields sparse models (involving only subset) that are simpler as well as more interpretable.
- The lasso can be expected to perform better in a setting where a relatively small number of predictors have substantial coefficients, and the remaining predictors have coefficients that are very small or equal to zero.

- **Subset selection**
  - Identify a subset of the predictors that is assumed to be related to the response and then fit a model using OLS on the selected reduced subset of variables.
  - There are two methods in which subset of the regression can be selected:
    - 1. Best subset selection (considers all the possible (2^k))
    - 2. Stepwise subset selection
      - (a) Forward stepwise selection (0 to k)
      - (b) Backward stepwise selection (k to 0)
  - In best subset selection, we fit a separate least squares regression for each possible subset of the k predictors.
  - For computational reasons, best subset selection cannot be applied with very large value of predictors (k).

- The best subset selection procedure considers all the possible (2^k) models containing subsets of the predictors.
- The stepwise subset selection method can he applied to choose the best subset.
- There are two stepwise subset selection:
  - 1. Forward stepwise selection (0 to k)
  - 2. Backward stepwise selection (k to 0)
- Forward stepwise selection is a computationally efficient alternative to best subset selection.
- Forward stepwise considers a much smaller set of models, that too step by step, compared to best set selection.
- Forward stepwise selection begins with a model containing no predictors, and then, predictors are added one by one to the model, until all the k predictors are included in the model.
- In particular, at each step, the variable (X) that gives the highest additional improvement to the fit is added.
- Backward stepwise selection begins with the least squares model which contains all k predictors and then iteratively removes the least useful predictor one by one.

- **Dimensionality reduction (Variable reduction)**
  - The earlier methods, namely subset selection and shrinkage, control variance either by using a subset of the original variables or by shrinking their coefficients towards zero.
  - In dimensionality reduction, predictors (X) are transformed, and the model is set up using the transformed variables after dimensionality reduction.
  - The number of variables is reduced using the dimensionality reduction method.
  - Principal component analysis is one of the most important dimensionality (variable) reduction techniques.

Polynomial regression model

- Polynomial regression model is the extension of the simple linear model by adding extra predictors obtained by raising (squaring) each of the original predictors to a power.
- For example, if there are three variables, X, X^2, and X^3 are used as predictors.
- This approach provides a simple way to yield a non-linear fit to data.

  $f(x) = c_0 + c_1.X^1 + c_2.X^2 + c_3.X^3$
- In the above equation, c_0, c_1, c_2, and c_3 are the coefficients.
- Example Let us use the below data set of (X, Y) for degree 3 polynomial.

Internal Exam (*X*)	15	23	18	23	24	22	22	19	19	16	24	11	24	16	23
External Exam (*Y*)	49	63	58	60	58	61	60	63	60	52	62	30	59	49	68

- As you can observe, the regression line (refer to Fig. 8.16) is slightly curved for polynomial degree 3 with the above 15 data points.
- The regression line will curve further if we increase the polynomial degree (refer to Fig. 8.17).
- At the extreme value as shown below, the regression line will be overfitting into all the original values of X.

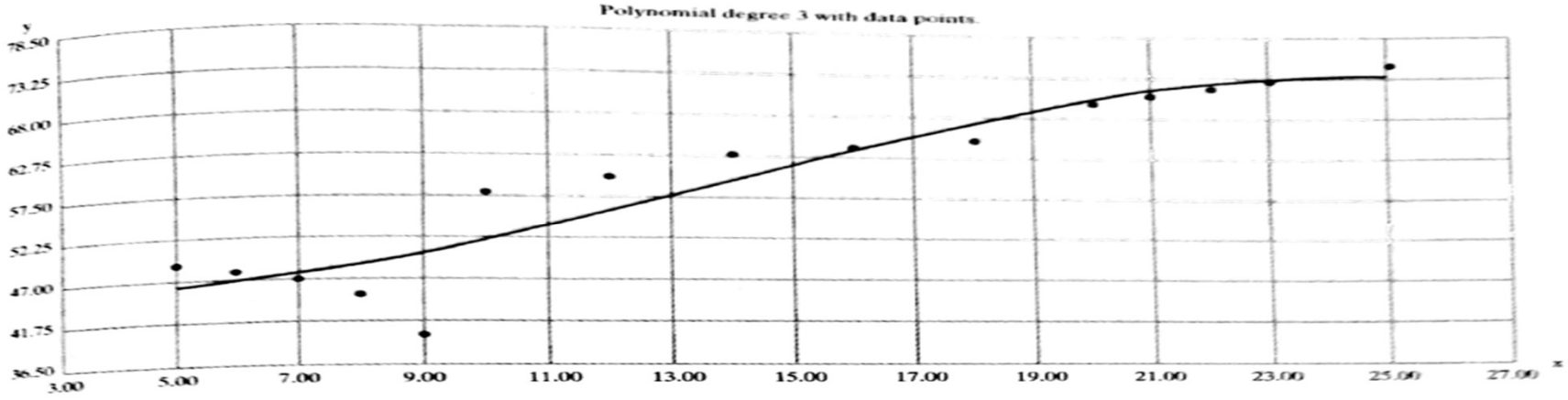


FIG. 8.16
Polynomial regression degree 3

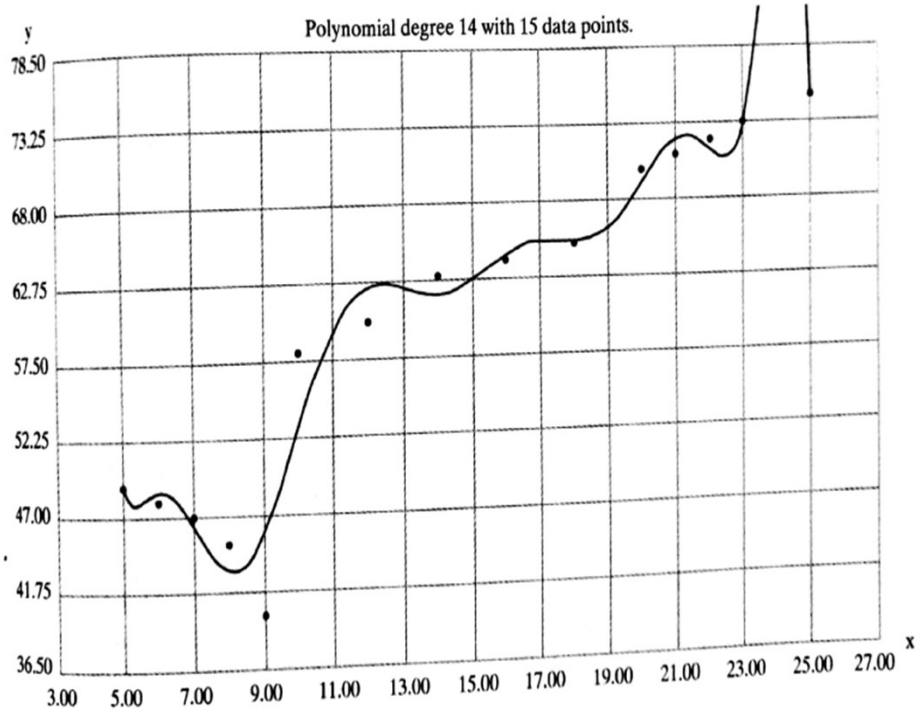
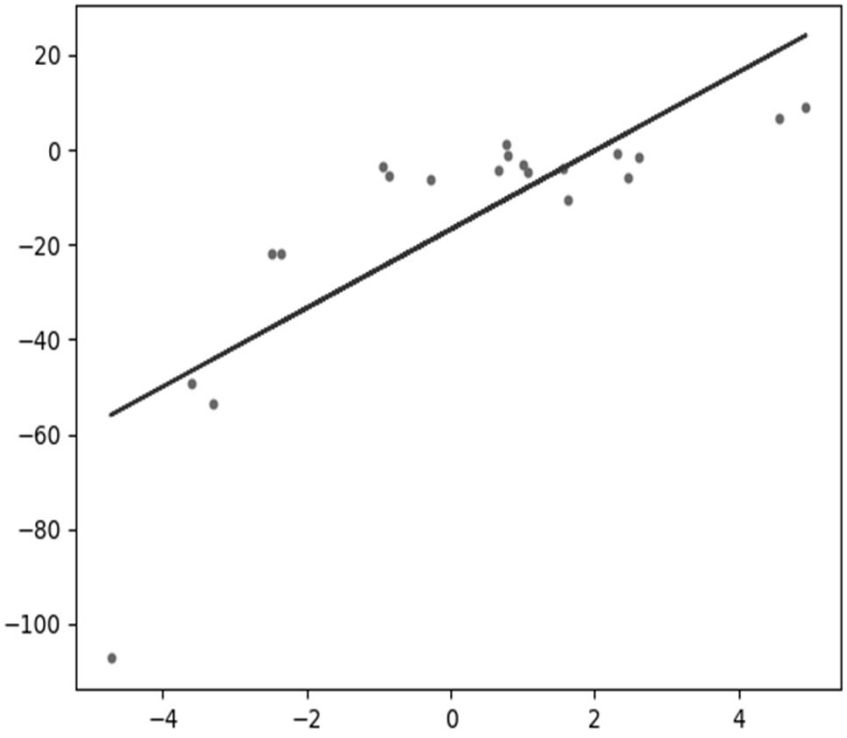
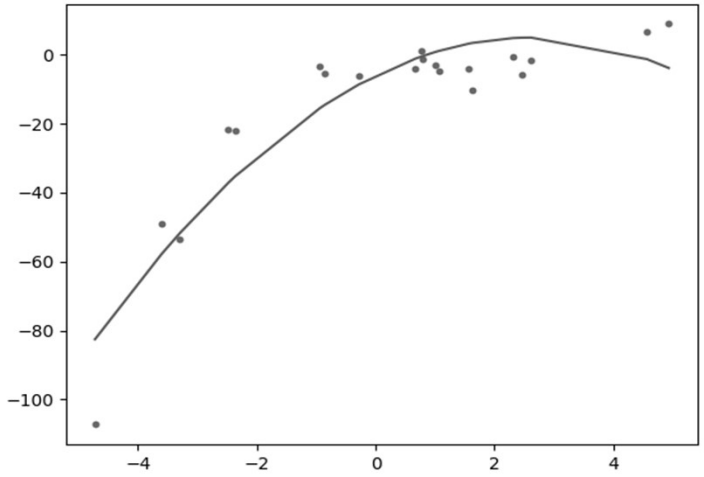
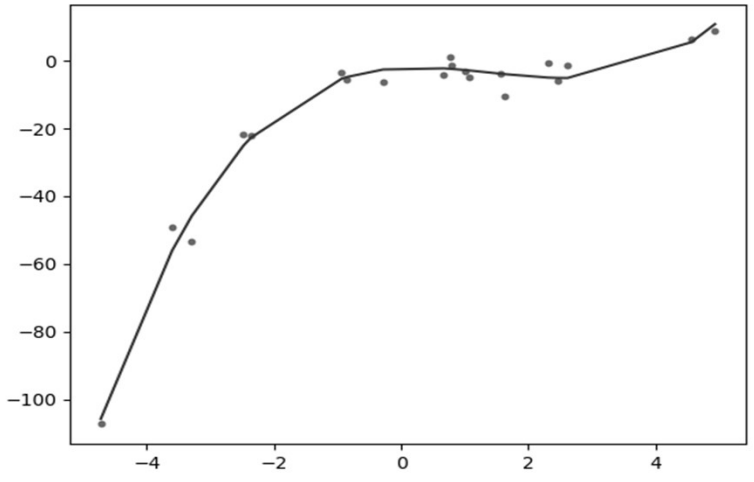
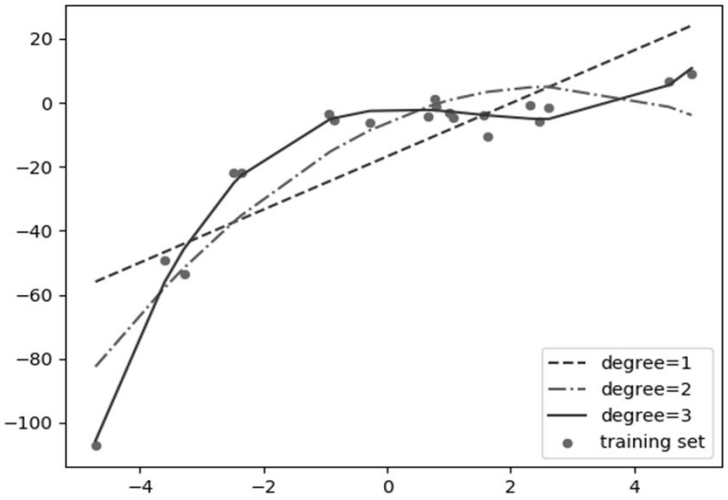


FIG. 8.17
Polynomial regression degree 14

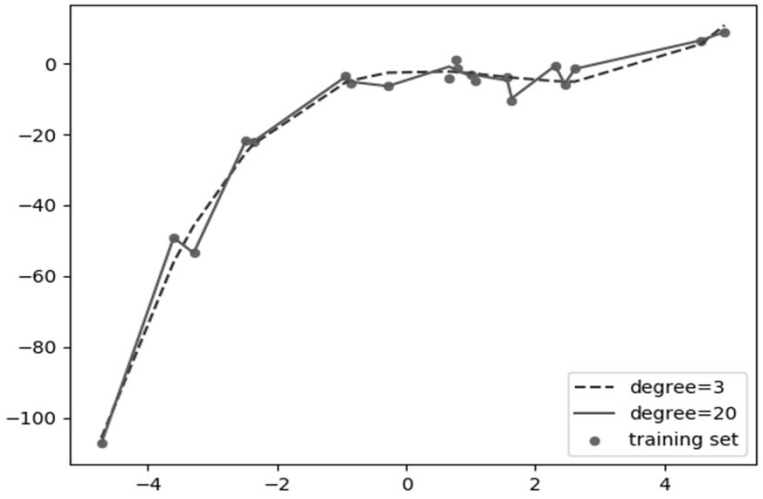


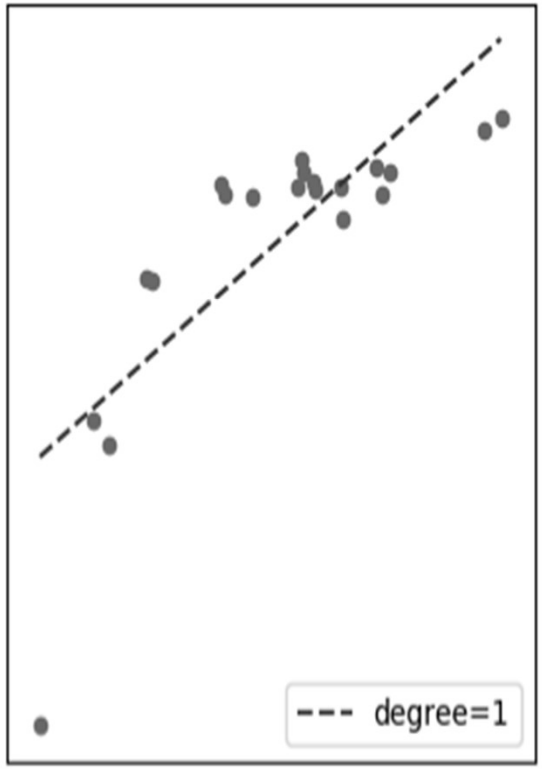




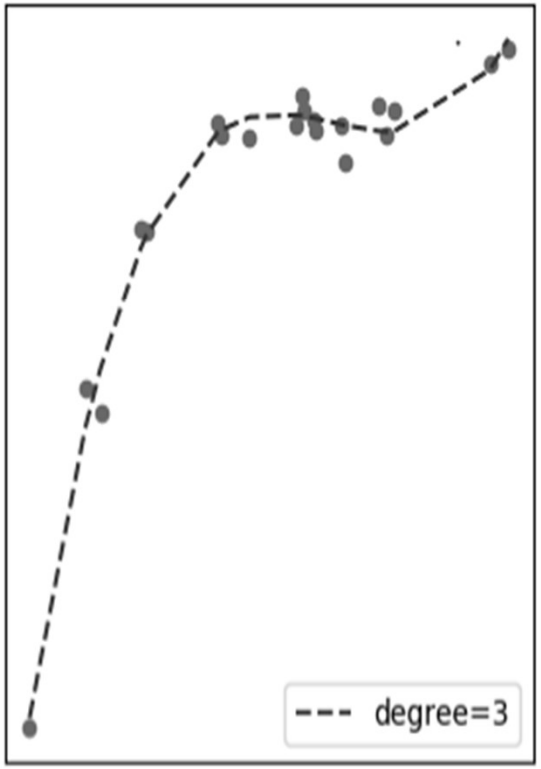


y : Prof. Nootan P

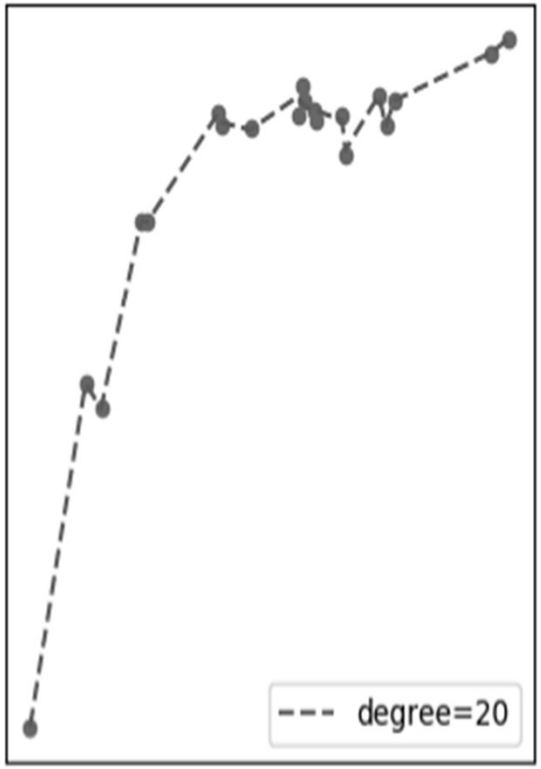




Underfit
High Bias
Low Variance



Correct Fit
Low Bias
Low Variance



Overfit
Low Bias
High Variance

Advantages of using Polynomial Regression:

- Polynomial provides the best approximation of the relationship between the dependent and independent variable.
- A Broad range of function can be fit under it.
- Polynomial basically fits a wide range of curvature.

Disadvantages of using Polynomial Regression

- The presence of one or two outliers in the data can seriously affect the results of the nonlinear analysis.
- These are too sensitive to the outliers.
- In addition, there are unfortunately fewer model validation tools for the detection of outliers in nonlinear regression than there are for linear regression.

Logistic Regression

- Logistic regression is both classification and regression technique depending on the scenario used.
- Logistic regression (logit regression) is a type of regression analysis used for predicting the outcome of a categorical dependent variable similar to OLS (ordinary least squares) regression.
- In logistic regression, dependent variable (Y) is binary (0,1) and independent variables (X) are continuous in nature.
- The probabilities describing the possible outcomes (probability that Y = 1) of a single trial are modeled as a logistic function of the predictor variables.
- In the logistic regression model, there is no R^2 to gauge the fit of the overall model; however, a chi-square test is used to gauge how well the logistic regression model fits the data.
- The goal of logistic regression is to predict the likelihood that Y is equal to 1 (probability that Y = 1 rather than 0) given certain values of X.

- That is, if X and Y have a strong positive linear relationship, the probability that a person will have a score of Y = 1 will increase as values of X increase.
- So, we are predicting probabilities rather than the scores of the dependent variable.
- For example, we might try to predict whether or not a small project will succeed or fail on the basis of the number of years of experience of the project manager handling the project.
- We presume that those project managers who have been managing projects for many years will be more likely to succeed.
- This means that as X (the number of years of experience of project manager) increases, the probability that Y will be equal to 1 (success of the new project) will tend to increase.

- If we take a hypothetical example in which 60 already executed projects were studied and the years of experience of project managers ranges from 0 to 20 years, we could represent this tendency to increase the probability that Y = 1 with a graph.
- To illustrate this, it is convenient to segregate years of experience into categories (i.e. 0-8, 9-16, 17-24, 25-32, 33-40).
- If we compute the mean score on Y (averaging the 0s and 1s) for each category of years of experience, we will get something like

X	Y
0–8	0.27
9–16	0.5
17–24	0.6
25–32	0.66
33–40	0.93

- When the graph is drawn for the above values of X and Y, it appears like the graph in Figure 8.18.
- As X increases, the probability that Y = 1 increases.
- In other words, when the project manager has more years of experience, a larger percentage of projects succeed.
- A perfect relationship represents a perfectly curved S rather than a straight line, as was the case in OLS regression.
- So, to model this relationship, we need some fancy algebra / mathematics that accounts for the bends in the curve.
- An explanation of logistic regression begins with an explanation of the logistic function, which always takes values between zero and one.
- The logistic formulae are stated in terms of the probability that Y = 1, which is referred to as P.

- The probability that Y is 0 is 1 — P.

$$\ln\left(\frac{P}{1-P}\right) = a + bX$$

$$\ln(p/1-p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \varepsilon$$

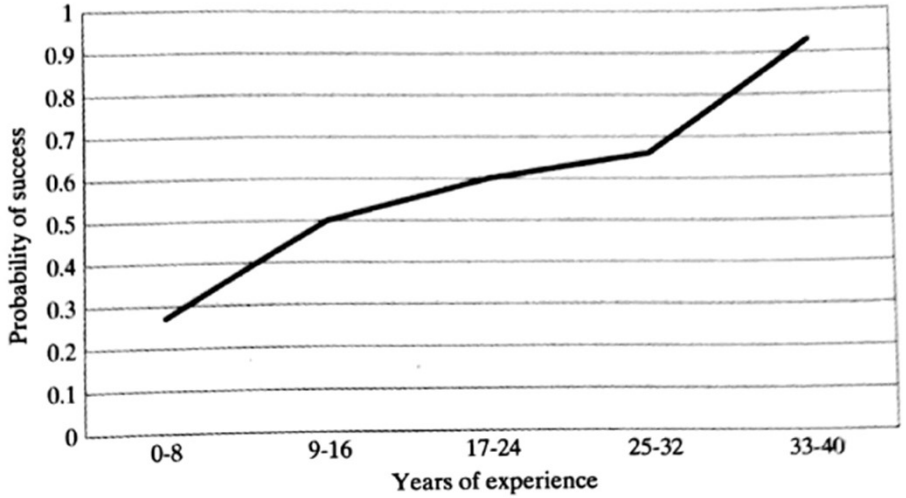


FIG. 8.18
Logistic regression

- The 'In' symbol refers to a natural logarithm and a + bX is the regression line equation.
- Probability (P) can also be computed from the regression equation.
- So, if we know the regression equation, we could, theoretically, calculate the expected probability that Y =1 for a given value of X.

$$P = \frac{\exp(a + bX)}{1 + \exp(a + bx)} = \frac{e^{a+bx}}{1 + e^{a+bx}}$$

'exp' is the exponent function, which is sometimes also written as e.

- Let us say we have a model that can predict whether a person is male or female on the basis of their height.
- Given a height of 150 cm, we need to predict whether the person is male or female.

- We know that the coefficients of a = —100 and b = 0.6.
- Using the above equation, we can calculate the probability of male given a height of 150 cm or more formally P(male|height = 150).

$$y = e\text{^}(a + b \times X)/(1 + e\text{^}(a + b \times X))$$
$$\text{y} = \text{exp}\,(-100 + 0.6 \times 150)/(1 + \text{EXP}(-100 + 0.6 \times X)$$
$$y = 0.000046$$

- or a probability of near zero that the person is a male.

Assumptions in logistic regression

- The following assumptions must hold when building a logistic regression model:
- There exists a linear relationship between logit function and independent variables
- The dependent variable Y must be categorical (1/0) and take binary value, e.g. if pass then Y =1; else Y = 0
- The data meets the 'iid' criterion, i.e. the error terms, E, are independent from one another and identically distributed
- The error term follows a binomial distribution [n, p]
- n = # of records in the data
- p = probability of success (pass, responder)