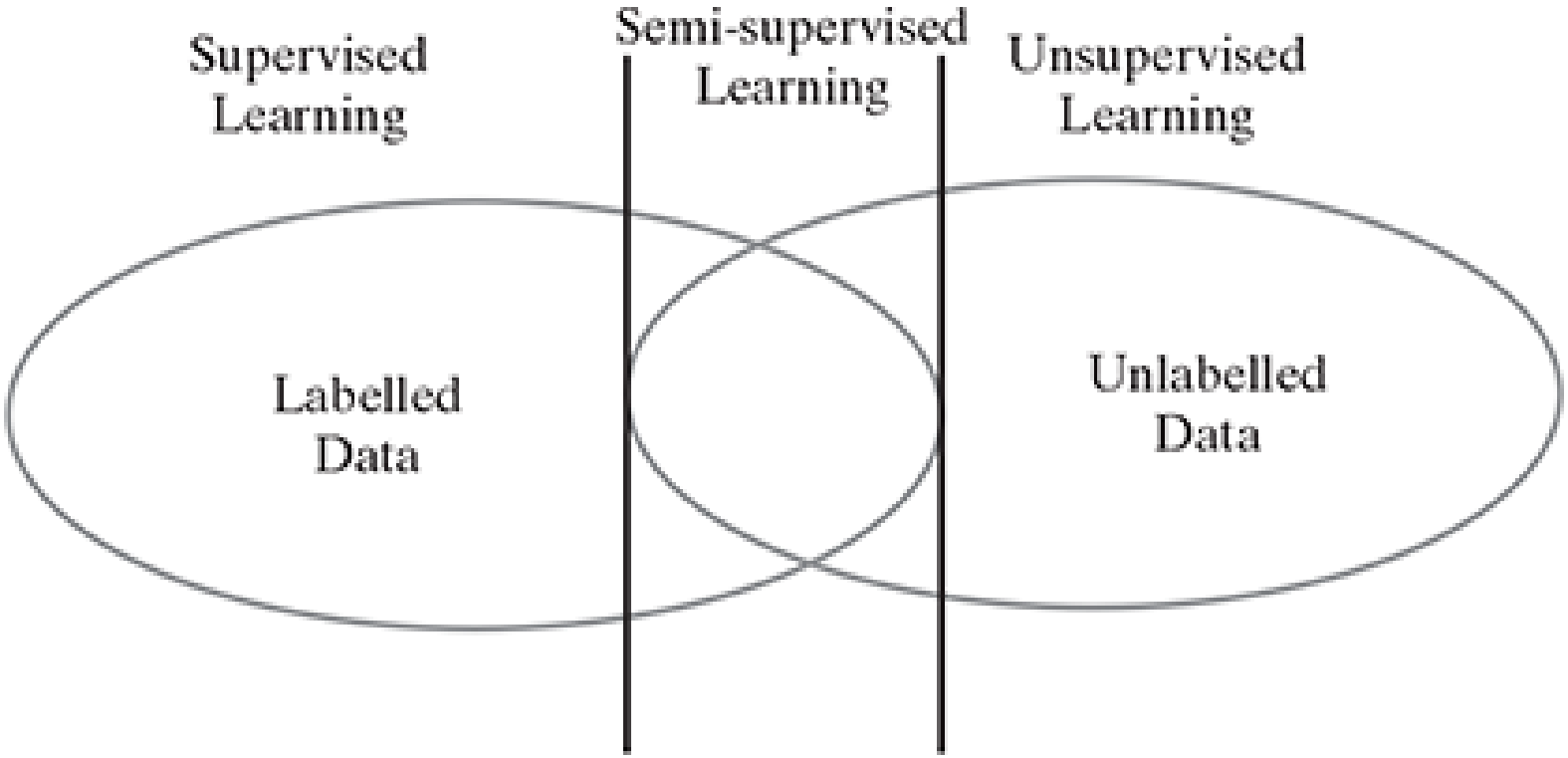
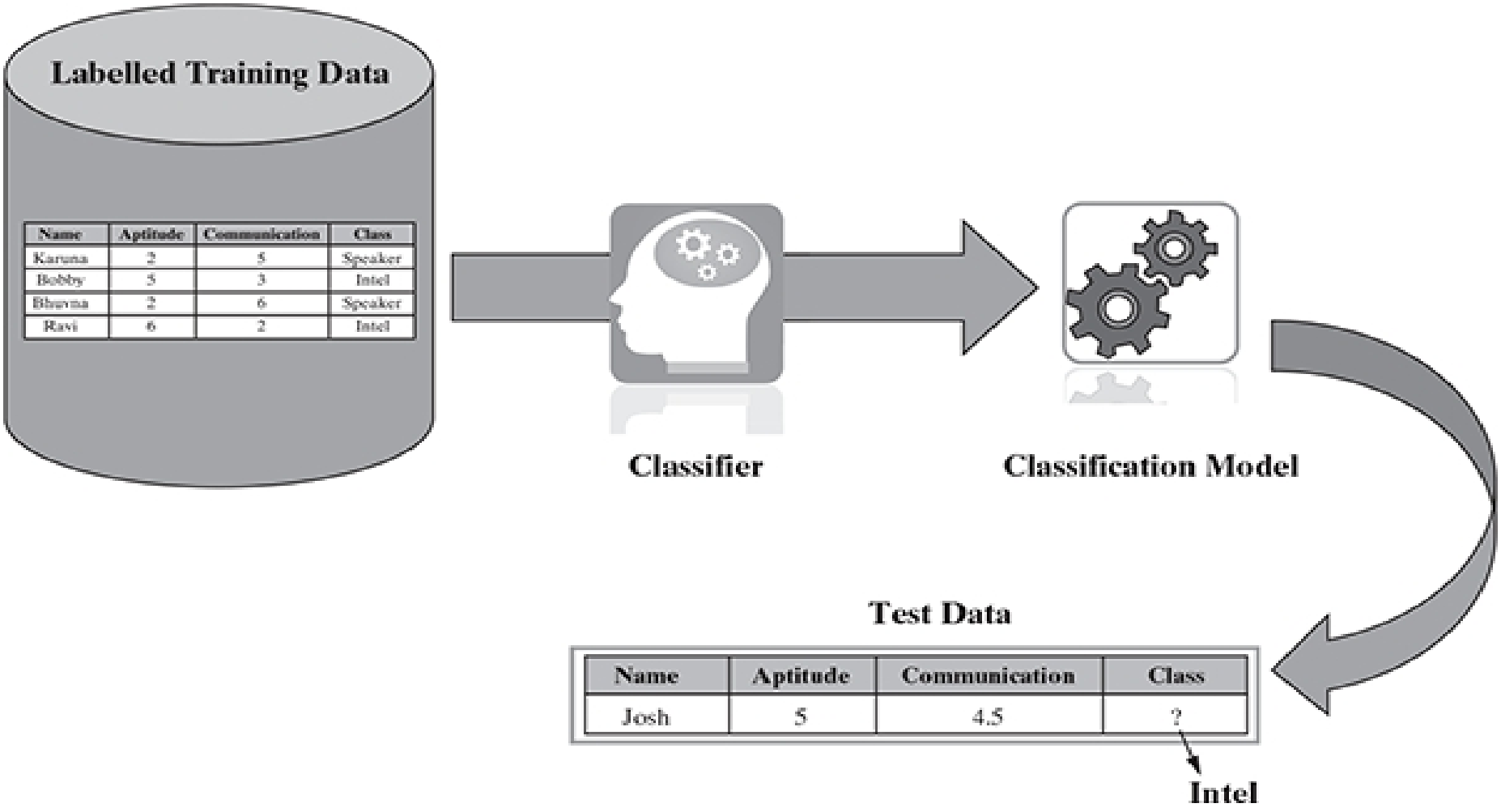
Chapter – 7

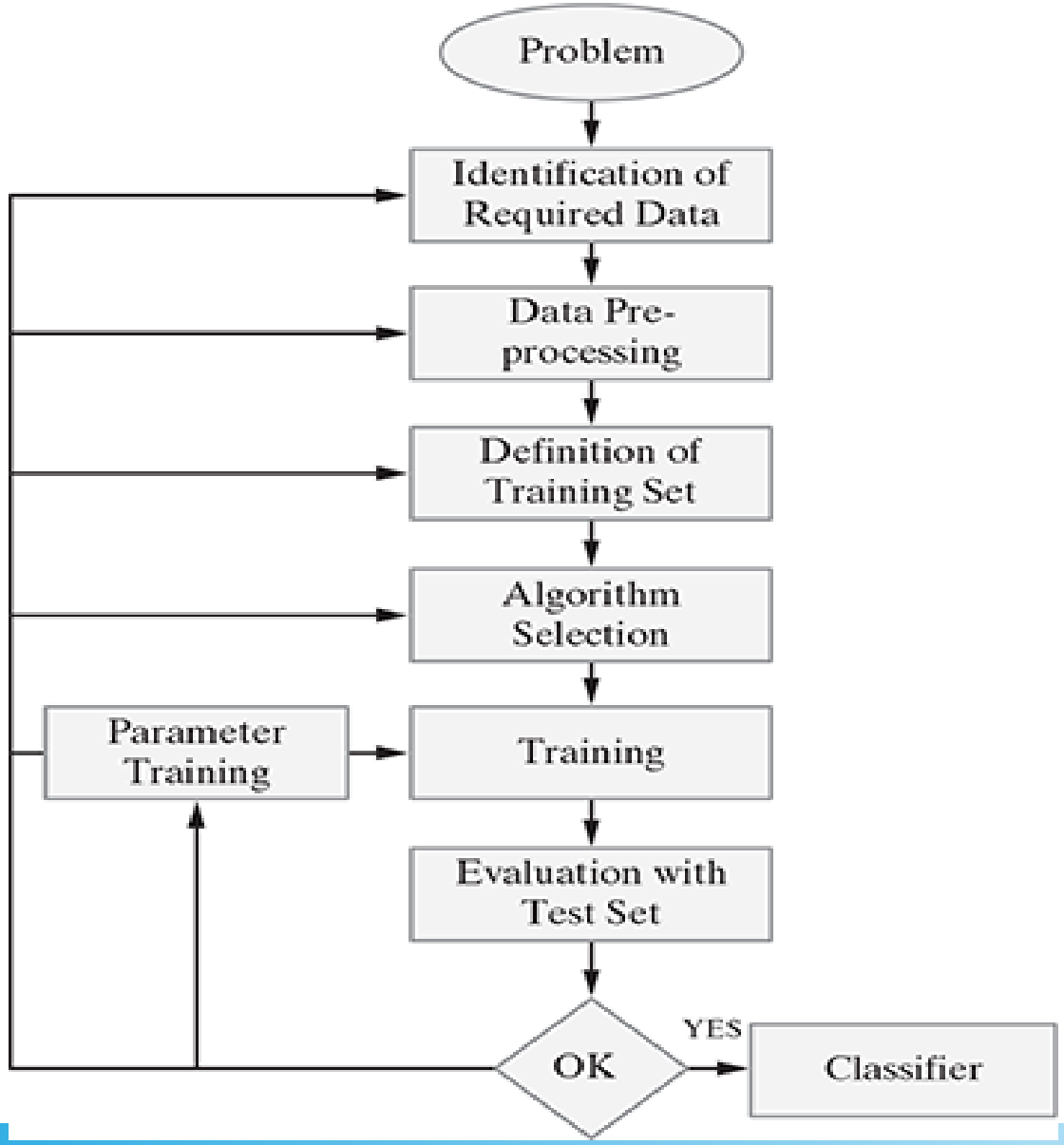
A study material for the students of GLS University
Compiled by Dr. Krupa Mehta


Supervised Learning
Semi-supervised Learning
Unsupervised Learning
Labelled Data
Unlabelled Data

Classification model



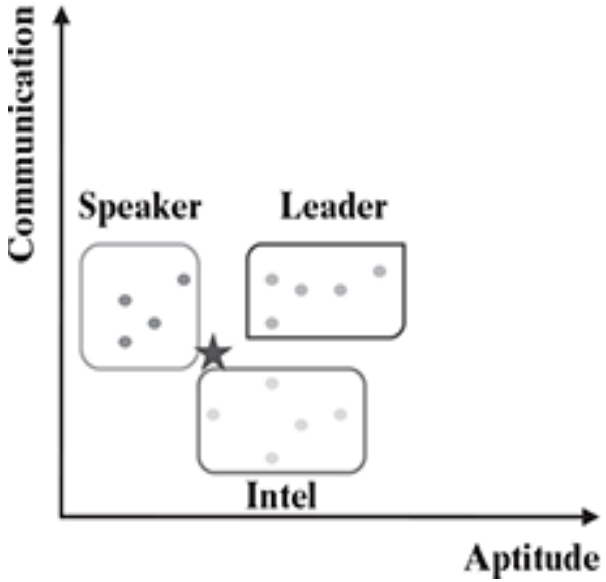
Classification model steps



Common Classification Algorithms

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

kNN



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	???

kNN

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	???				

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?

- stores the training data
- directly applies the philosophy of nearest neighbourhood finding to arrive at the classification

Strengths of the kNN algorithm

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

Weaknesses of the kNN algorithm

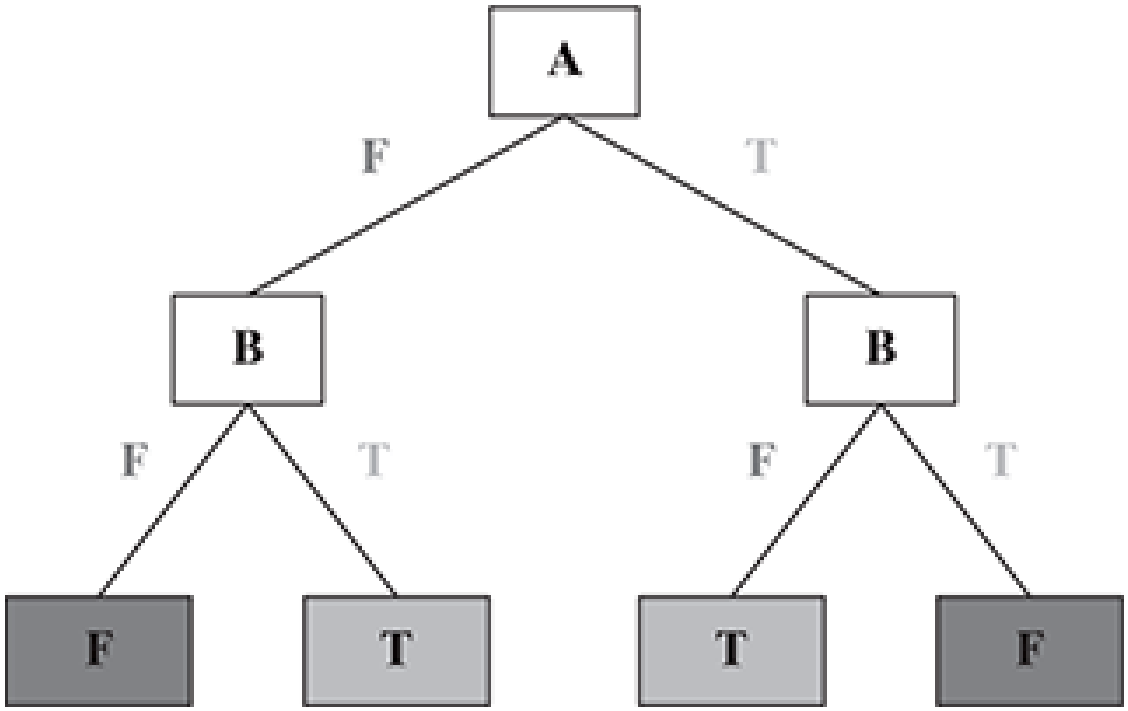
- Does not learn anything in the real sense
- classification is done completely on the basis of the training data, the classification process is very slow
- a large amount of computational space is required to load the training data for classification

Application of the kNN algorithm

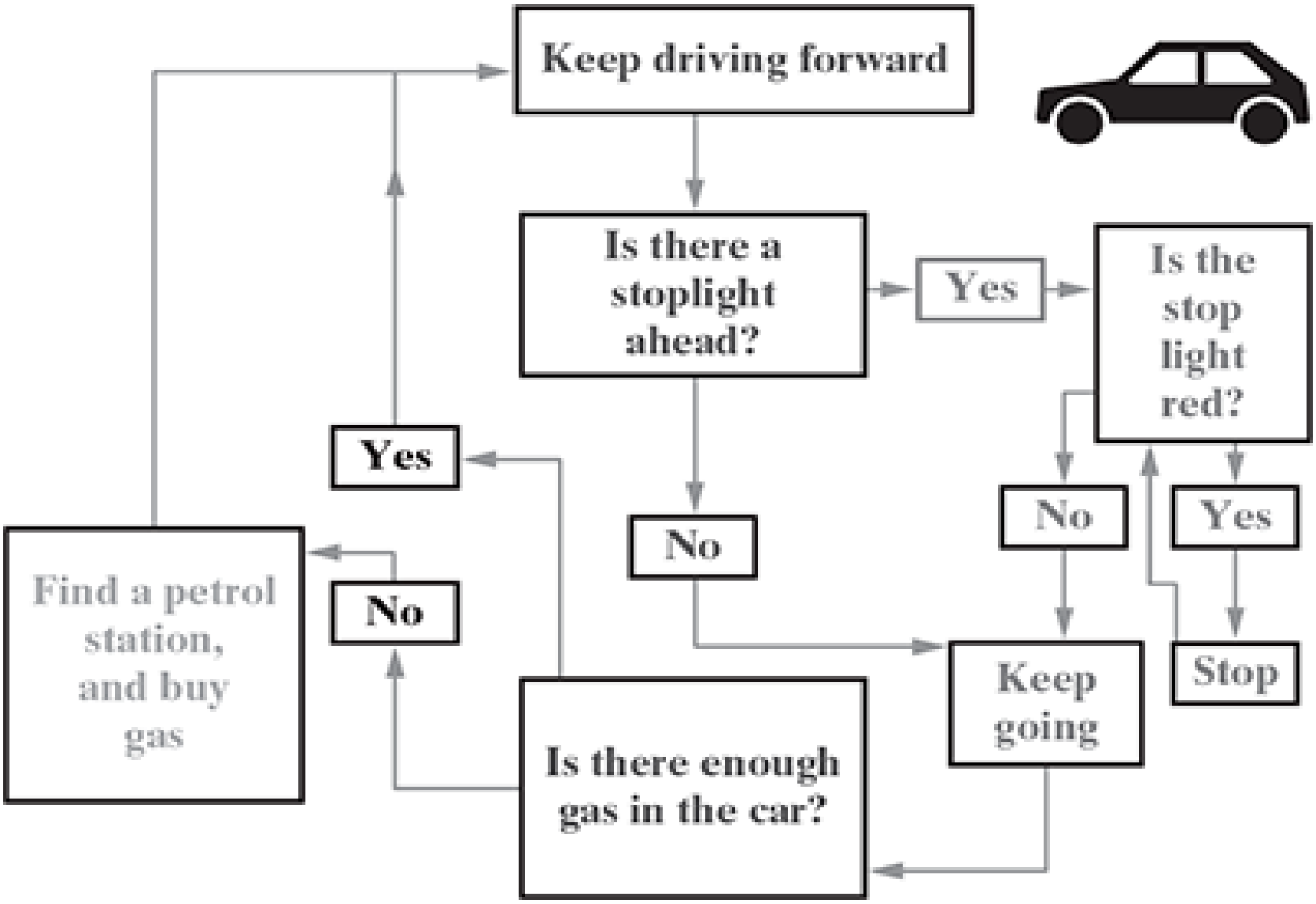
- recommender systems
- searching documents/ contents similar to a given document/content

Decision Tree

- Root Node
- Branch Node
- Leaf Node



Decision Tree



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

Decision Tree

- predict whether Chandra might get a job offer for the given parameter values:
  - CGPA = High
  - Communication = Bad
  - Aptitude = High
  - Programming skills = Bad

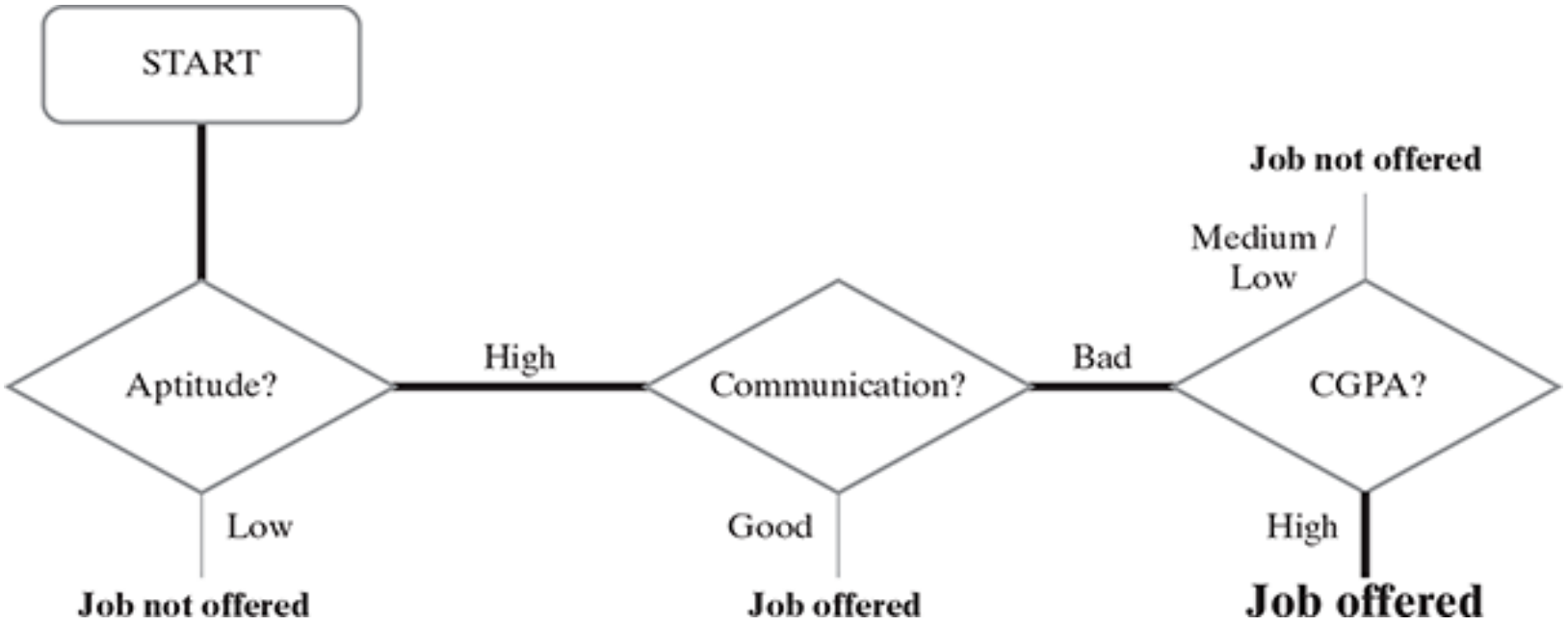
Decision Tree

- **Exhaustive search**
  - Place the item in the first group (class). Recursively examine solutions with the item in the first group (class).
  - Place the item in the second group (class). Recursively examine solutions with the item in the second group (class).
  - Repeat the above steps until the solution is reached.

Decision Tree

- Branch and bound search
  - existing best solution to sidestep searching of the entire decision tree in full

Decision Tree



- predict whether Chandra might get a job offer for the given parameter values:
  - CGPA = High
  - Communication = Bad
  - Aptitude = High
  - Programming skills = Bad

Decision Tree

- **Decsion Tree Implementation**
  - C5.0
  - CART (Classification and Regression Tree)
  - CHAID (Chi-square Automatic Interaction Detector)
  - ID3 (Iterative Dichotomiser 3) algorithms

Decision Tree

- biggest challenge of a decision tree algorithm is to find out which feature to split upon
- Pure Partitions: data should be split in such a way that the partitions created by the split should contain examples belonging to a single class
- Entropy: 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)
- decision tree is all about finding an attribute that returns the highest information gain

Decision Tree

- Entropy of a decision tree

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

- For target class: Job Offered
  - P_i for “Yes” = 0.44 (8/18)
  - P_i for “No” = 0.56 (10/18)

Entropy(S) = -0.44 log 2 (0.44) - 0.56 log 2 (0.56) = 0.99.

Decision Tree

- **Information gain of a decision tree**
  - created on the basis of the decrease in entropy (S) after a data set is split according to a particular attribute (A)
  - 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

Information Gain (S, A) = Entropy (S bs) – Entropy (S as)

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

Decision Tree

- Compare values for
  - when the feature ‘CGPA’ is used for the split
  - when the feature ‘Communication’ is used for the split
  - when the feature ‘Aptitude’ is used for the split
  - when the feature ‘Programming Skills’ is used for the split

(a) Original data set:

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

Total Entropy = 0.99

Decision Tree

(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

LEVEL 1

Decision Tree

Aptitude = High

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

(a) Level 2 starting set:

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

Total Entropy = 0.85

Decision Tree

(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

CGPA = Medium

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

CGPA = Low

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

Total Entropy = 0.33

Information Gain = 0.52

(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

LEVEL 2

Algorithm for decision tree

- **Input:** Training data set, test data set (or data points)
- **Steps:**
  - Do for all attributes
  - Calculate the entropy E_i of the attribute F_i
  - if $E_i < E_{min}$
  - then $E_{min} = E_i$ and $F_{min} = F_i$
  - end if
  - End do
  - Split the data set into subsets using the attribute F_{min}
  - Draw a decision tree node containing the attribute F_{min} and split the data set into subsets. Repeat the above steps until the full tree is drawn covering all the attributes of the original table

Decision Tree

- **Avoiding overfitting in decision tree – 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

Decision Tree

- Strengths of decision tree
  - produces very simple understandable rules
  - Works well for most of the problems
  - handle both numerical and categorical variables
  - Can work well both with small and large training data sets
  - provide a definite clue of which features are more useful for classification

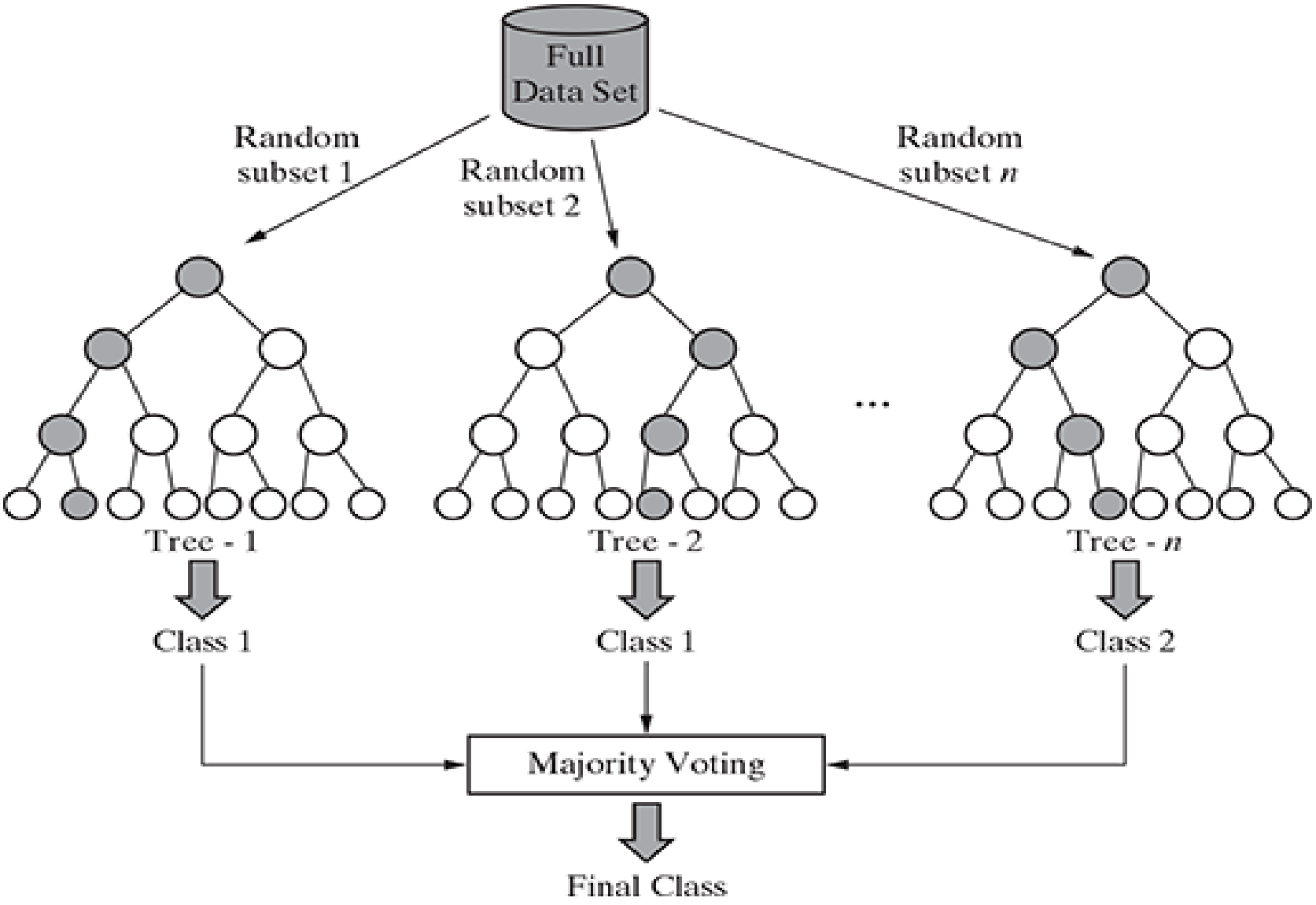
Decision Tree

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

Decision Tree

- Applications
  - 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
  - Can be extended to handle real-value attributes
  - Boolean classification
  - missing attributes and instances with errors in the classification

Random Forest



Random Forest

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

Random Forest

- **Out-of-bag (OOB) error in random forest**
  - each tree is constructed using a different bootstrap sample from the original data
  - 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
  - Predictions for each such sample evaluated each time are tallied, and the final prediction for that sample is obtained by taking a vote
  - The total error rate of predictions for such samples is termed as out-of-bag (OOB) error rate

Random Forest

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

Random Forest

- **Weaknesses of random forest**
  - not as easy to understand as a decision tree model
  - computationally much more expensive than a simple model like decision tree

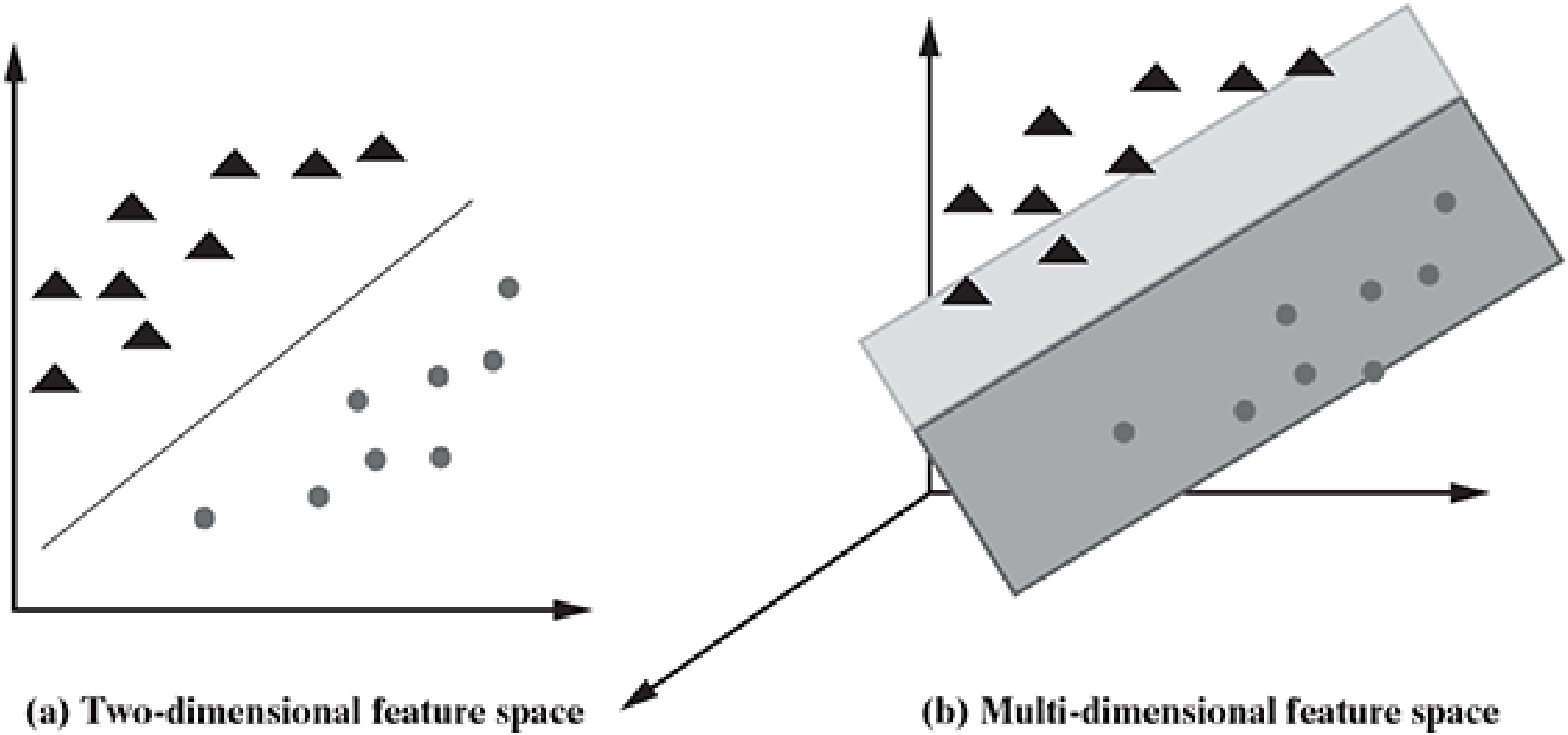
Random Forest

- **Application of random forest**
  - Classification Problems

Support Vector Machines

- linear classification + regression
- based on the concept of a surface
  - Hyperplane: draws a boundary between data instances plotted in the multi-dimensional feature space
- The output prediction of an SVM is one of two conceivable classes which are already defined in the training data
- builds an N-dimensional hyperplane model that assigns future instances into one of the two possible output classes

Support Vector Machines



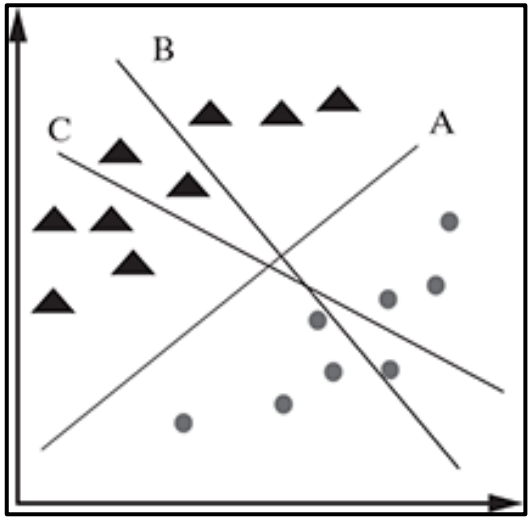
(a) Two-dimensional feature space

(b) Multi-dimensional feature space

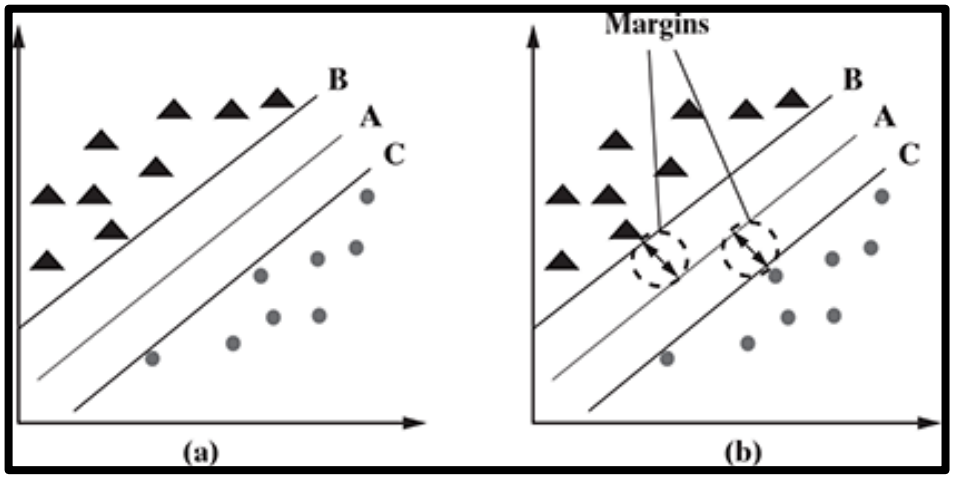
Support Vector Machines

- **Support Vectors:**
  - Support vectors are the data points (representing classes), the critical component in a data set, which are near the identified set of lines (hyperplane). If support vectors are removed, they will alter the position of the dividing hyperplane
- **Hyperplane and Margin:**
  - For an N-dimensional feature space, hyperplane is a flat subspace of dimension (N−1) that separates and classifies a set of data
  - The distance between hyperplane and data points is known as margin

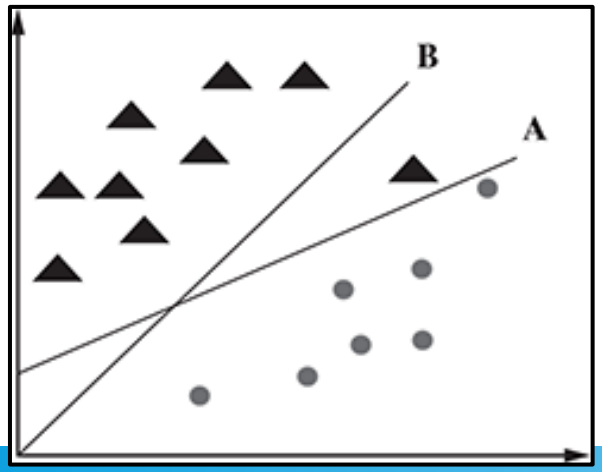
Support Vector Machines



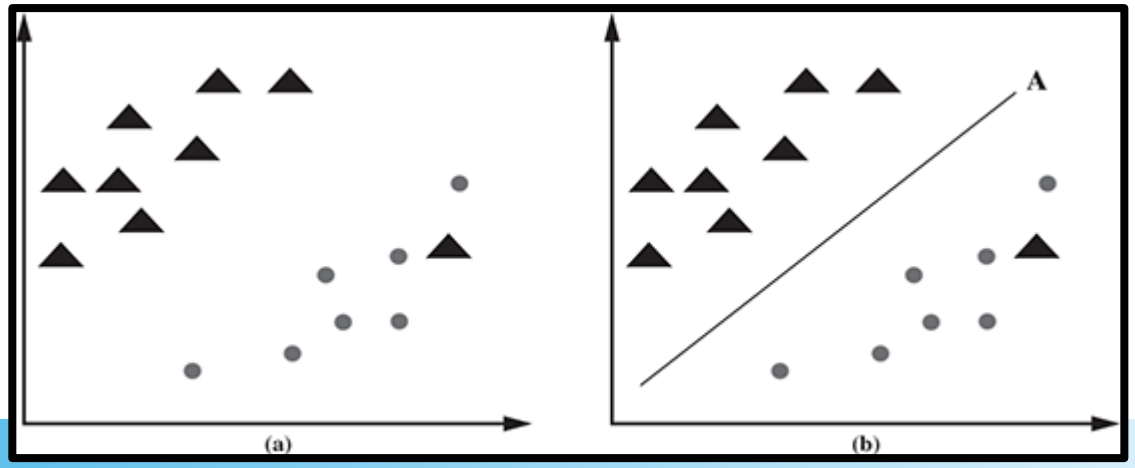
Scenario – 1



Scenario – 2



Scenario – 3



Scenario – 4

Support Vector Machines

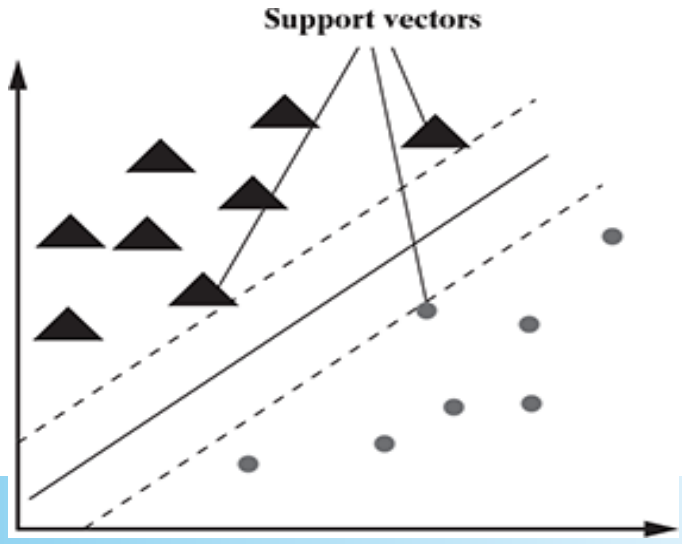
1. The hyperplane should segregate the data instances belonging to the two classes in the best possible way.

2. It should maximize the distances between the nearest data points of both the classes, i.e. maximize the margin.

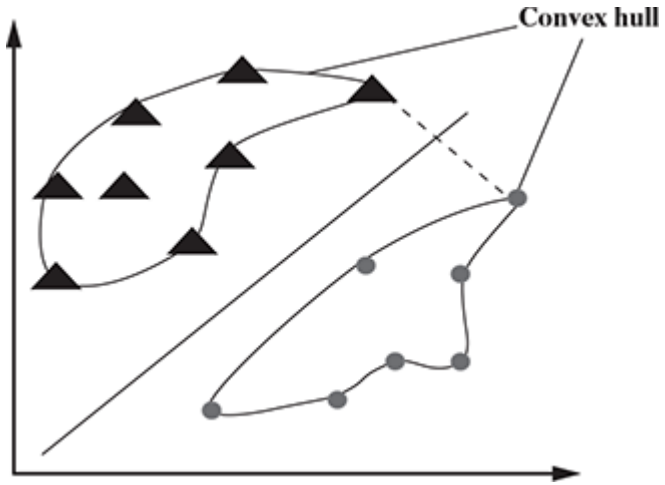
3. If there is a need to prioritize between higher margin and lesser misclassification, the hyperplane should try to reduce misclassifications

Support Vector Machines

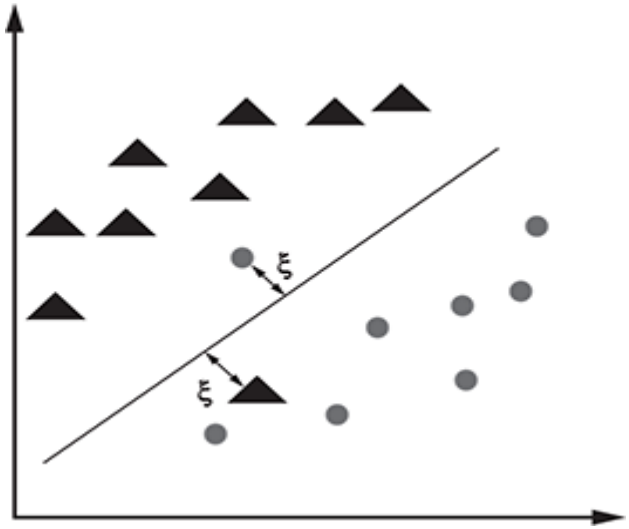
- Maximum Margin Hyperplane (MMH)
  - identifying the hyperplane which has the largest separation with the data instances of the two classes
  - Support vectors are data instances from the two classes which are closest to the MMH



Support Vector Machines



Drawing the MMH for inearly separable data

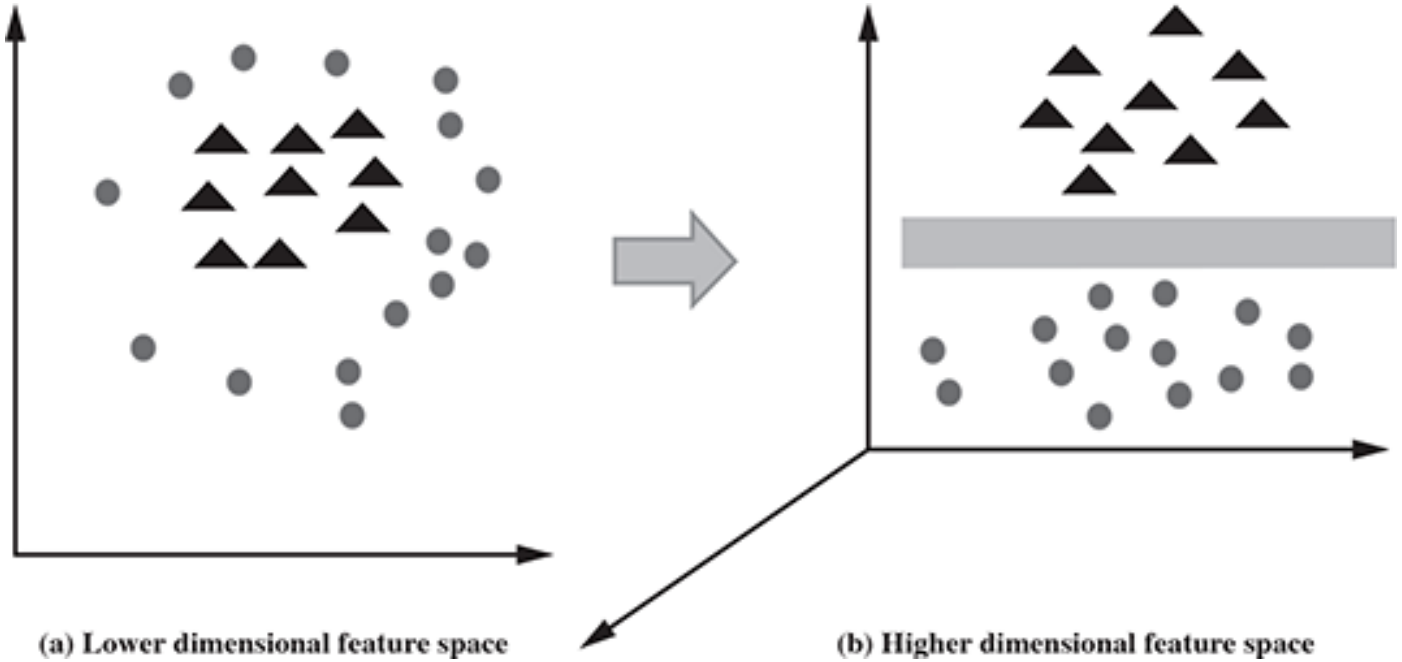


Drawing the MMH for non-linearly separable data

- SVM has a technique called the **kernel trick** to deal with non-linearly separable data.

Support Vector Machines

- SVM has a technique called the kernel trick to deal with non-linearly separable data



(a) Lower dimensional feature space

(b) Higher dimensional feature space

Support Vector Machines

- **Strengths of SVM**
  - used for both classification and regression
  - Robust
  - prediction results using this model are very promising

Support Vector Machines

- **Weaknesses of SVM**
  - only for binary classification
  - very complex
  - slow for a large dataset
  - memory-intensive

Support Vector Machines

- **Applications of SVM**
  - for binary classification
  - in the field of bioinformatics – more specifically, in detecting cancer and other genetic disorders
  - detecting the image of a face by binary classification of images into face and non-face components

Reference

- Machine Learning by Saikat Dutt, Subramanian Chandramouli, Amit Kumar Das published by Pearson