# Unit: 5
# Supervised Learning : Regression and Classification

## Topics

- Regression: Introduction, Example of Regression, Common Regression Algorithms:
- Simple linear Regression, Multiple linear regression
- Classification: Introduction, Classification Model, Classification Learning Steps,
- Classification Algorithms: kNN, Decision Tree, Random Forest, Support Vector Machine

# Regression

## Introduction

- In supervised learning, when we are trying to predict a real-value variable such as ‘Price’, ‘Weight’, etc., the problem falls under the category of regression.
- A regression problem tries to forecast results as a continuous output.
- Dependent Variable (Y) is the value to be predicted. This variable is presumed to be functionally related to the independent variable (X).
- In other words, dependent variable(s) depends on independent variable(s). Independent Variable (X) is called as predictor. The independent variable (X) is used in a regression model to estimate the value of the dependent variable (Y).
- Regression is essentially finding a relationship (or) association between the dependent variable (Y) and the independent variables (X).

## COMMON REGRESSION ALGORITHMS

- Simple linear regression
- Multiple linear regression
- Polynomial regression
- Multivariate adaptive regression splines
- Logistic regression
- Maximum likelihood estimation (least squares)

## Simple linear regression

- If the regression involves only one independent variable, it is called simple regression.
- Thus, if we take 'Price of a used car' as the dependent variable and the 'Year of manufacturing of the car' as the independent variable, we can build a simple regression.
- Slope represents how much the line in a graph changes in the vertical direction (Y-axis) over a change in the horizontal direction (X-axis). Slope is also referred as the rate of change in a graph.
- Maximum and minimum points on a graph are found at points where the slope of the curve is zero. It becomes zero either from positive or from negative value

## Simple linear regression

- a linear relationship between the dependent variable and the predictor variable as shown in Figure

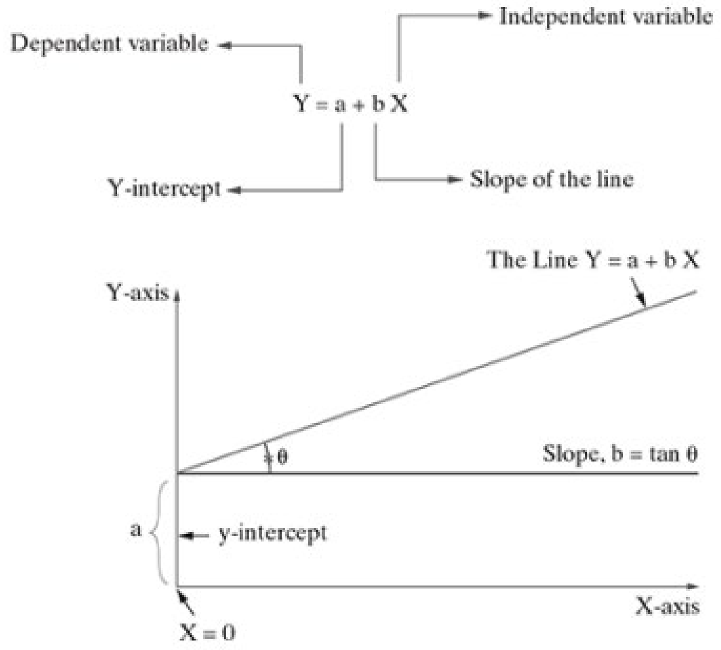


**FIG. 8.1** Simple linear regression

## Multiple linear regression

- If two or more independent variables are involved, it is called multiple regression.
- If we take ‘Price of a used car’ as the dependent variable and year of manufacturing (Year), brand of the car (Brand), and mileage run (Miles run) as the independent variables.
- The following expression describes the equation involving the relationship with two predictor variables, namely X and X.

$$\hat{Y} = a + b_1X_1 + b_2X_2$$

- The model describes a plane in the three-dimensional space of Ŷ, X , and X . Parameter ‘a’ is the intercept of this plane.
- Parameters ‘b ’ and ‘b ’ are referred to as partial regression coefficients
- Consider the example of a multiple linear regression model with two predictor variables, namely X1 and X2:

$$\hat{Y} = 22 + 0.3X_1 + 1.2X_2$$

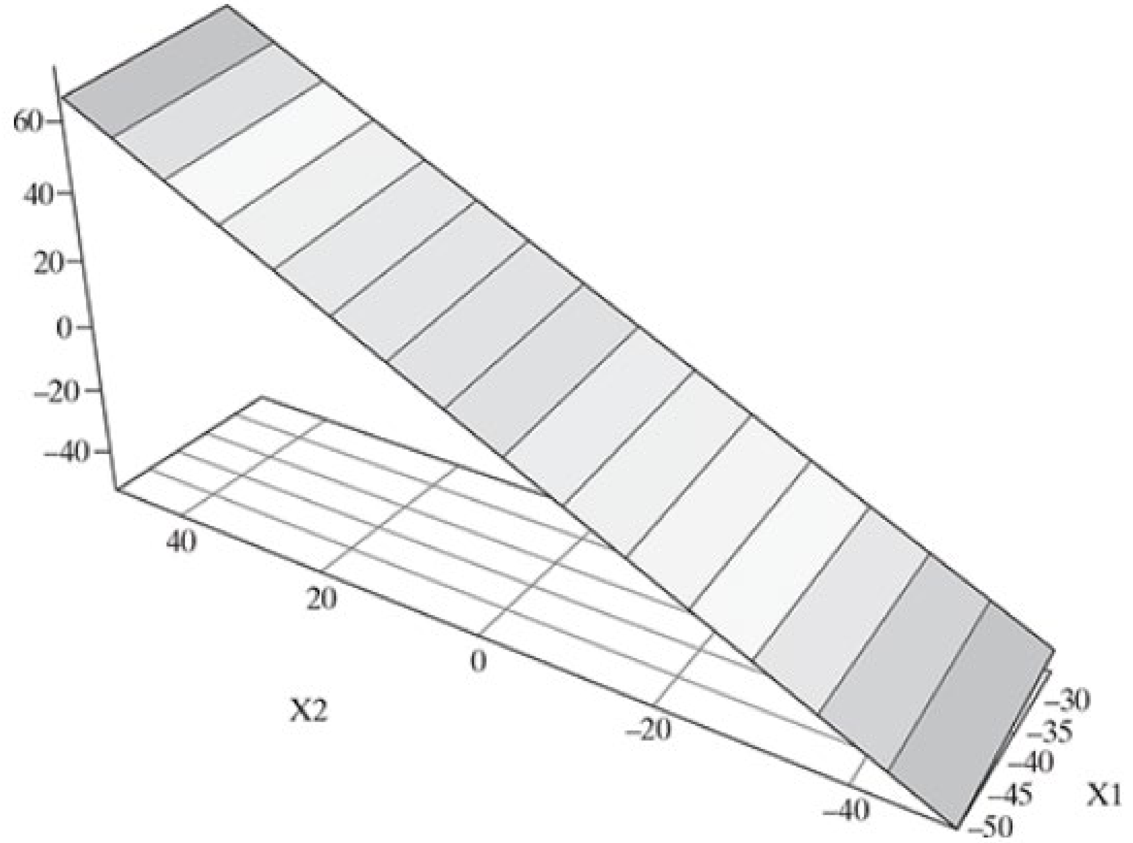


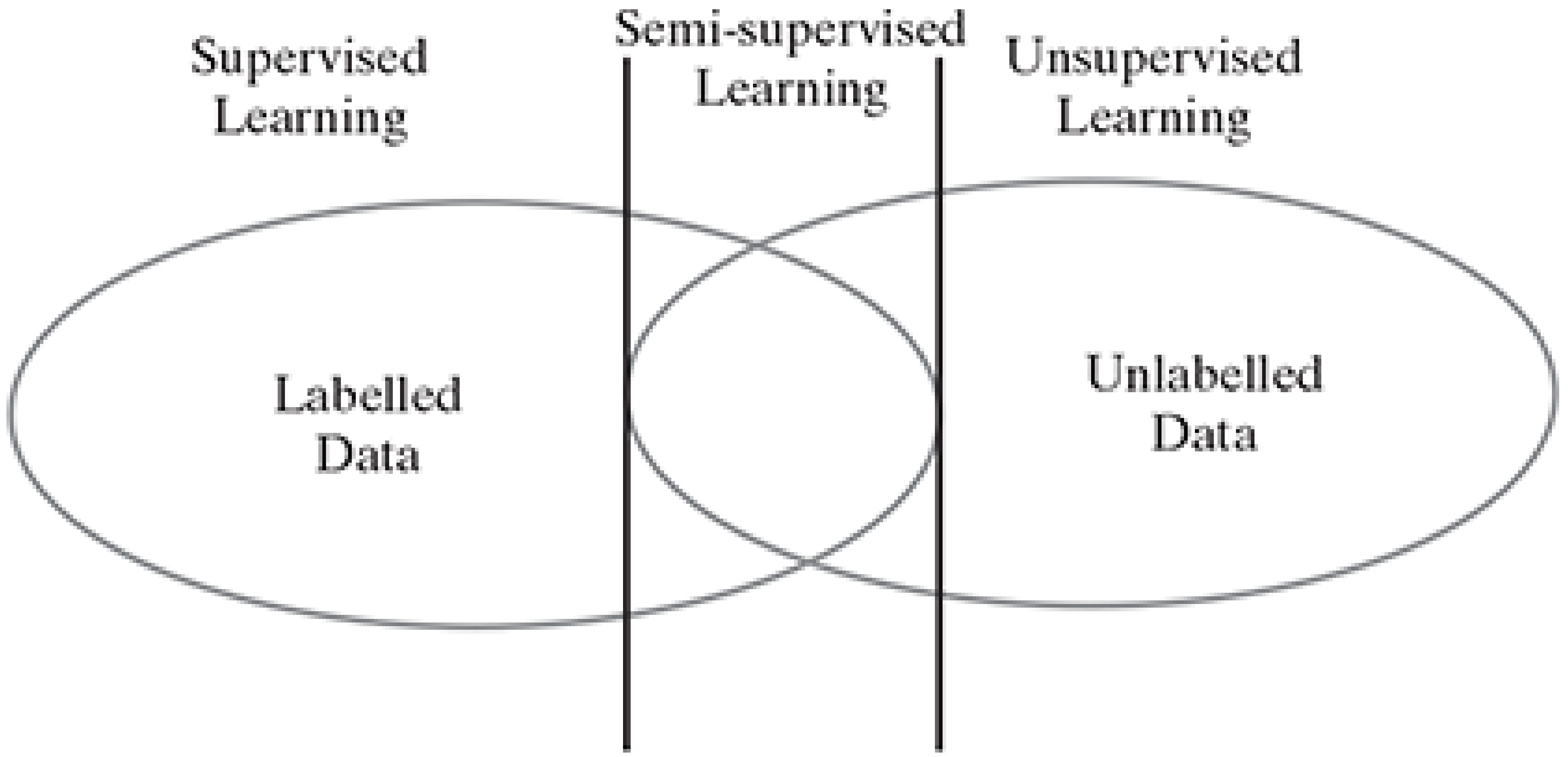
**FIG. 8.15** Multiple regression plane

## Classification

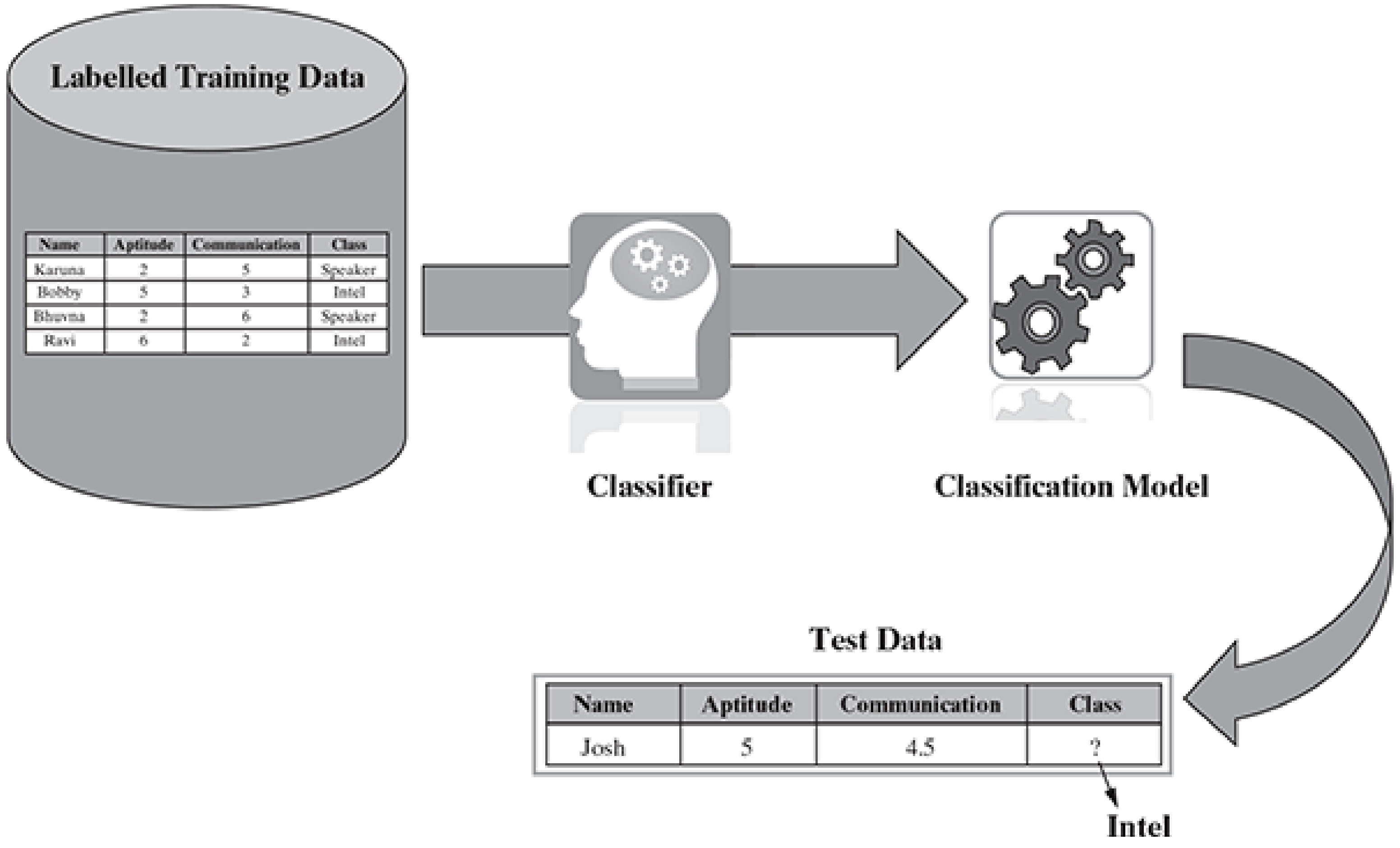
## Intoduction

- In supervised learning, the labelled training data provides the learning basis.
- According to the definition of machine learning, this labelled training data is the experience or prior knowledge or belief.
- Training data is the past information with known value of the class field or ‘label’.

## Supervised learning vs. unsupervised learning



# CLASSIFICATION MODEL



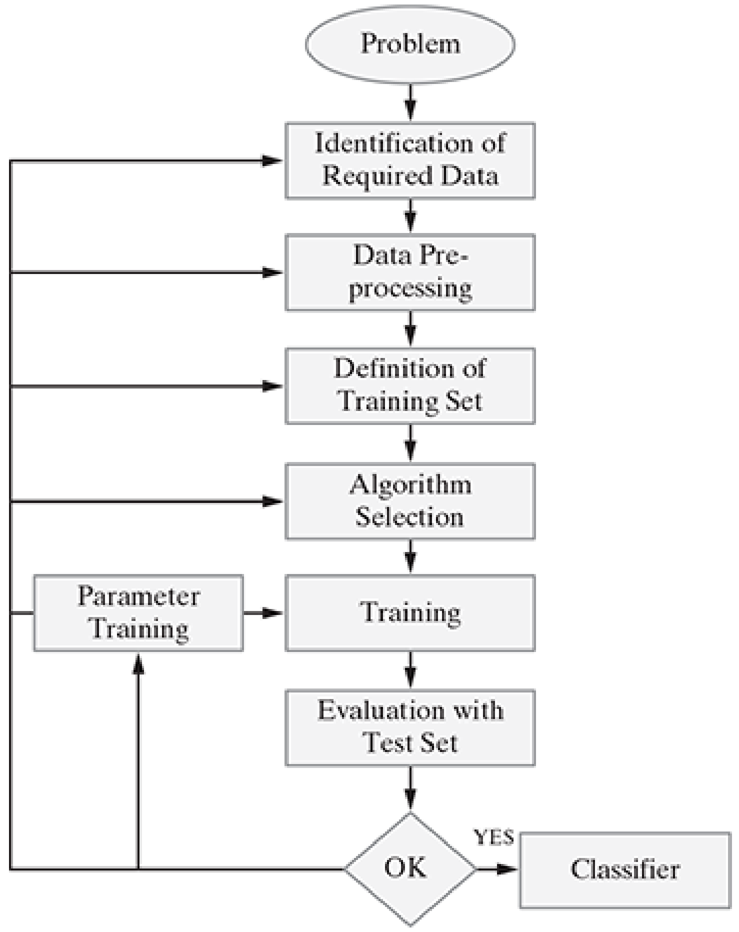
| Name | Aptitude | Communication | Class |
|---|---|---|---|
| Karuna | 2 | 5 | Speaker |
| Bobby | 5 | 3 | Intel |
| Bhuvna | 2 | 6 | Speaker |
| Ravi | 6 | 2 | Intel |

| Name | Aptitude | Communication | Class |
|---|---|---|---|
| Josh | 5 | 4.5 | ? |

## CLASSIFICATION MODEL

- 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
  - Handwriting recognition

## CLASSIFICATION LEARNING STEPS



## Problem Identification

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

## Identification of Required Data

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

## Data Pre-processing

- This is related to the cleaning/transforming the data set.
- This step ensures that all the unnecessary/irrelevant data elements are removed.
- Data pre-processing refers to the transformations applied to the identified data before feeding the same into the algorithm.
- Because the data 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.

## 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).

## Algorithm Selection

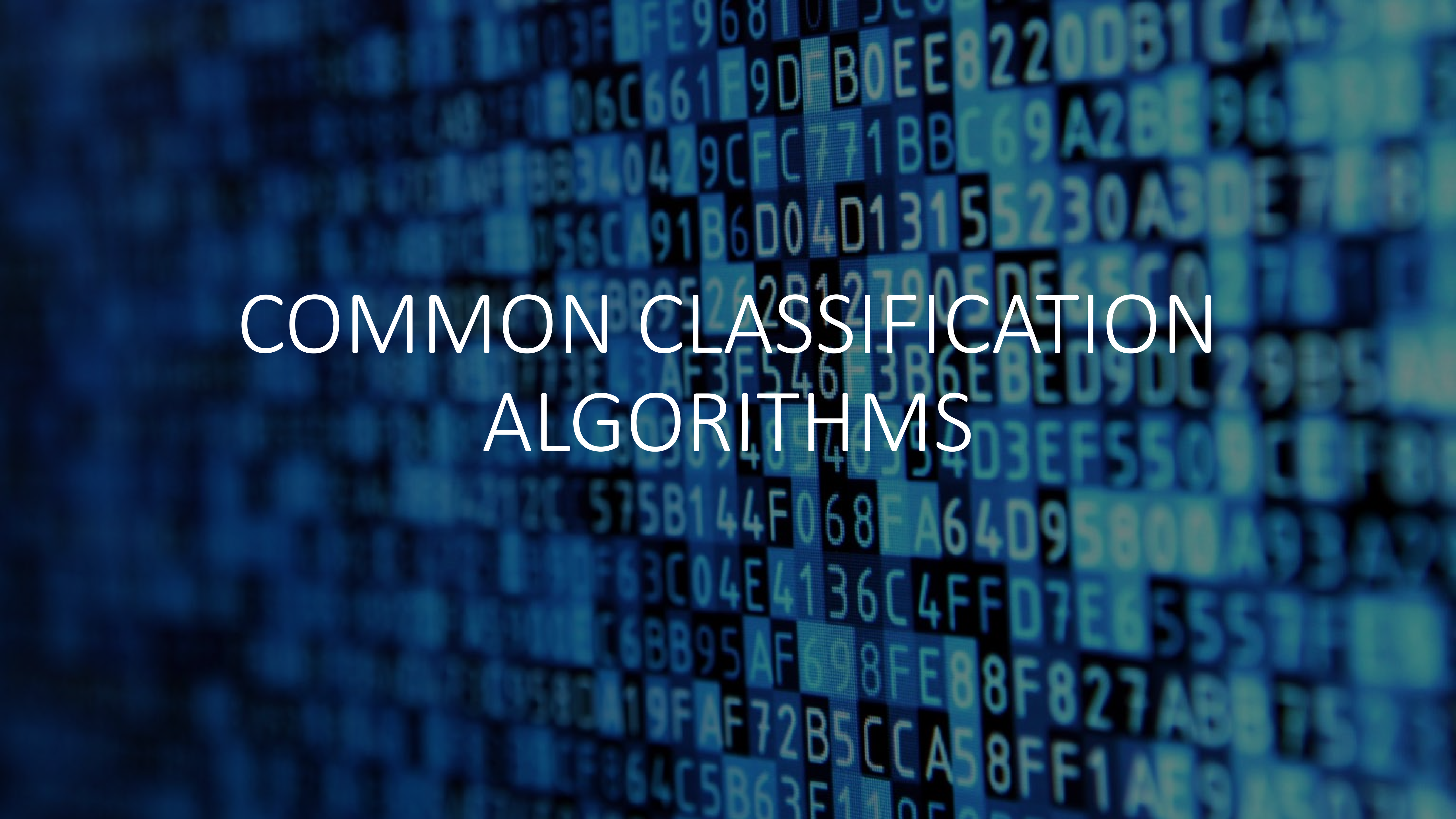
- This involves determining the structure of the learning function and the corresponding learning algorithm. This is the most critical step of supervised learning model.
- On the basis of various parameters, the best algorithm for a given problem is chosen.

## 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

## Evaluation with the Test Data Set

- Training data is run on the algorithm, and its performance is measured here



## COMMON CLASSIFICATION ALGORITHMS

## Classification Algorithms

- k -Nearest Neighbour (kNN)
- Decision tree
- Random forest
- Support Vector Machine (SVM)

# k-Nearest Neighbour (kNN)

- The kNN algorithm is a simple but extremely powerful classification algorithm.
- There are many measures of similarity, the most common approach adopted by kNN to measure similarity between two data elements is Euclidean distance.

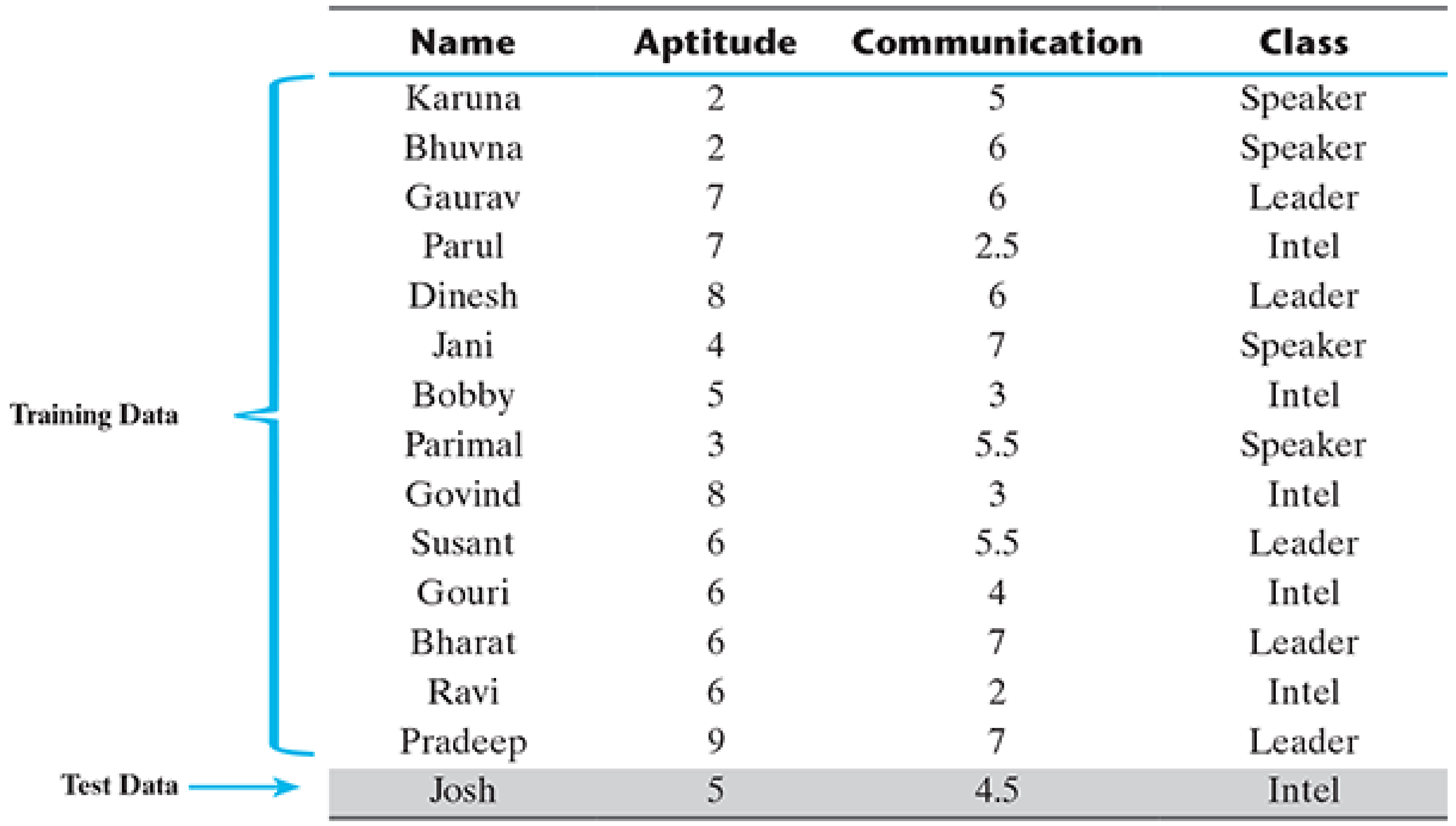
## k-Nearest Neighbour (kNN)

- Considering a very simple data set having two features (say f1 and f2)
- Euclidean distance between two data elements d1 and d2 can be measured by:

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

- Where
- f11 = value of feature f1 for data element d1
- f12 = value of feature f1 for data element d2
- f21 = value of feature f2 for data element d1
- f22 = value of feature f2 for data element d2

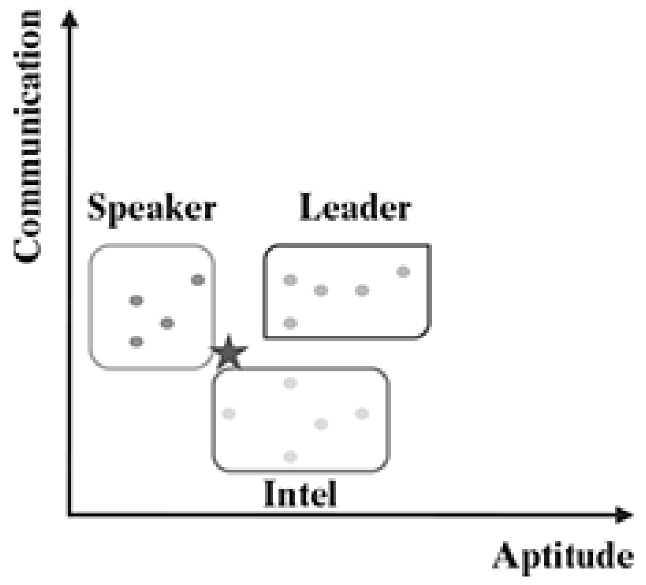
# k-Nearest Neighbour (kNN) - Example

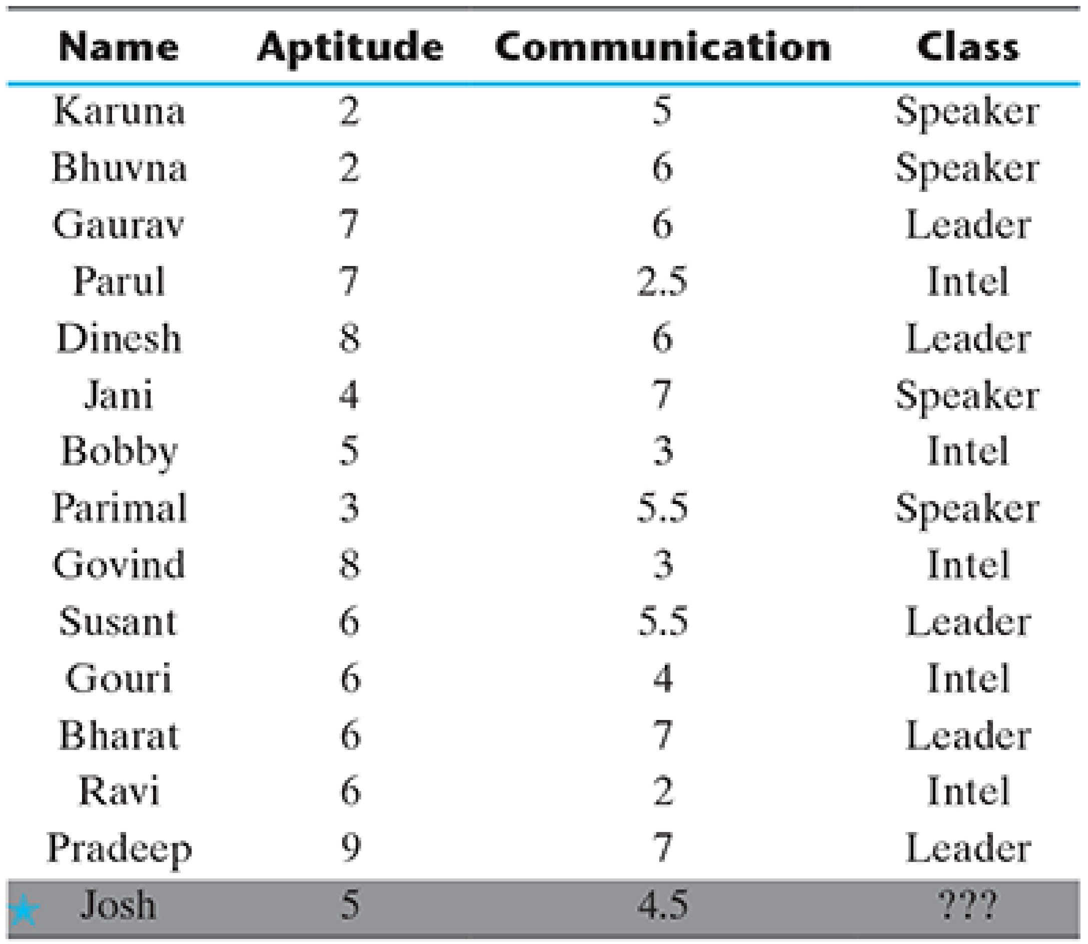

Training Data
Test Data

| 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 |

## k-Nearest Neighbour (kNN) - Example

**2-D representation of the student data set**

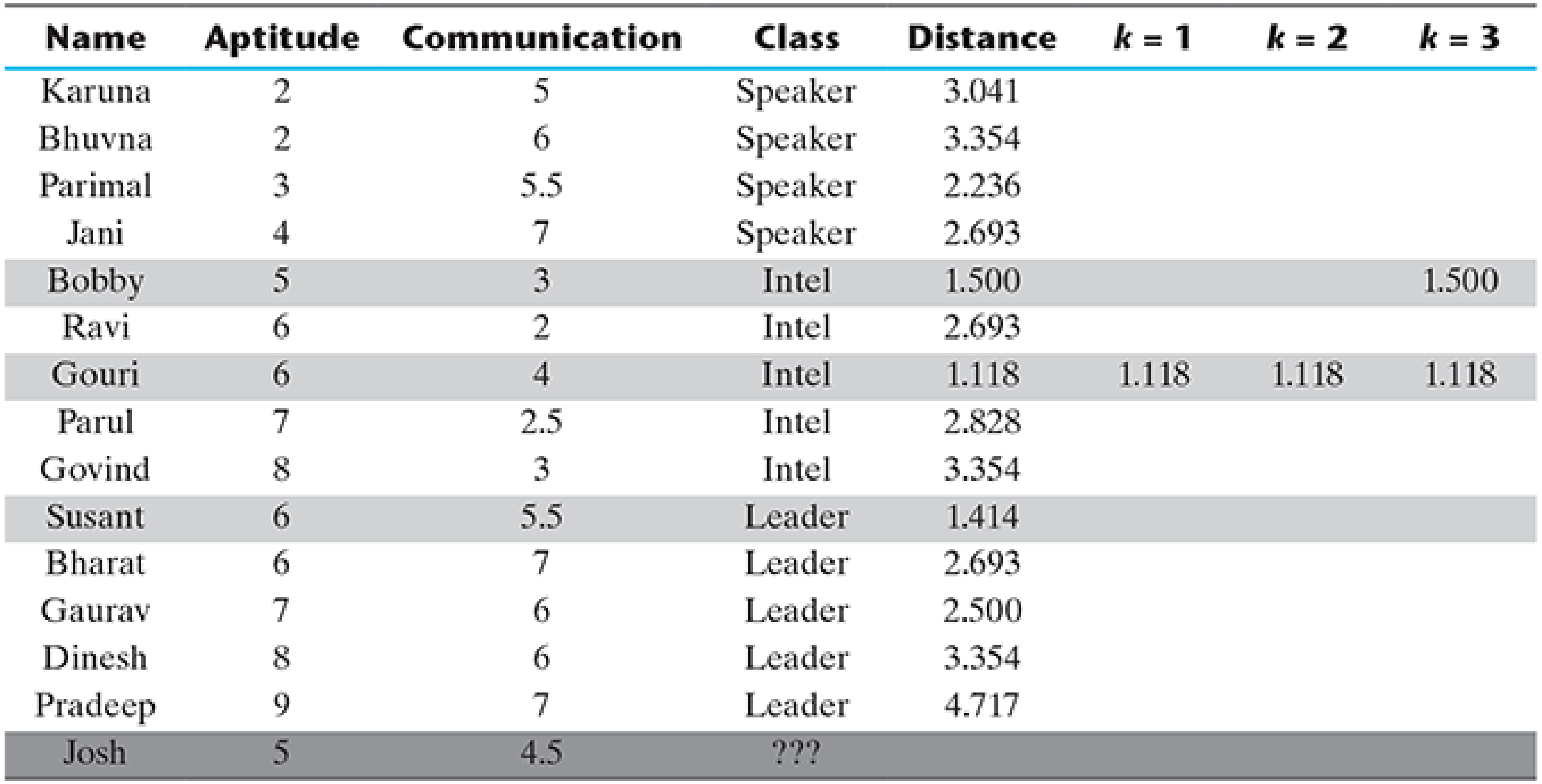




| Name | Aptitude | Communication | Class |
|---|---|---|---|
| Karuna | 2 | 5 | Speaker |
| Bhuvna | 2 | 6 | Speaker |
| Gaurav | 7 | 6 | Leader |
| Parul | 7 | 2.5 | Intel |
| Dinesh | 8 | 6 | Leader |
| Jani | 4 | 7 | Speaker |
| Bobby | 5 | 3 | Intel |
| Parimal | 3 | 5.5 | Speaker |
| Govind | 8 | 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 | ??? |

## k-Nearest Neighbour (kNN) - Example

**Distance calculation between test and training points**



| 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 neighbors to be considered)
Steps:
Do for all test data points
Calculate the distance (usually Euclidean distance) of the test data pointfrom 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
```

## Strengths of the kNN algorithm

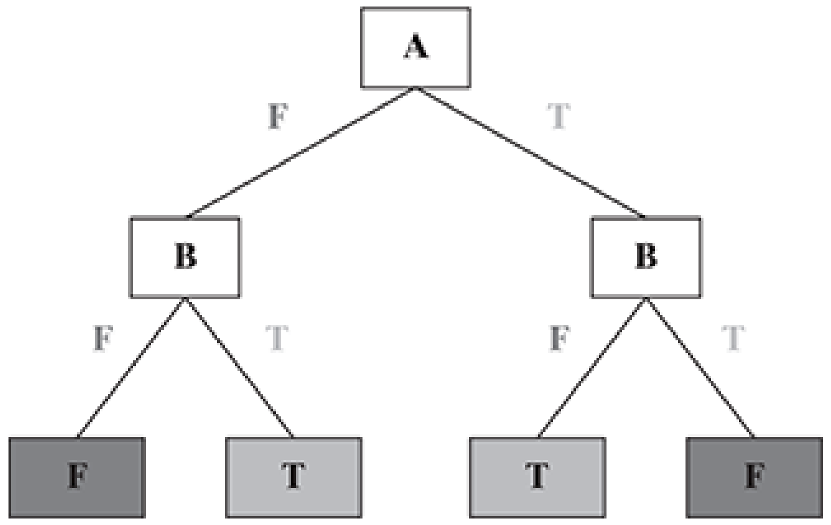
- Extremely simple algorithm – easy to understand
- Very effective in certain situations, e.g. for recommender system design
- Very fast or almost no time required for the training phase

## Weaknesses of the kNN algorithm

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

## Decision tree

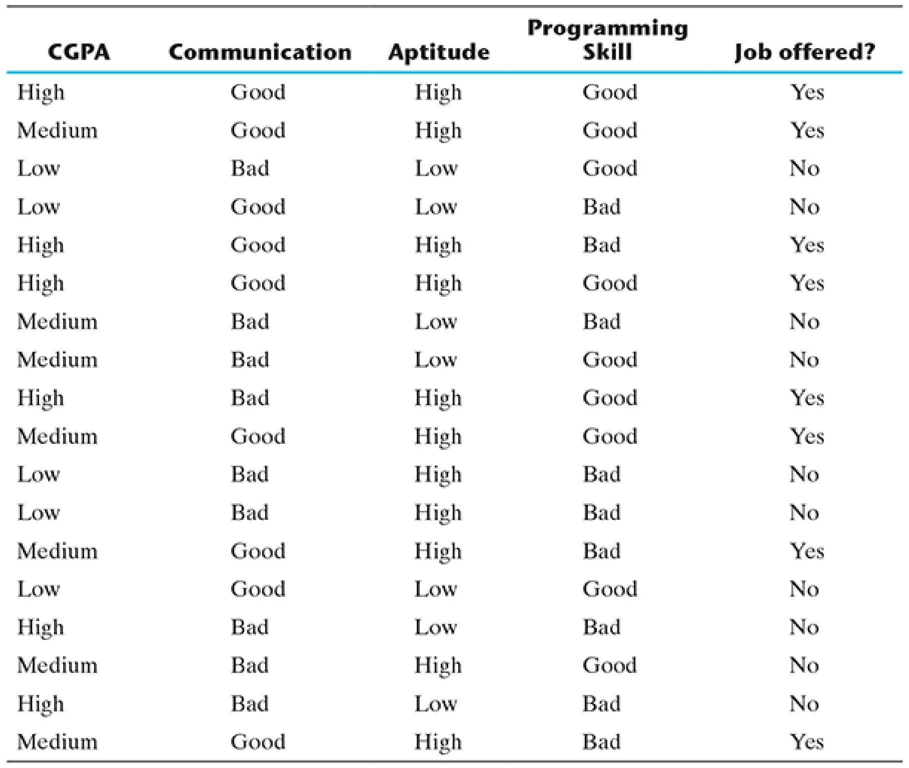
- Decision tree learning is one of the most widely adopted algorithms for classification.
- A decision tree is used for multi-dimensional analysis with multiple classes.
- The goal of decision tree learning is to create a model(based on the past data called past vector) that predicts the value of the output variable based on the input variables in the feature vector.



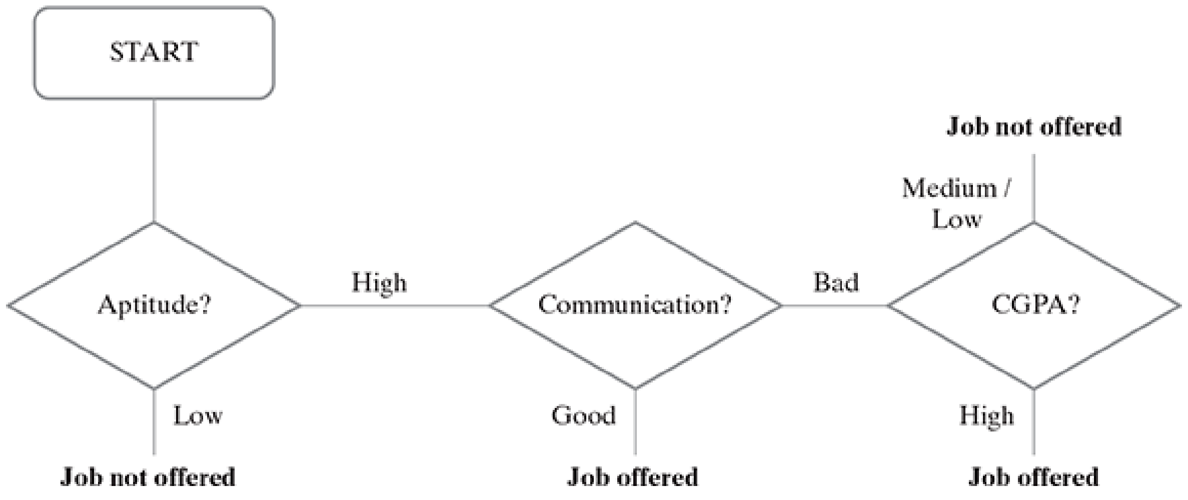
## Decision tree

- Each internal node 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.
- A decision tree consists of three types of nodes:
    - Root Node
    - Branch Node
    - Leaf Node

# Decision Tree - Example



| 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 |



## Entropy of a decision tree

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

$$Entropy(S) = \sum_{i=1}^{c} -pi \log 2 \text{ pi}$$

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

## Entropy of a decision tree

- For example, with respect to the training data we have two values for the target class ‘Job Offered?’ – Yes and No.
- The value of pi 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 (0.44) - 0.56 log (0.56) = 0.99.

## Information gain of a decision tree

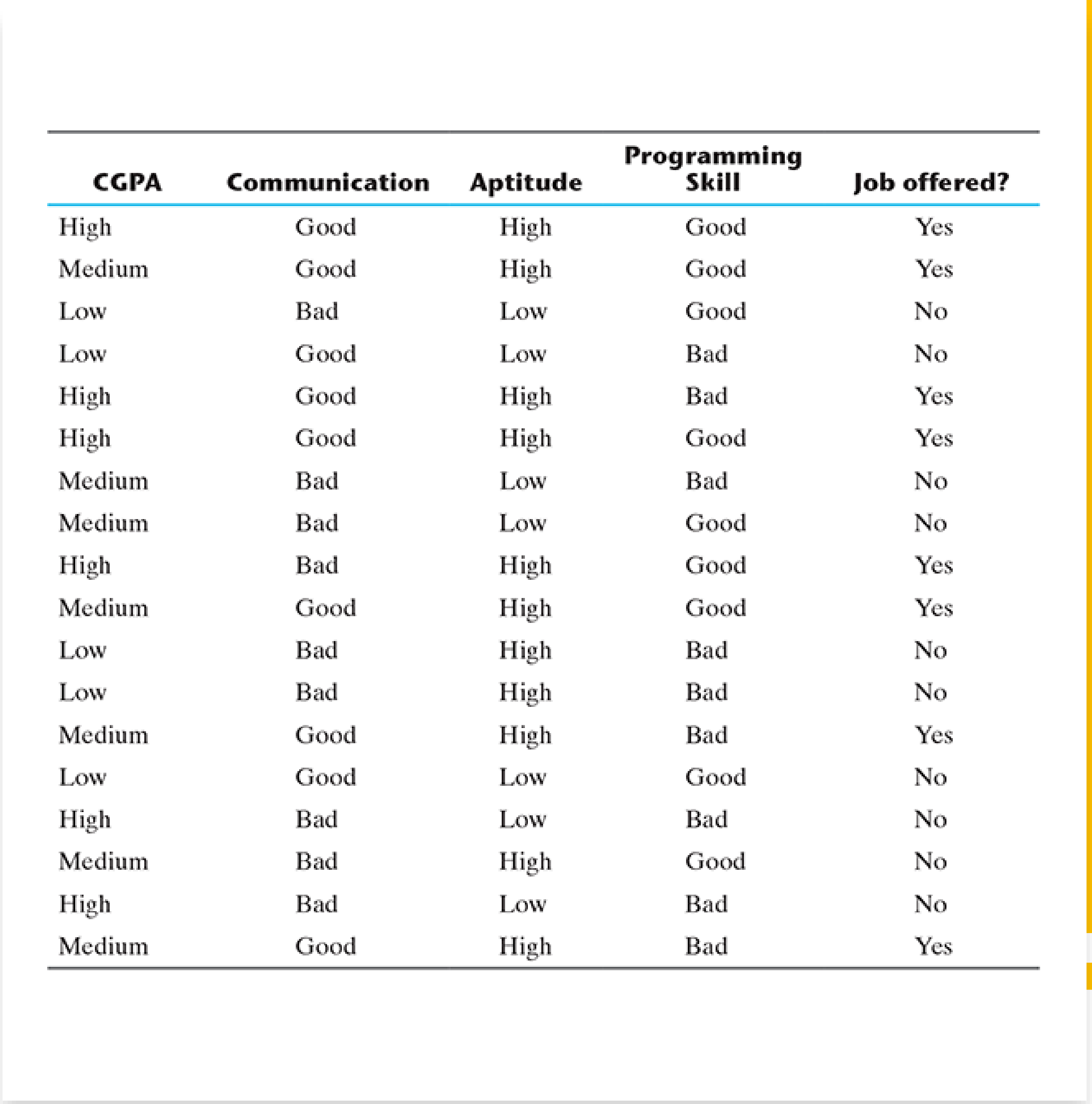
- The information gain is created on the basis of the decrease in entropy (S)after a data set is split according to a particular attribute (A).
- Constructing a decision tree is all about finding an attribute that returns the highest information gain.
- If the information gain is 0, it means that there is no reduction in entropy due to split of the data set according to that particular feature.
- The maximum amount of information gain which may happen is the entropy of the data set before the split.

## Information gain of a decision tree

- Information gain for a particular feature A is calculated by the difference in entropy before a split (Sbs) with the entropy after the split (Sas).
- Information Gain (S,A) = Entropy (Sbs) − Entropy (Sas)
- For performing weighted summation, the proportion of examples falling into each partition is used as weight.

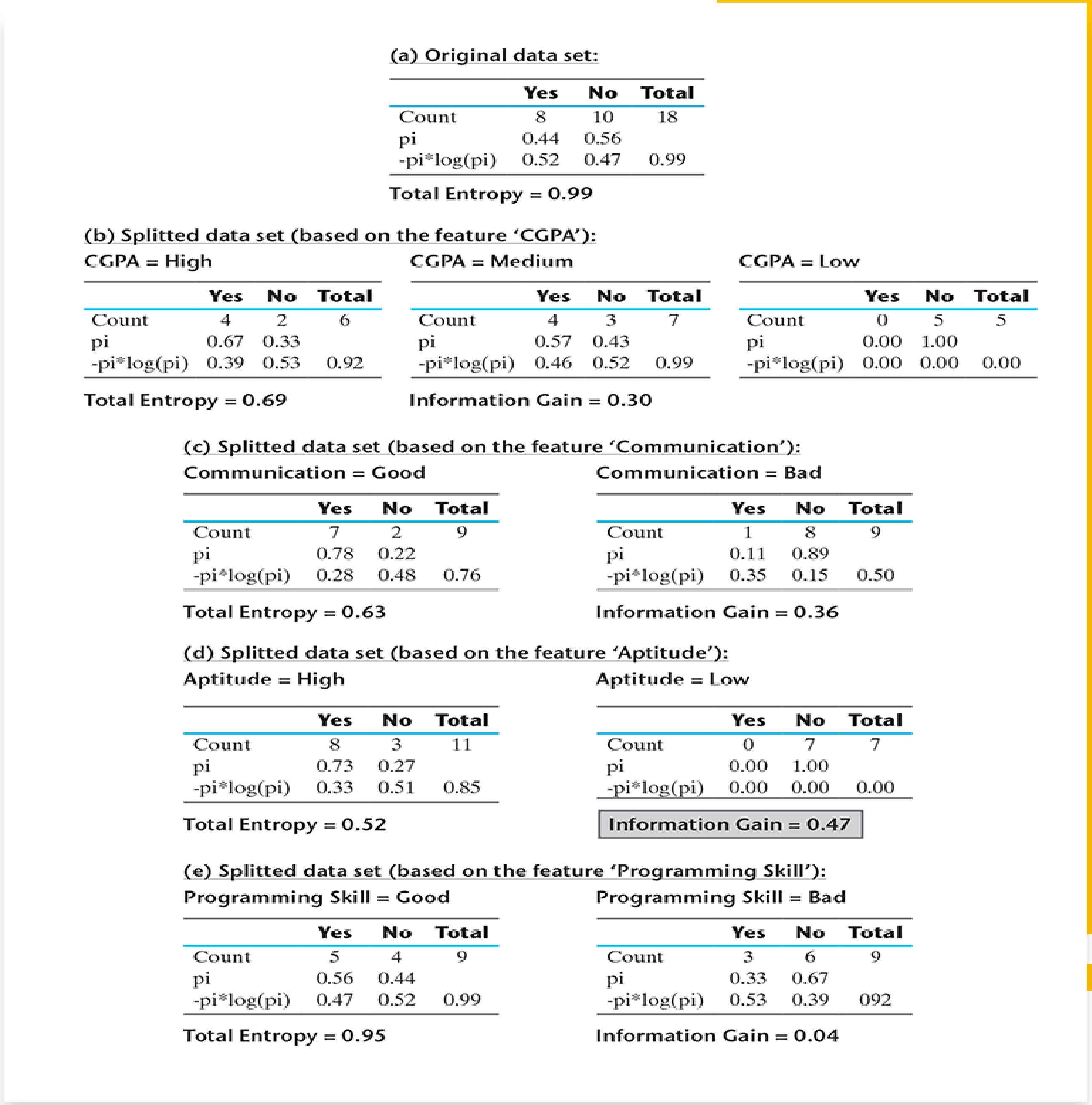
$$\textbf{Entropy}(S_{\text{as}}) = \sum_{i=1}^{n} w_i \,\text{Entropy}\,(p_i)$$

## Example



| 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 |

# 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 |

CGPA = Medium

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

CGPA = Low

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

Total Entropy = 0.69

Information Gain = 0.30

(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 |

$\boxed{\text{Information Gain = 0.47}}$

(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

## Entropy and information gain calculation (Level 1)

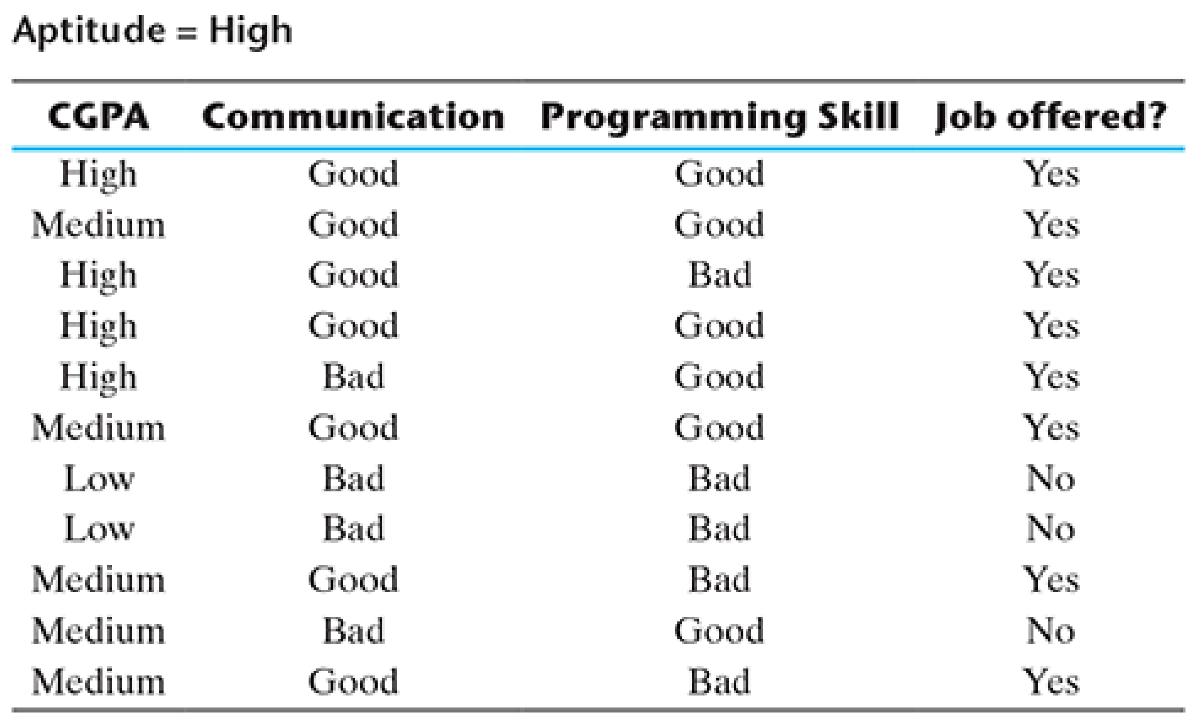
- For Attribute = ‘CGPA’,
  - Total Entropy (Sas) = (6/18)*0.92 + (7/18)*0.99 + (5/18)*0.0 = 0.69
  - Information gain = 0.99 – 0.69 = 0.30

- Similarly calculate for all attributes

## Example

- 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.
- 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

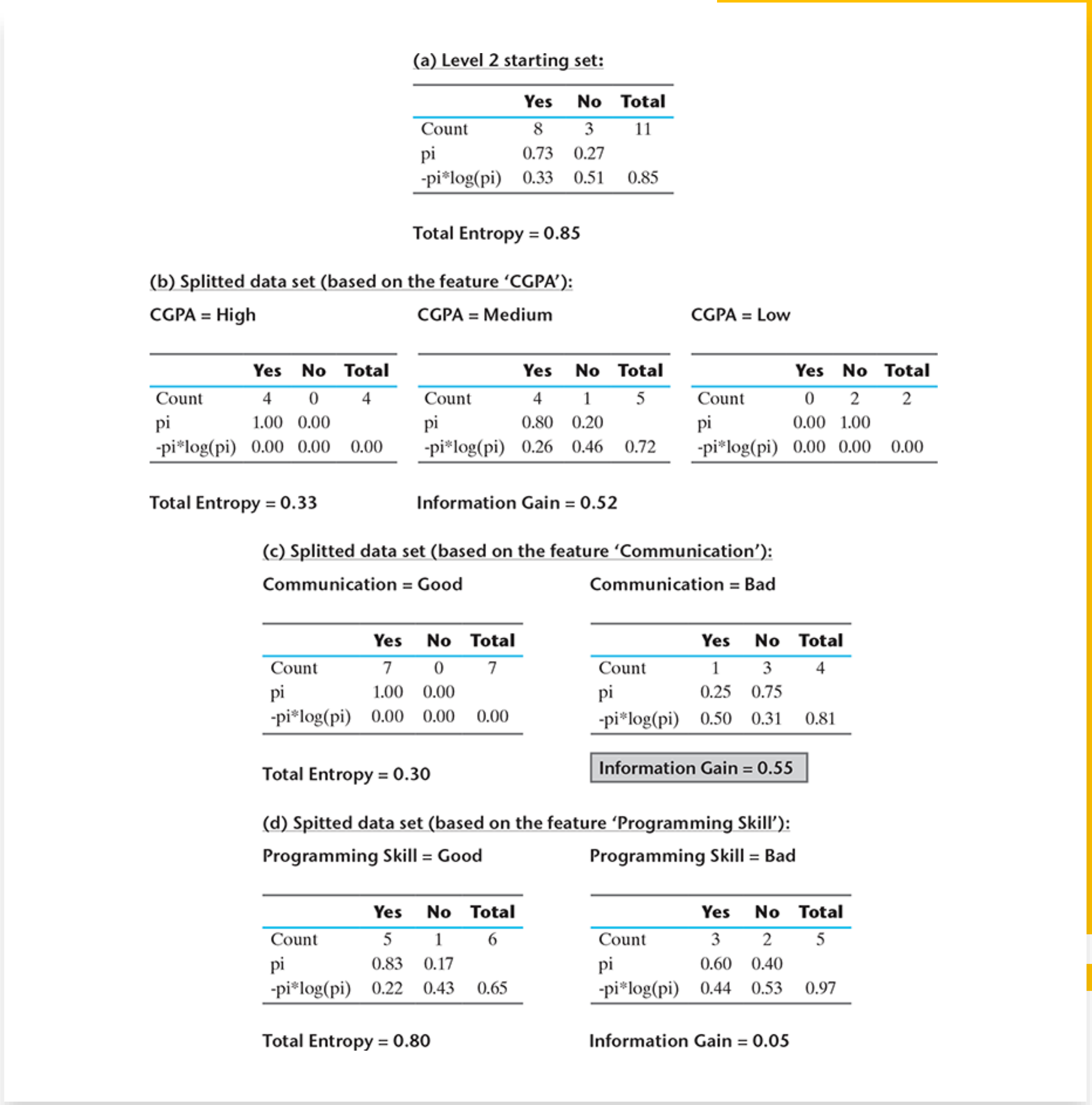
# Data for Aptitude = High



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 |

# 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 |

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 |

Communication = Bad

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

Total Entropy = 0.30

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 |

Programming Skill = Bad

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

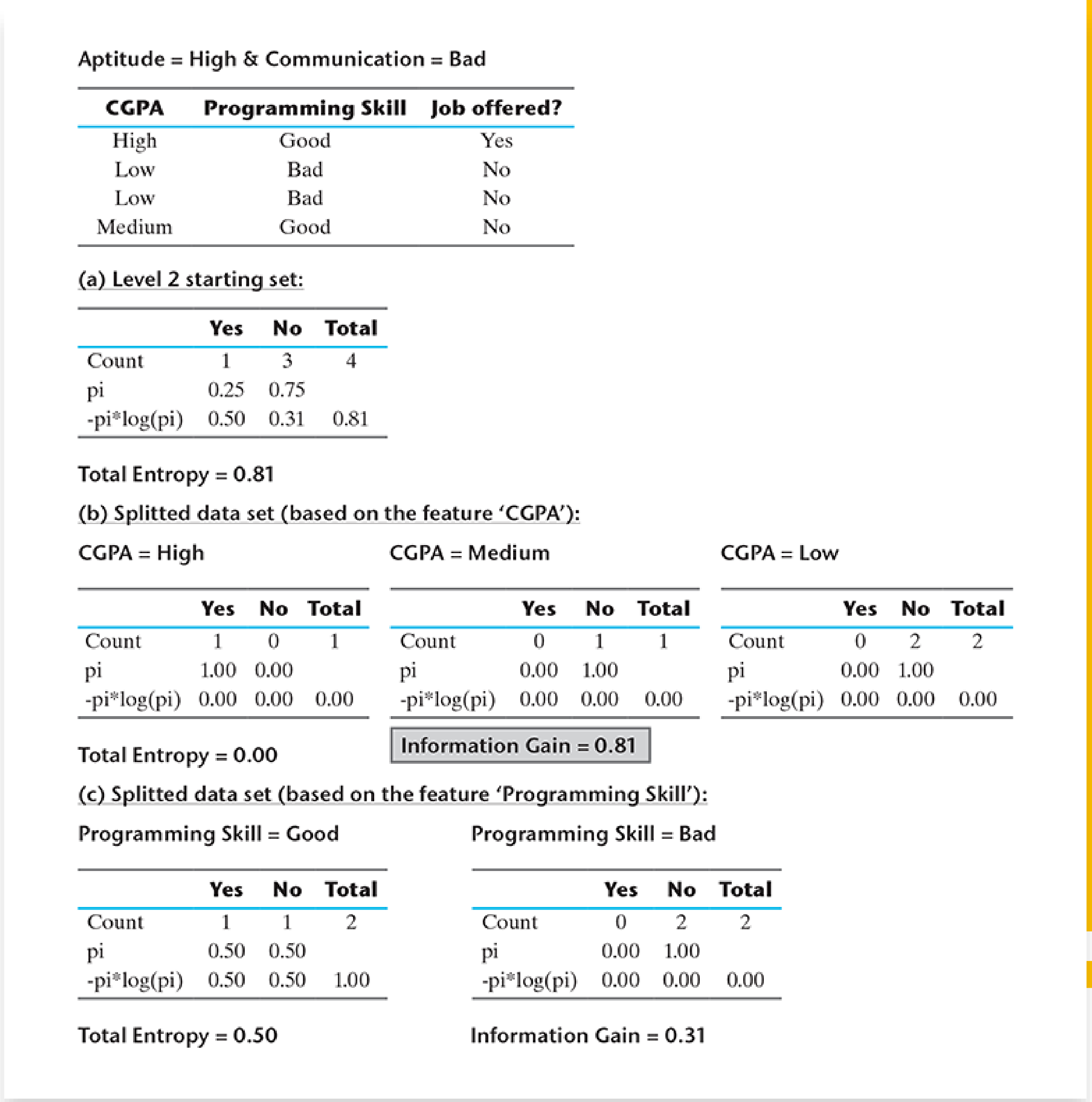
Total Entropy = 0.80

Information Gain = 0.05

## Level 3 Calculations

- As a part of level 3, we will thus have only one branch to navigate in this case – the one for Communication = Bad

# Entropy and information gain calculation (Level 3)



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 |

CGPA = Medium

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

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 = 0.00

Information Gain = 0.81

(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 |

Programming Skill = Bad

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

Total Entropy = 0.50

Information Gain = 0.31

- The information gain after split with the feature CGPA is 0.81, which is the maximum possible information gain (as the entropy before the 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.

# Algorithm for decision tree

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

**Steps:**

```
 Do for all attributes
     Calculate the entropy Ei of the attribute F
      if Ei < Emin
         then Emin = Ei and Fmin = Fi
  end if
  End do
```

**Split the data set into subsets using the attribute Fmin**

**Draw a decision tree node containing the attribute Fmin and split the dataset 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 an 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.

## Avoiding overfitting in decision tree – pruning

- 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.

## Strengths of decision tree

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

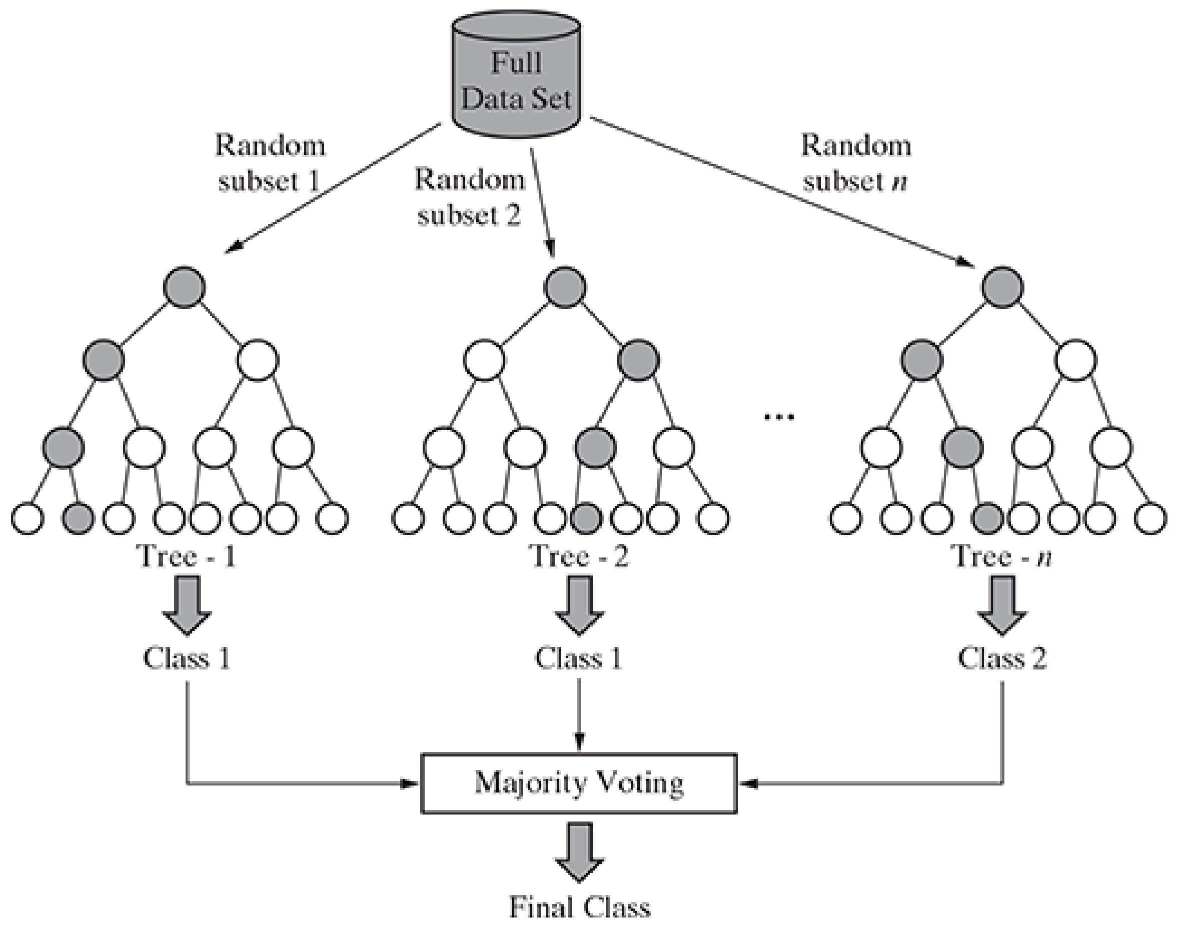
## Weaknesses of decision tree

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

## Random forest model

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

# A simplified random forest model



## How does random forest work?

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. 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.

## 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 generalisation error as the forest generation progresses.
- Generated forests can be saved for future use on other data.
- Lastly, the random forest algorithm can be used to solve both classification and regression problems.

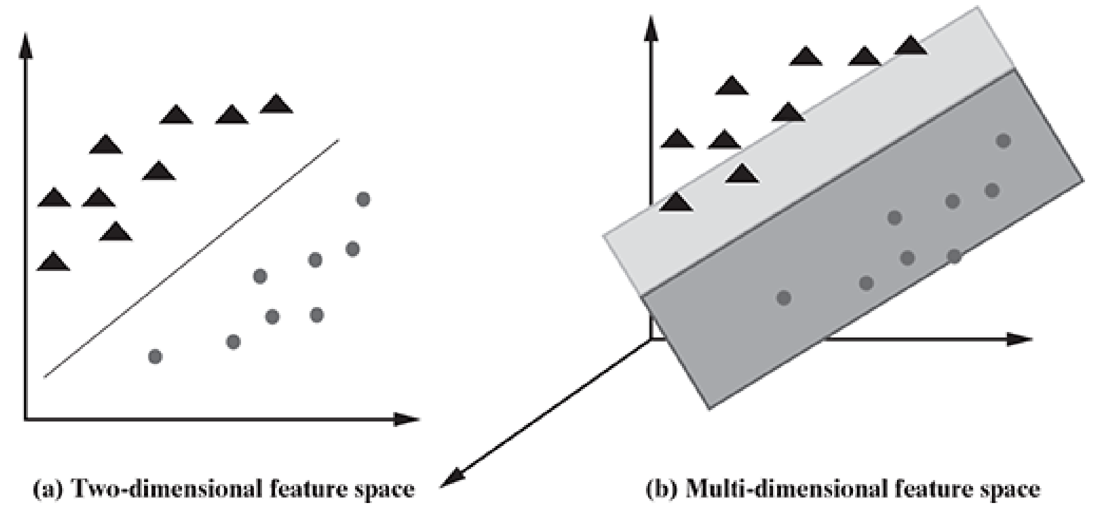
## Weaknesses of random forest

- 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.

## Support vector machines

- 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 conceivable classes which are already defined in the training data.

# Classification using hyperplanes


(a) Two-dimensional feature space
(b) Multi-dimensional feature space

Linearly separable data instances

## Classification using hyperplanes

- 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, hyper-plane is a flat subspace of dimension (N−1) that separates and classifies a set of data.

## Classification using hyperplanes

Mathematically, in a two-dimensional space, hyperplane can be defined by the equation:

c0+c1X1+c2X2 = 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:

c0+c1X1+c2X2+ … +cNXN= 0 which, in short, can be represented as follows:

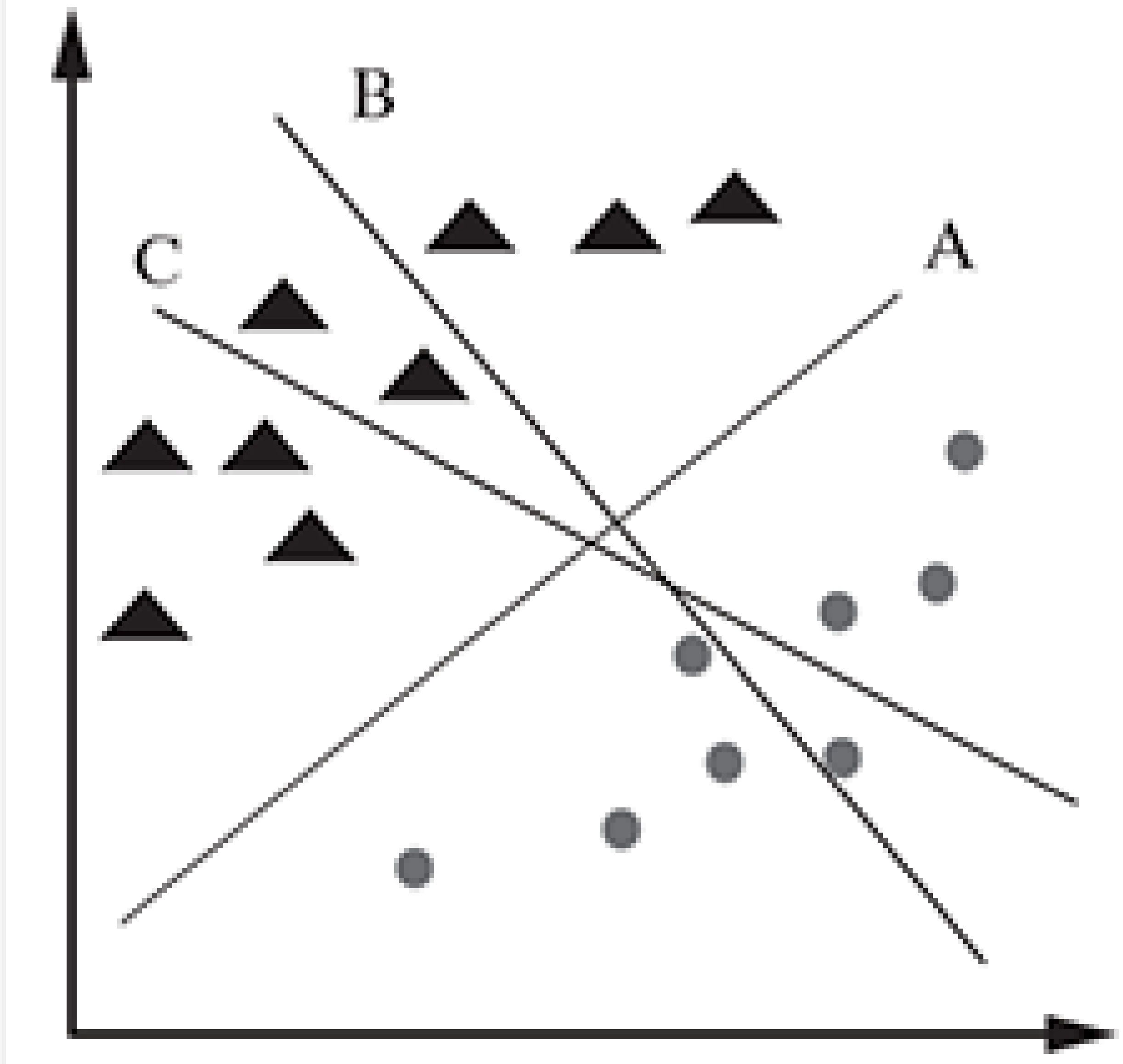
$$\vec{c}.\vec{X} + c_0 = 0$$

The distance between hyperplane and data points is known as margin

## Identifying the correct hyperplane in SVM

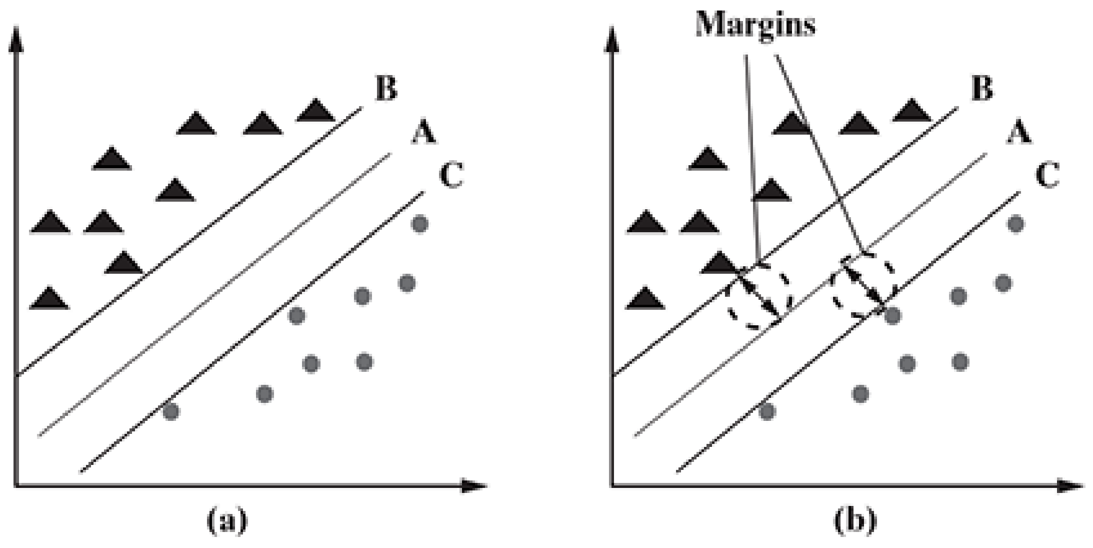
- There may be multiple options for hyper-planes dividing the data instances belonging to the different classes.
- Need to identify which one will result in the best classification.

# SVM



- Three hyperplanes: A,B, and C.
- To identify the correct hyperplane which better segregates the two classes represented by the triangles and circles.
- Hyperplane ‘A’ has performed this task quite well.

- Three hyperplanes: A, B, and C
- 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.



## Kernel trick

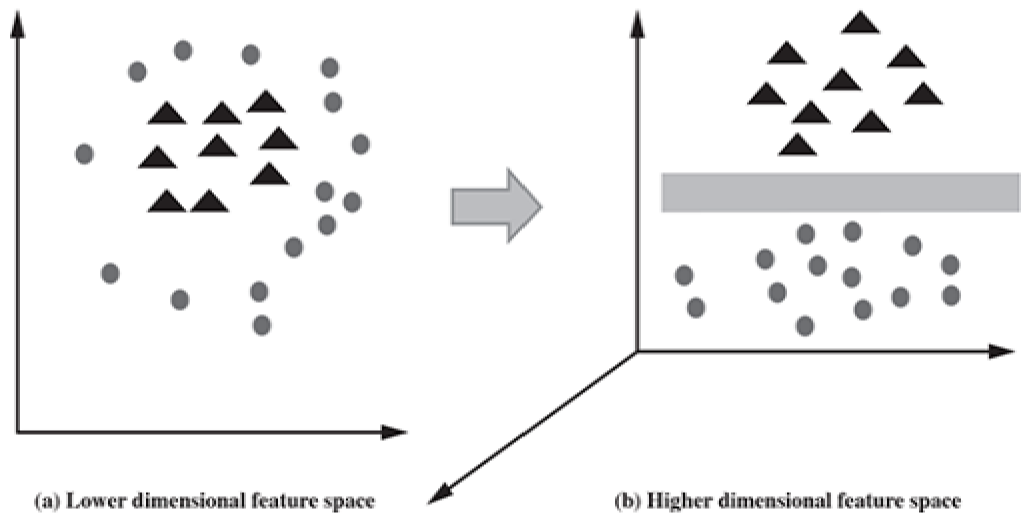
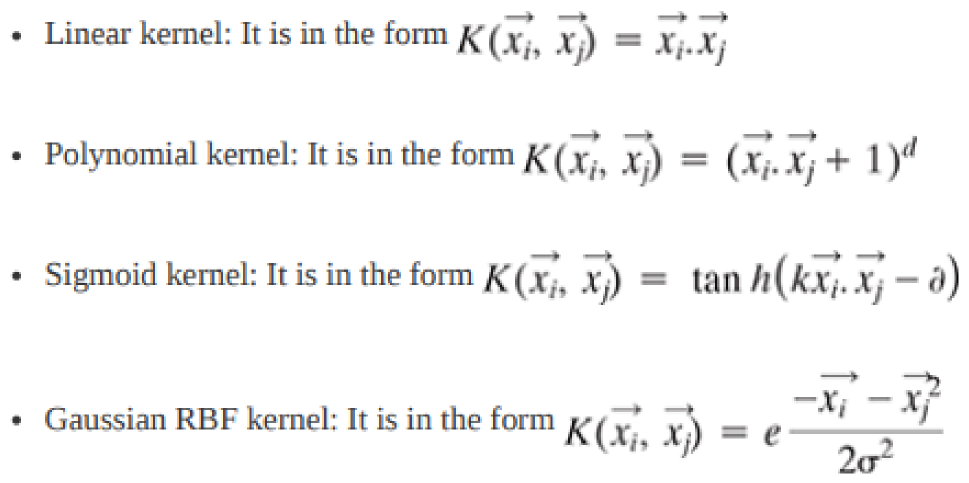

(a) Lower dimensional feature space
(b) Higher dimensional feature space

FIG. 7.23 Kernel trick in SVM

- SVM has a technique called the kernel trick to deal with nonlinearly separable data.
- these are functions which can transform lower dimensional input space to a higher dimensional space.
- converts linearly non-separable data to a linearly separable data. These functions are called kernels
- .

# Kernel trick

- Some of the common kernel functions for transforming from a lower dimension ‘i’ to a higher dimension ‘j’ used by different SVM implementations are as follows:



- Linear kernel: It is in the form $K(\vec{x_i}, \vec{x_j}) = \vec{x_i}.\vec{x_j}$
- Polynomial kernel: It is in the form $K(\vec{x_i}, \vec{x_j}) = (\vec{x_i}.\vec{x_j} + 1)^d$
- Sigmoid kernel: It is in the form $K(\vec{x_i}, \vec{x_j}) = \tan h\left(k\vec{x_i}.\vec{x_j} - \partial\right)$
- Gaussian RBF kernel: It is in the form $K(\vec{x_i}, \vec{x_j}) = e^{\frac{-\vec{x_i} - \vec{x_j}^2}{2\sigma^2}}$

## Kernel trick

- When data instances of the classes are closer to each other, this method can be used. The effectiveness of SVM depends both on the
  - Selection of the kernel function
  - Adoption of values for the kernel parameters

## 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.

## Weaknesses of SVM

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

# Thank YOU