Scheme of Evaluation
Internal Assessment Test 3 – July 2023

Sub:	DATA MINING AND DATA WAREHOUSING							**Code:**	18CS641
Date**:**	5/7/2023	Duration:	90mins	Max Marks:	50	**Sem:**	VI	**Branch:**	ISE

Note: Answer Any five full questions.

Question #		**Description**	**Marks Distribution**		**Max Marks**
1	a)	What is Baye's Theorem? Explain how it is used for classification with example? Bayes Theorem Example Explanation	2M 4M 4M	10M	10M
2	a)	Explain with example, how to build decision tree using Hunt's algorithm Algorithm Pseudo code Explanation Example	4M 3M 3M	10M	10M
3	a)	Explain the measures for selecting the best split with example Entropy Gini Index Classification error	2M 2M 2M	6M	10M
3	b)	Explain characteristics of decision tree induction Any 4 characteristics	1M*4	4M	

<table>
<tr><td>4</td><td>a)</td><td>With example, explain Agglomerative Hierarchical Clustering with example.
Agglomerative clustering algorithm
Explanation
Example + Diagram + clusters</td><td>3M
2M
5M</td><td>10M</td><td>10M</td></tr>
<tr><td>5</td><td>a)</td><td>With time and space complexity, explain DBSCAN clustering algorithm.
DBSCAN algorithm pseudo code
Explanation
Example +diagram + clusters</td><td>3M
2M
5M</td><td>10M</td><td>10M</td></tr>
<tr><td>6</td><td>a)</td><td>Apply K-Means clustering algorithm on the following dataset for two clusters (K=2).
<table><tr><td></td><td>X</td><td>Y</td></tr><tr><td>1</td><td>185</td><td>72</td></tr><tr><td>2</td><td>170</td><td>56</td></tr><tr><td>3</td><td>168</td><td>60</td></tr><tr><td>4</td><td>179</td><td>68</td></tr><tr><td>5</td><td>182</td><td>72</td></tr><tr><td>6</td><td>188</td><td>77</td></tr></table>Euclidean distance
Distance matrix
Cluster s after every iteration</td><td>2M
2M
6M</td><td>10M</td><td>10M</td></tr>
</table>

Scheme Of Evaluation
Internal Assessment Test 3 – July 2023

Sub:	DATA MINING AND DATA WAREHOUSING							Code:	18CS641
Date:	5/7/2023	Duration:	90mins	Max Marks:	50	Sem:	VI	Branch:	ISE

Note: Answer Any full five questions

1. What is Baye's Theorem? Explain how it is used for classification with example?

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:

	Outlook	Play
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No

12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35

P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

2. Explain with example, how to build decision tree using Hunt's algorithm.

Hunt's Algorithm

In Hunt's algorithm, a decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets. Let D_t be the set of training records that are associated with node t and $y = \{y_1, y_2, \ldots, y_c\}$ be the class labels. The following is a recursive definition of Hunt's algorithm.

Step 1: If all the records in D_t belong to the same class y_t, then t is a leaf node labeled as y_t.

Step 2: If D_t contains records that belong to more than one class, an **attribute test condition** is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition and the records in D_t are distributed to the children based on the outcomes. The algorithm is then recursively applied to each child node.

	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Figure 4.6. Training set for predicting borrowers who will default on loan payments.

- To illustrate how the algorithm works, consider the problem of predicting whether a loan applicant will repay her loan obligations or become delinquent, subsequently defaulting on her loan.
- A training set for this problem can be constructed by examining the records of previous borrowers.
- In the example shown in Figure 4.6, each record contains the personal information of a borrower along with a class label indicating whether the borrower has defaulted on loan payments.
- The initial tree for the classification problem contains a single node with class label Defaulted = No (see Figure4.7(a), which means that most of the borrowers successfully repaid their loans.
- **The tree, however, needs to be refined since the root node contains records from both classes. The records are subsequently divided into smaller subsets based on the outcomes of the *Home Owner* test condition,** as shown in Figure 4.7(b).
- The justification for choosing this attribute test condition will be discussed later.
- For now, we will assume that this is the best criterion for splitting the data at this point.
- Hunt's algorithm is then applied recursively to each child of the root node.
- From the training set given in Figure 4.6, notice that all borrowers who are home owners successfully repaid their loans.
- The left child of the root is therefore a leaf node labeled Defaulted = No (see Figure 4.7(b)).
- For the right child, we need to continue applying the recursive step of Hunt's algorithm until all the records belong to the same class. The trees resulting from each recursive step are shown in Figures 4.7(c) and (d).

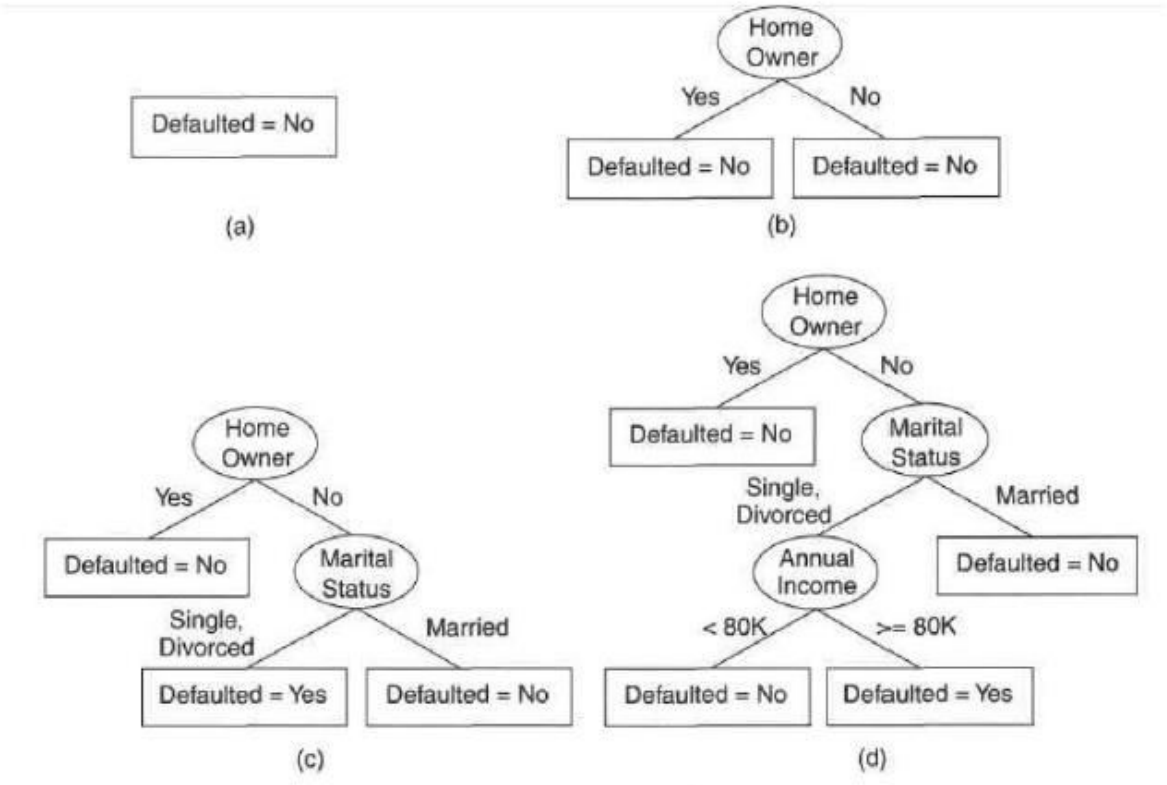


Figure 4.7. Hunt's algorithm for inducing decision trees.

3. Explain the measures for selecting the best split with example

Measures for Selecting the Best Split

There are many measures that can be used to determine the best way to split the records. These measures are defined in terms of the class distribution of the records before and after splitting.

Let $p(i|t)$ denote the fraction of records belonging to class i at a given node t. We sometimes omit the reference to node t and express the fraction as p_i. In a two-class problem, the class distribution at any node can be written as (p_0, p_1), where $p_1 = 1 - p_0$. To illustrate, consider the test conditions shown in Figure 4.12. The class distribution before splitting is $(0.5, 0.5)$ because there are an equal number of records from each class. If we split the data using the `Gender` attribute, then the class distributions of the child nodes are $(0.6, 0.4)$ and $(0.4, 0.6)$, respectively. Although the classes are no longer evenly distributed, the child nodes still contain records from both classes. Splitting on the second attribute, `Car Type`, will result in purer partitions.

The measures developed for selecting the best split are often based on the degree of impurity of the child nodes. The smaller the degree of impurity, the more skewed the class distribution. For example, a node with class distribution $(0, 1)$ has zero impurity, whereas a node with uniform class distribution $(0.5, 0.5)$ has the highest impurity. Examples of impurity measures include

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

The measures developed for selecting the best split are often based on the degree of impurity of the child nodes. The smaller the degree of impurity, the more skewed the class distribution. For example, a node with class distribution $(0, 1)$ has zero impurity, whereas a node with uniform class distribution $(0.5, 0.5)$ has the highest impurity. Examples of impurity measures include

$$\text{Entropy}(t) = -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t), \tag{4.3}$$

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2, \tag{4.4}$$

$$\text{Classification error}(t) = 1 - \max_i [p(i|t)], \tag{4.5}$$

where c is the number of classes and $0 \log_2 0 = 0$ in entropy calculations.

Figure 4.13 compares the values of the impurity measures for binary classification problems. p refers to the fraction of records that belong to one of the two classes. Observe that all three measures attain their maximum value when the class distribution is uniform (i.e., when $p = 0.5$). The minimum values for the measures are attained when all the records belong to the same class (i.e., when p equals 0 or 1). We next provide several examples of computing the different impurity measures.

Node N_1	Count
Class=0	0
Class=1	6

Gini $= 1 - (0/6)^2 - (6/6)^2 = 0$
Entropy $= -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$
Error $= 1 - \max[0/6, 6/6] = 0$

Node N_2	Count
Class=0	1
Class=1	5

Gini $= 1 - (1/6)^2 - (5/6)^2 = 0.278$
Entropy $= -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$
Error $= 1 - \max[1/6, 5/6] = 0.167$

Node N_3	Count
Class=0	3
Class=1	3

Gini $= 1 - (3/6)^2 - (3/6)^2 = 0.5$
Entropy $= -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$
Error $= 1 - \max[3/6, 3/6] = 0.5$

3 b) Explain characteristics of decision tree induction.

Characteristics of Decision Tree Based Classification:

Advantages :

- Decision tree induction is a nonparametric approach for building classification models. In other words, it does not require any prior assumptions regarding the type of probability distributions satisfied by the class and other attributes.
- Finding an optimal decision tree is an NP-complete problem
- Techniques developed for constructing decision trees are computationally inexpensive, making it possible to quickly construct models even when the training set size is very large. Once a decision tree has been built, classifying a test record is extremely fast, with a worst-case complexity of O(W), where ,W is the maximum depth of the tree.
- Decision trees, especially smaller-sized trees, are relatively easy to interpret.
- Decision tree algorithms are quite robust to the presence of noise.
- The presence of redundant attributes does not adversely affect the accuracy of decision trees.

4. With example, explain Agglomerative Hierarchical Clustering with example.

8.3 Agglomerative Hierarchical Clustering

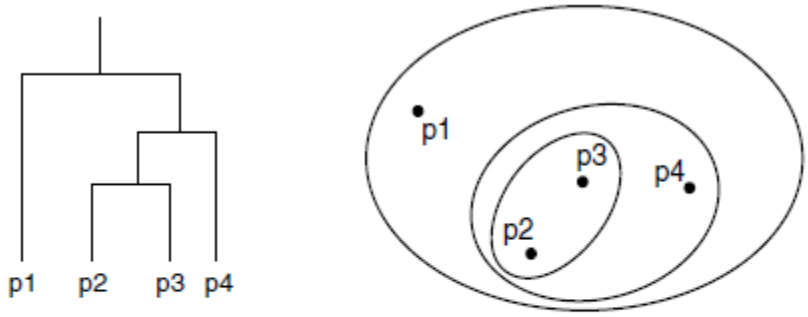
Hierarchical clustering techniques are a second important category of clustering methods. As with K-means, these approaches are relatively old compared to many clustering algorithms, but they still enjoy widespread use. There are two basic approaches for generating a hierarchical clustering:

Agglomerative: Start with the points as individual clusters and, at each step, merge the closest pair of clusters. This requires defining a notion of cluster proximity.

Divisive: Start with one, all-inclusive cluster and, at each step, split a cluster until only singleton clusters of individual points remain. In this case, we need to decide which cluster to split at each step and how to do the splitting.

Agglomerative hierarchical clustering techniques are by far the most common, and, in this section, we will focus exclusively on these methods. A divisive hierarchical clustering technique is described in Section 9.4.2.

A hierarchical clustering is often displayed graphically using a tree-like diagram called a **dendrogram**, which displays both the cluster-subcluster



(a) Dendrogram. (b) Nested cluster diagram.

Figure 8.13. A hierarchical clustering of four points shown as a dendrogram and as nested clusters.

relationships and the order in which the clusters were merged (agglomerative view) or split (divisive view). For sets of two-dimensional points, such as those that we will use as examples, a hierarchical clustering can also be graphically represented using a nested cluster diagram. Figure 8.13 shows an example of these two types of figures for a set of four two-dimensional points. These points were clustered using the single-link technique that is described in Section 8.3.2.

8.3.1 Basic Agglomerative Hierarchical Clustering Algorithm

Many agglomerative hierarchical clustering techniques are variations on a single approach: starting with individual points as clusters, successively merge the two closest clusters until only one cluster remains. This approach is expressed more formally in Algorithm 8.3.

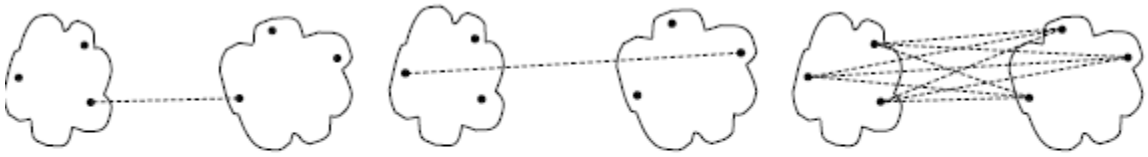
Algorithm 8.3 Basic agglomerative hierarchical clustering algorithm.

1: Compute the proximity matrix, if necessary.
2: **repeat**
3: Merge the closest two clusters.
4: Update the proximity matrix to reflect the proximity between the new cluster and the original clusters.
5: **until**

100%

Defining Proximity between Clusters

The key operation of Algorithm 8.3 is the computation of the proximity between two clusters, and it is the definition of cluster proximity that differentiates the various agglomerative hierarchical techniques that we will discuss. Cluster proximity is typically defined with a particular type of cluster in mind—see Section 8.1.2. For example, many agglomerative hierarchical clustering techniques, such as MIN, MAX, and Group Average, come from a graph-based view of clusters. **MIN** defines cluster proximity as the proximity between the closest two points that are in different clusters, or using graph terms, the shortest edge between two nodes in different subsets of nodes. This yields contiguity-based clusters as shown in Figure 8.2(c). Alternatively, **MAX** takes the proximity between the farthest two points in different clusters to be the cluster proximity, or using graph terms, the longest edge between two nodes in different subsets of nodes. (If our proximities are distances, then the names, MIN and MAX, are short and suggestive. For similarities, however, where higher values indicate closer points, the names seem reversed. For that reason, we usually prefer to use the alternative names, **single link** and **complete link**, respectively.) Another graph-based approach, the **group average** technique, defines cluster proximity to be the average pairwise proximities (average length of edges) of all pairs of points from different clusters. Figure 8.14 illustrates these three approaches.



(a) MIN (single link.) (b) MAX (complete link.) (c) Group average.

Figure 8.14. Graph-based definitions of cluster proximity

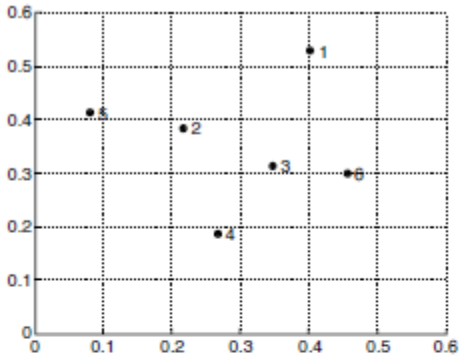


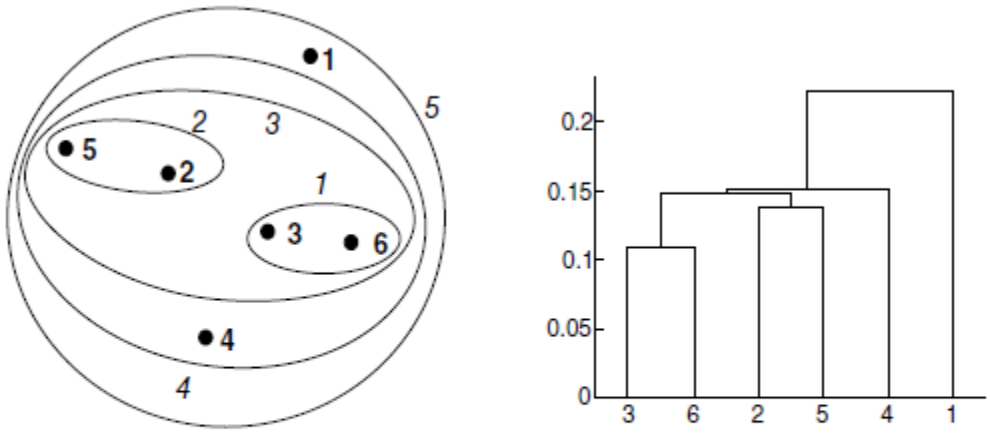
Figure 8.15. Set of 6 two-dimensional points.

Point	x Coordinate	y Coordinate
p1	0.40	0.53
p2	0.22	0.38
p3	0.35	0.32
p4	0.26	0.19
p5	0.08	0.41
p6	0.45	0.30

Table 8.3. xy coordinates of 6 points.

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Table 8.4. Euclidean distance matrix for 6 points.



(a) Single link clustering.

(b) Single link dendrogram.

Figure 8.16. Single link clustering of the six points shown in Figure 8.15.

is 0.11, and that is the height at which they are joined into one cluster in the dendrogram. As another example, the distance between clusters $\{3,6\}$ and $\{2,5\}$ is given by

$$\begin{aligned} dist(\{3,6\},\{2,5\}) &= \min(dist(3,2), dist(6,2), dist(3,5), dist(6,5)) \\ &= \min(0.15, 0.25, 0.28, 0.39) \\ &= 0.15. \end{aligned}$$

5. With time and space complexity, explain DBSCAN clustering algorit

8.4.2 The DBSCAN Algorithm

Given the previous definitions of core points, border points, and noise points, the DBSCAN algorithm can be informally described as follows. Any two core points that are close enough—within a distance *Eps* of one another—are put in the same cluster. Likewise, any border point that is close enough to a core point is put in the same cluster as the core point. (Ties may need to be resolved if a border point is close to core points from different clusters.) Noise points are discarded. The formal details are given in Algorithm 8.4. This algorithm uses the same concepts and finds the same clusters as the original DBSCAN, but is optimized for simplicity, not efficiency.

Algorithm 8.4 DBSCAN algorithm.

1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.

Time and Space Complexity

The basic time complexity of the DBSCAN algorithm is $O(m \times$ time to find points in the *Eps*-neighborhood), where m is the number of points. In the worst case, this complexity is $O(m^2)$. However, in low-dimensional spaces, points within a given distance of a specified point, and the time complexity can be as low as $O(m \log m)$. The space requirement of DBSCAN, even for high-dimensional data, is $O(m)$ because it is only necessary to keep a small amount of data for each point, i.e., the cluster label and the identification of each point as a core, border, or noise point.

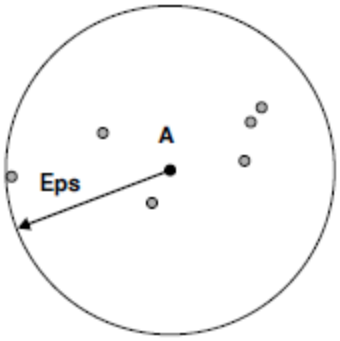


Figure 8.20. Center-based density.

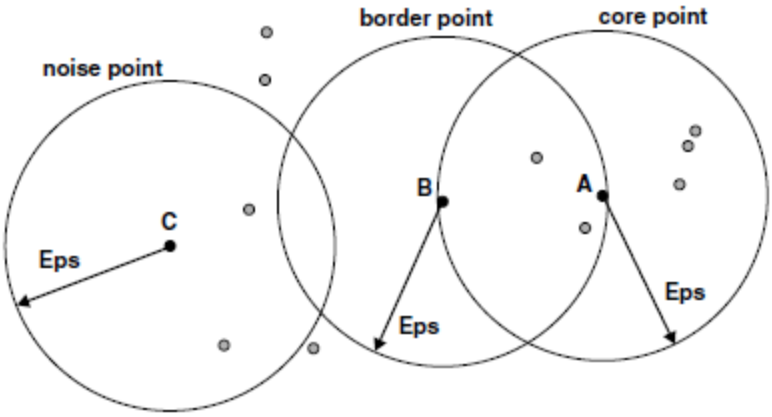


Figure 8.21. Core, border, and noise points.

6. Apply K-Means clustering algorithm on the following dataset for two clusters (K=2).

	X	Y
1	185	72
2	170	56
3	168	60
4	179	68
5	182	72
6	188	77

Given k = 2

Initial Centroid

Cluster	X	Y
k1	185	72
k2	170	56

- Calculate Euclidean distance for the next dataset (168,60)

$$\text{Distance } [(x,y), (a,b)] = \sqrt{(x-a)^2+(x-b)^2}$$

Distance from Cluster 2 $= \sqrt{(168-170)^2+(60-56)^2}$

(170,56) $= \sqrt{(-2)^2+(-4)^2}$

$= \sqrt{4+16}$

$= \sqrt{20}$

$= 4.472$

Final Assignment

Dataset No	X	Y	Assignment
1	185	72	1
2	170	56	2
3	168	60	2
4	179	68	1
5	182	72	1
6	188	77	1