Data Mining: Decision Tree & K-Nearest Neighbor Algorithm

B.Sc. 6th Semester
Department of Computer Science & Applications (Paper Code: CC3)
(Paper Code: DSE4)

Prof. Paulami Basu Ray

Contents

- Decision Tree
- K-Nearest Neighbor Algorithm

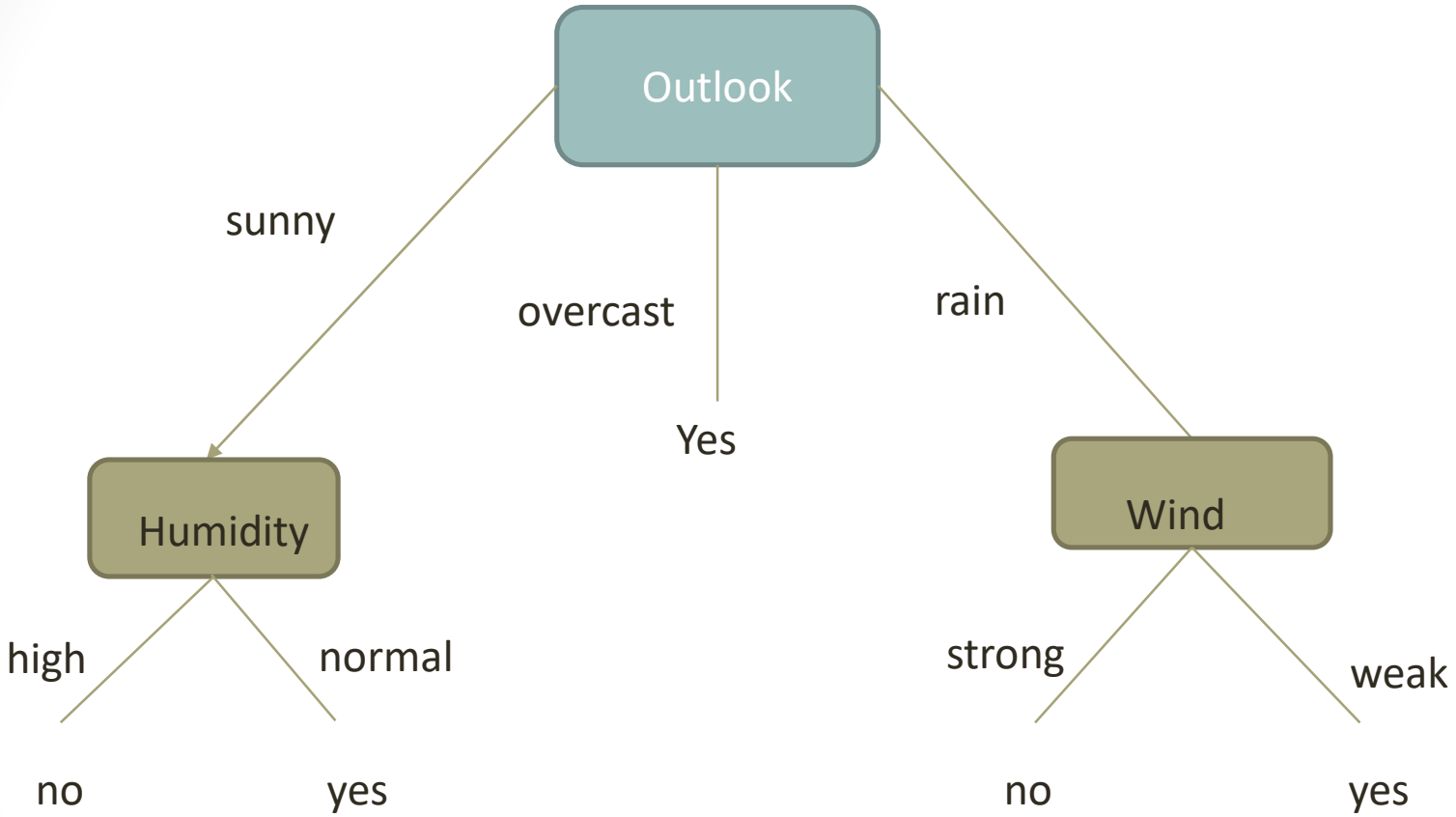
Decision Tree

- A Decision Tree is a Classification Algorithm where the training data is represented as a tree, where each node represents a test performed on an attribute and each edge represents the value of the attribute tested, leaves represent the class labels.
- This tree is basically a disjunction of conjunctions.
- A decision tree may be re-represented as IF-THEN-ELSE rules.

Training Data

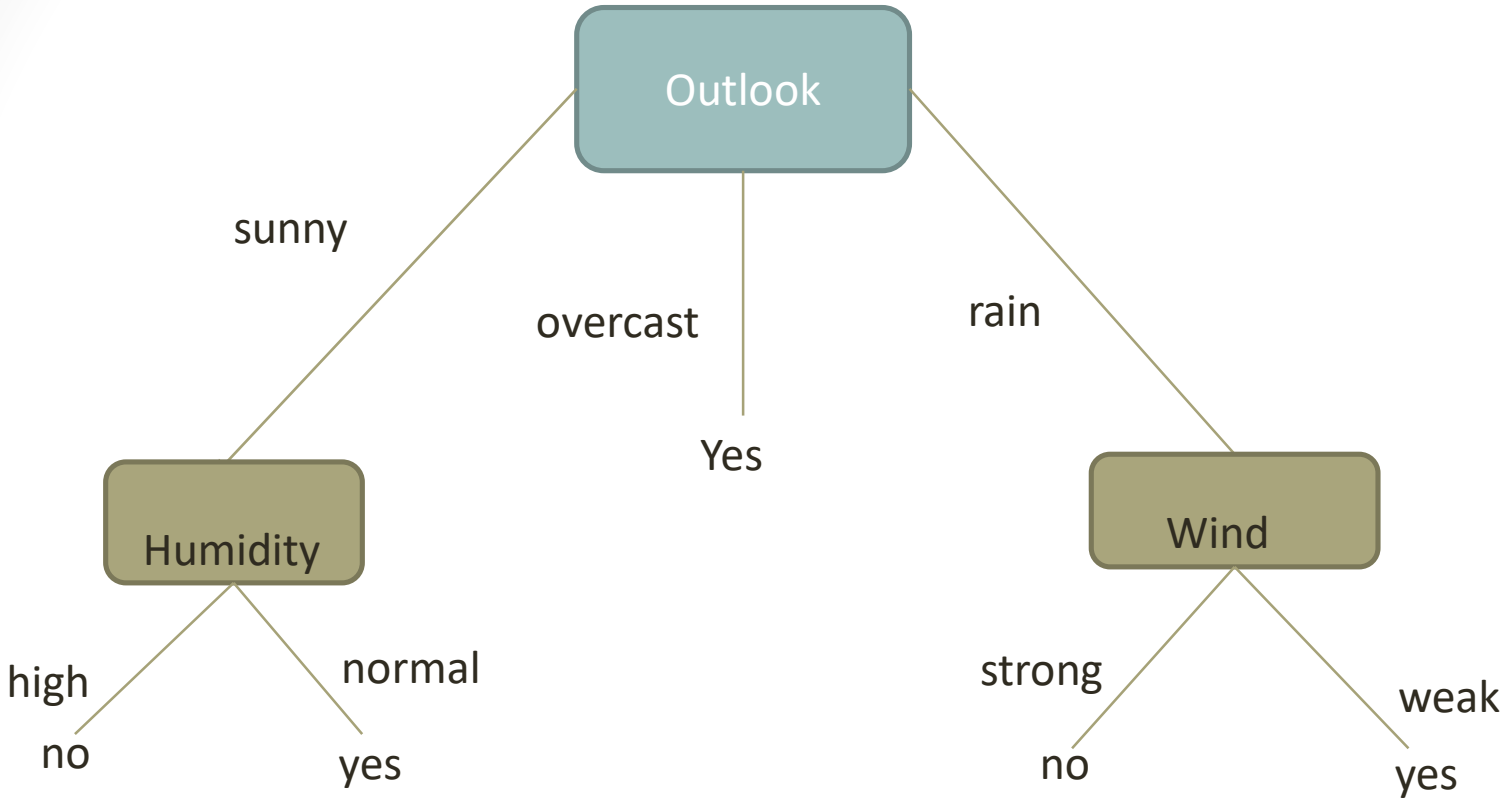
Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Weak	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision Tree



A decision tree for the concept PlayTennis. An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf (yes or no). This tree classifies Saturday mornings according to whether or not they are suitable for playing tennis.

Decision Tree



(Outlook = sunny ^ Humidity = normal) V
(Outlook = overcast) V
(Outlook = rain ^ Wind = weak)

ID3 (Iterative Dichotomiser 3)

- Start with empty tree.
- Main loop:
  - Split the “best” decision attribute(A) for next node.
  - Assign A as decision attribute for node.
  - For each value of A, create new descendant of node.
  - Sort training examples to leaf nodes.
  - If training examples perfectly classified, STOP else iterate over new leaf nodes.
- Grow tree just deep enough for perfect classification.

Which attribute is the Best Classifier?

- We will use a statistical property called Information Gain, that measures how well a given attribute separates the training examples, according to their target classification.
- Entropy measures the homogeneity of an attribute which in turn is used to calculate the information gain. It is the optimum number of bits to encode the information about the certainty or uncertainty of information.

Use Case: Training Data

RID	Age	Income	Student	Credit-rating	Buy's Computer
1	Youth	High	No	Fair	no
2	Youth	High	No	Excellent	no
3	Middle-aged	High	No	Fair	yes
4	Senior	Medium	No	Fair	yes
5	Senior	Low	Yes	Fair	yes
6	Senior	Low	Yes	Excellent	no
7	Middle-aged	Low	Yes	Excellent	yes
8	Youth	Medium	No	Fair	no
9	Youth	Low	Yes	Fair	Yes
10	Senior	Medium	Yes	Fair	yes
11	Youth	Medium	Yes	Excellent	yes
12	Middle-aged	Medium	No	Excellent	yes
13	Middle-aged	High	Yes	Fair	yes
14	Senior	Medium	No	Excellent	no

Use Case: Test Data

RID	Age	Income	Student	Credit rating	Buys computer
15	Senior	High	No	Fair	Yes
16	Youth	Low	Yes	Fair	Yes
17	Senior	High	No	Excellent	No

Calculation of Information Gain

In order to define information gain, we begin by defining entropy, that characterizes the (im)purity of an arbitrary collection of elements.

Entropy(D)= - $p_+ \log_2(p_+) - p_- \log_2(p_-)$

Where p+ is the proportion of positive examples and p- is the proportion of negative examples. This can be extended in case of multi-class classification.

$$\text{Entropy(D)} = \sum_{i=1}^{c} -pi \lg pi$$

Entropy([9+,5-])= -9/14 X $\log_2(9/14)$ – 5/14 X $\log_2(5/14)$ =0.940

- Entropy is 0 if all members of D belong to the same class.
- Entropy is 1 when the collection contains an equal number of positive and negative examples.
- If the collection contains unequal number of positive and negative examples, the entropy is between 0 and 1.

Calculation of Information Gain

- Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data.
- The measure to use, called information gain, is the expected reduction in entropy caused by partitioning the examples, according to this attribute.
- The information gain, Gain(A) of an attribute A is defined as:

$Gain(A)=Entropy(D) - \sum_{j=1}^{v}(|Dj|/|D|)$ X Entropy(D_j) , where A ἐ D

$Info_A(D)= \sum_{j=1}^{v}(|Dj|/|D|)$ X Entropy(D_j) , where A ἐ D

$Gain(A)=Entropy(D)-Info_A(D)$

Calculation of Information Gain

Values(Age) = Youth, Middle-aged, Senior

$Info_{Age}(D)$= [5/14 {-1/4lg(1/4) – 3/4lg(3/4)}]+ [4/14 {-4/4lg(1)}] + [5/14{-3/4lg(3/4)-1/4lg(1/4)}]

$Info_{Age}(D)$= 0.694

Gain(Age)=0.940 – 0.694 = 0.246

Similarly,

Gain(Income)= 0.029

Gain(student)=0.151

Gain(credit_rating)=0.048

Youth

Middle_aged

Senior

D1

Income	Student	Credit_rating	Buys_Computer
High	No	Fair	no
High	No	Excellent	no
Medium	No	Fair	no
Low	Yes	Fair	yes
Medium	Yes	Excellent	yes

D2

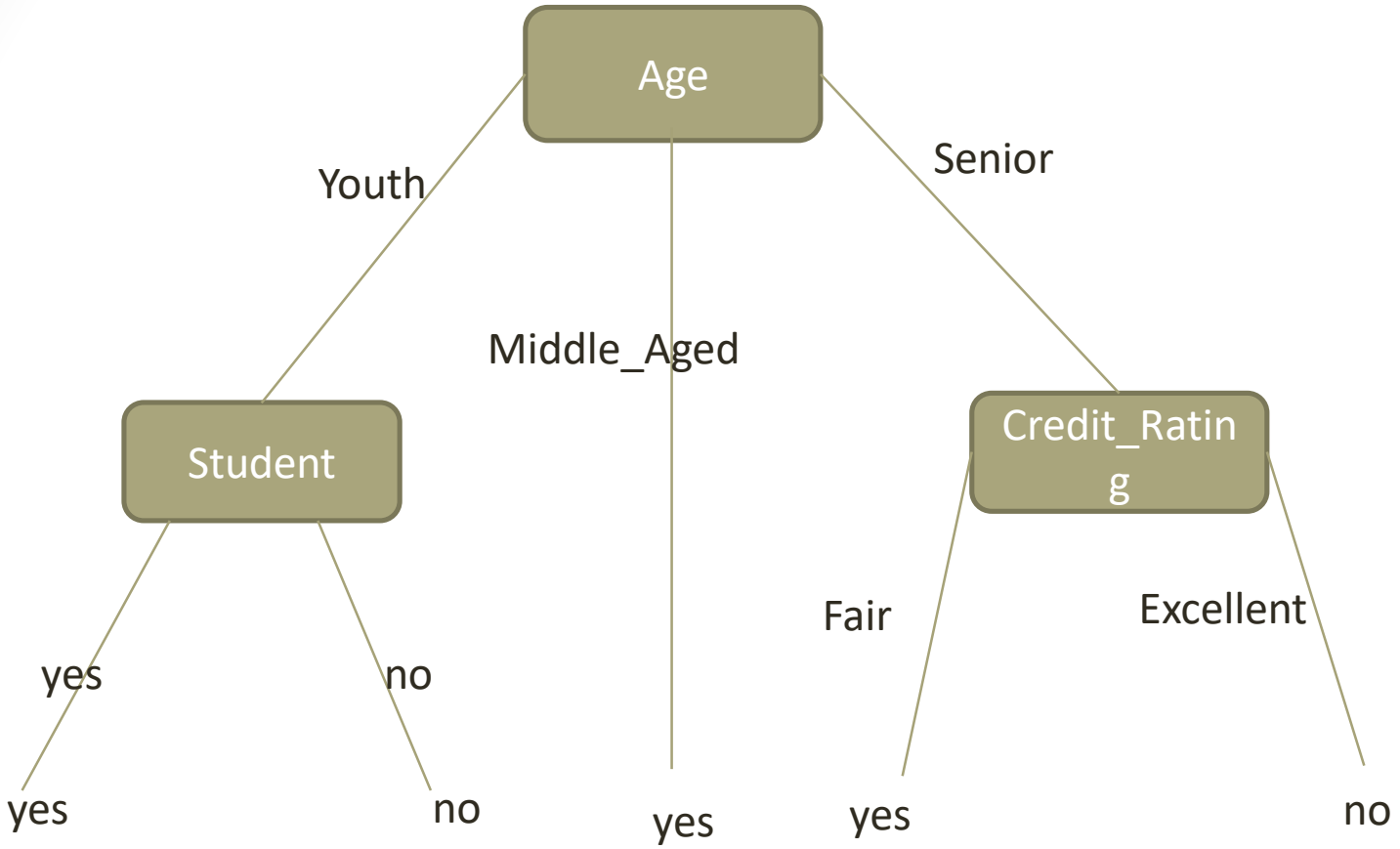
Income	Student	Credit_rating	Buys_Computer
Medium	No	Fair	yes
Low	Yes	Fair	yes
Low	Yes	Excellent	no
Medium	Yes	Fair	yes
Medium	No	Excellent	no

Buys_Computer= yes

Calculating Gain for D1 & D2

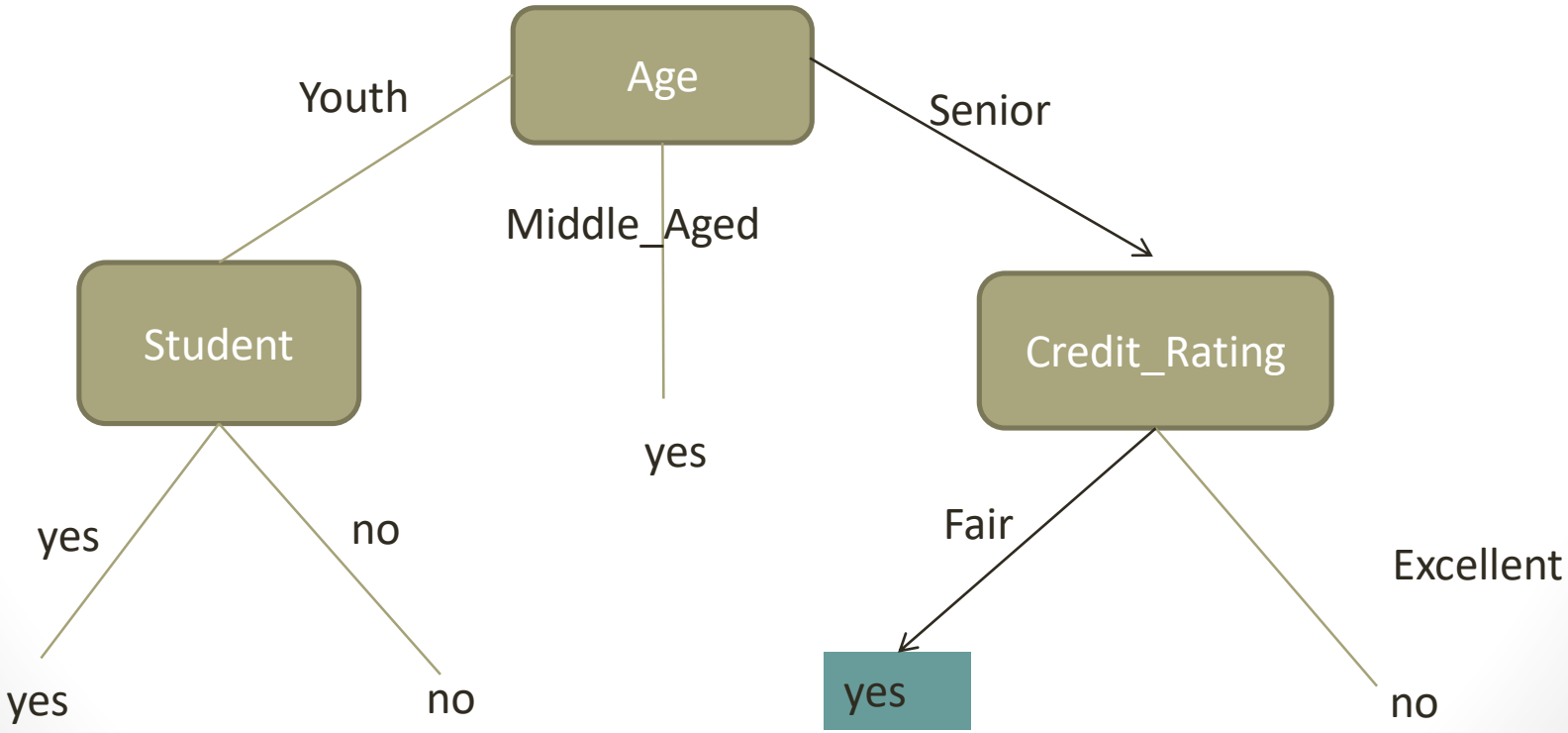
- Entropy(D1)= -2/5 X $\log_2(2/5)$ – 3/5 X $\log_2(3/5)$ =0.9709
- $\text{Info}_{\text{Income}}$ (D1)= 0.4
- Gain (Income) = 0.9709 - 0.4 = 0.5709
- Gain (Student) = 0.9709 - 0 = 0.9790
- Gain (Credit_Rating) = 0.9709 - 0.9507 = 0.0202
- Entropy(D2)=– 3/5 X $\log_2(3/5)$ -2/5 X $\log_2(2/5)$ =0.9709
- $\text{Info}_{\text{Income}}$ (D2)= 0.9507
- Gain (Income) = 0.9709 - 0.9507 = 0.0202
- Gain (Student) = 0.9709 - 0.9507 = 0.0202
- Gain (Credit_Rating) = 0.9709 - 0= 0.9709

Final Decision Tree



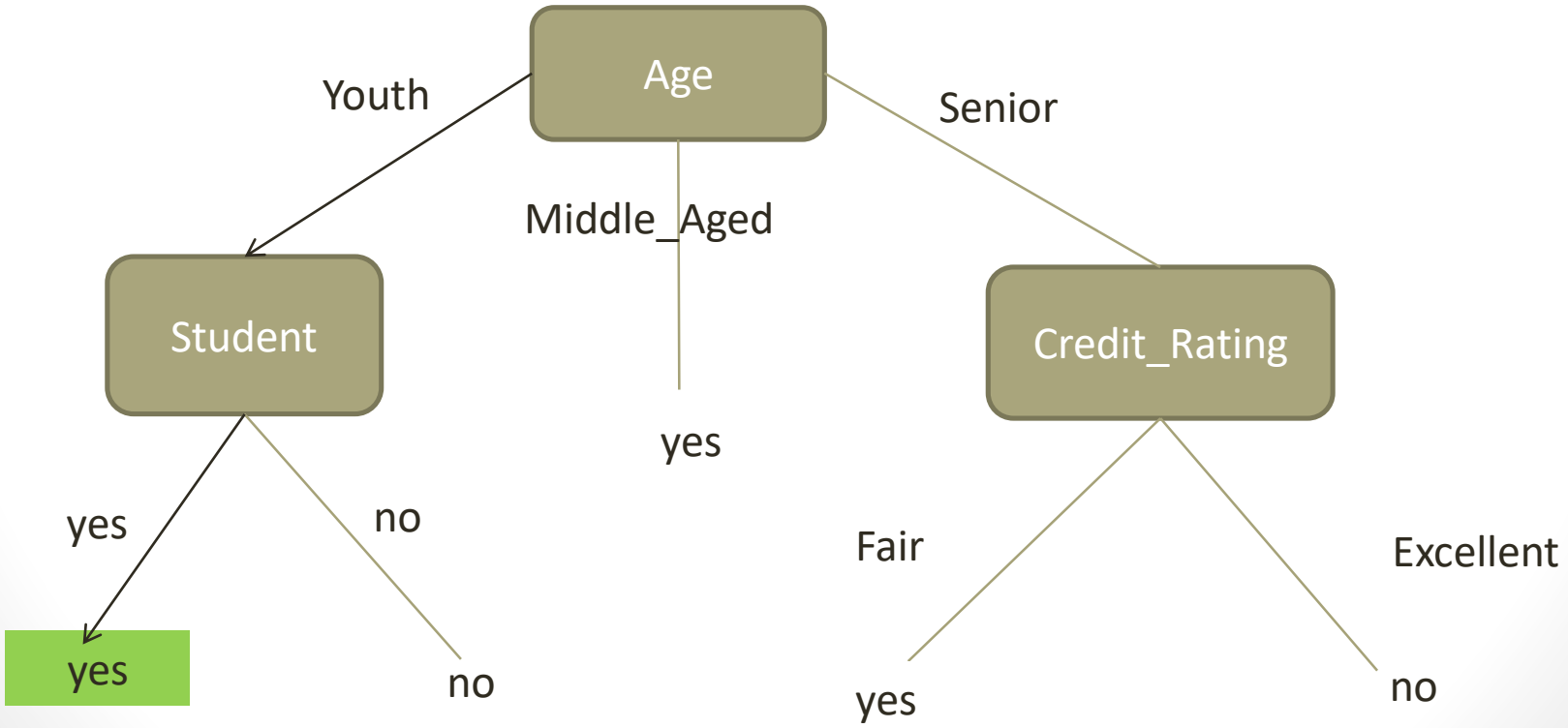
New Data

RID	Age	Income	Student	Credit_Rating	Buys_Computer
15	Senior	High	No	Fair	?



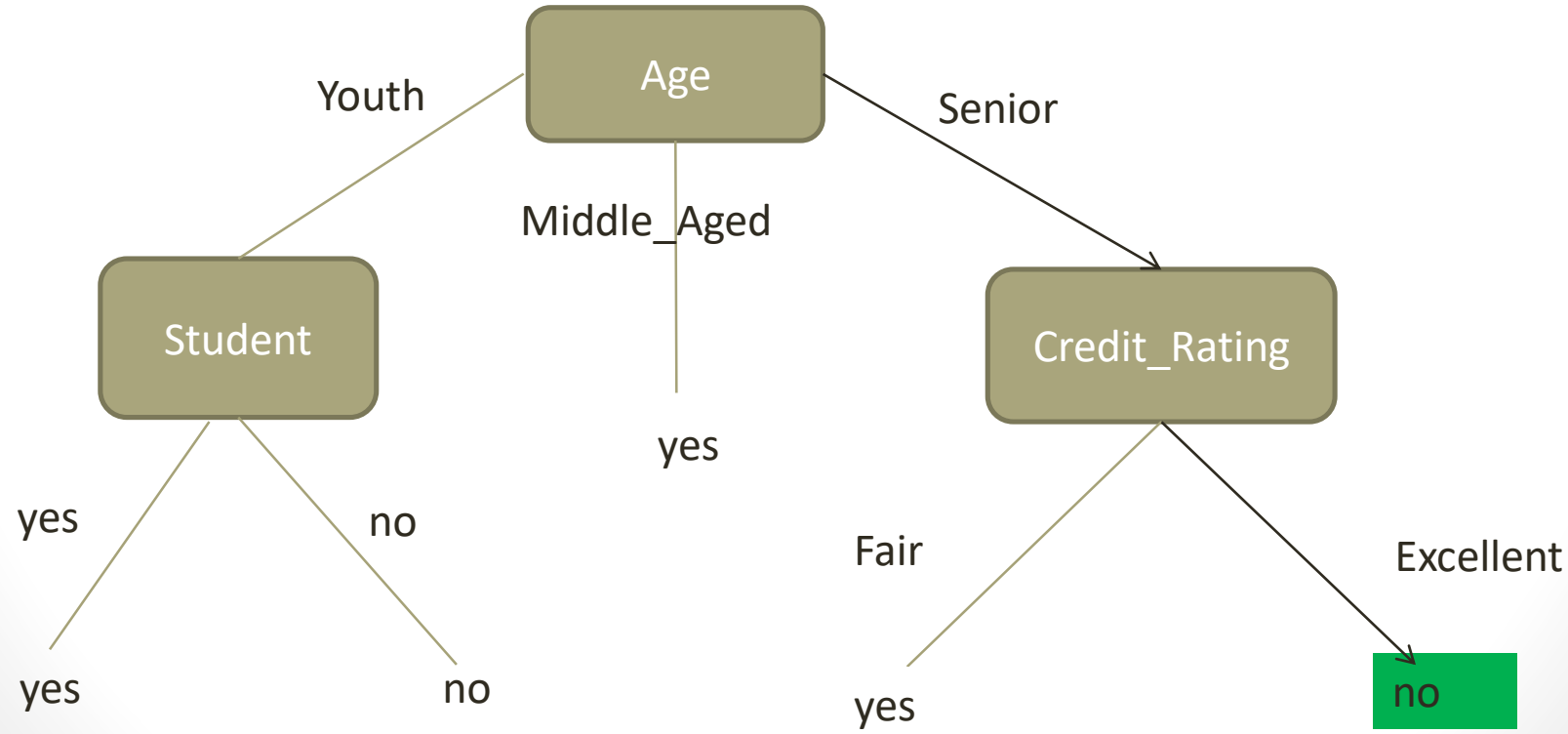
New Data

RID	Age	Income	Student	Credit_Rating	Buys_Computer
16	Youth	Low	Yes	Fair	?



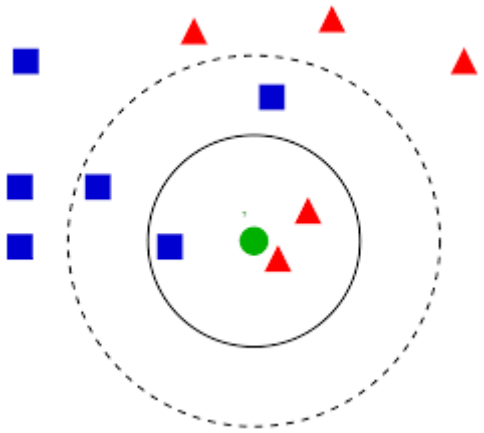
New Data

RID	Age	Income	Student	Credit_Rating	Buys_Computer
17	Senior	High	No	Excellent	?

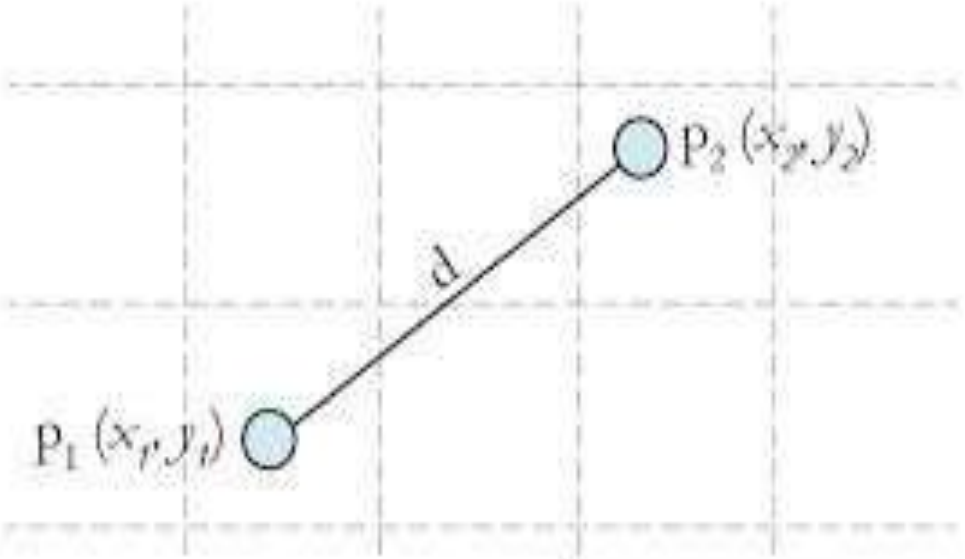


K Nearest Neighbor (KNN)

- It is an example of instance based learning approach, where there is need to store the entire training data.
- The KNN algorithm assumes that similar things exist in close proximity.



Euclidean Distance



Euclidean distance (d) $= \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$

Example/ Demonstration

Training Data:

X1	X2	CLASS(Y)
1	2	1
9	12	2
2	3	1
4	5	2
10	11	2

Test Point/New Point

X1	X2
3	4

Calculation of Distance of each Training Point from Test Point

X1	X2	CLASS(Y)	Dist.
1	2	1	√8
9	12	2	√100
2	3	1	√2
4	6	2	√5
10	11	2	√98

Sort the previous table in Ascending order of Distance(ignoring √)

X1	X2	CLASS(Y)	Dist.
2	3	1	2
4	6	2	5
1	2	1	8
10	11	2	98
9	12	2	100

Assuming the value of K=3

X1	X2	CLASS(Y)	Dist.
2	3	1	2
4	6	2	5
1	2	1	8
10	11	2	98
9	12	2	100

After Majority Voting : Y = 1