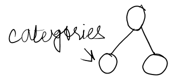
Decision Tree
02 April 2023 18:17

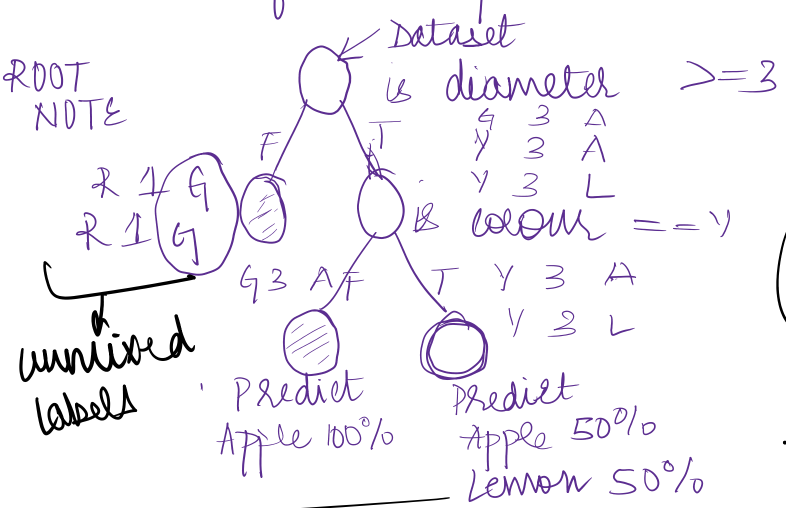
→ classification Algorithm



Color	Diameter	Label (Fruit)
Green	3	Apple
Yellow	3	Apple *
Red	1	Grape
Red	1	Grape –
Yellow	3	Lemon *
↓ C	↓ N	↳ Goal: Predict this label

No way to separate these 2 data pts. DT handles such case.

decision tree for above problem



→ Gini Impurity.

① We need to quantify how much a part. Q helps in unmixing labels.

INFORMATION GAIN

② Quantify how much Q helps reduce a label uncertainity

As purest tree as poss
↳ largest no of pure Nodes

Working of a DT

1. Which questions to ask & when?

Data → Tree

CART: Classi & Regre. Trees.

gives a procedure to decide when, which Qs to ask.

For a node to be pure:

AT EVERY NODE GI = 0

Creating a list of all possible Qs.
& iterating over them

QUANTIFY UNCERTAINITY

The BEST Q: it is the one that reduces the uncertainty the most

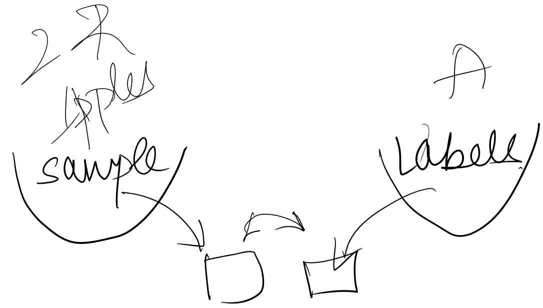
GI: how much uncertainity is there at every Node

IG: ~~tells~~ us how much a Q reduces the uncertainity

Impurity :

4

0



GI → incorrect label prediction at a node if at random a sample (eg) & a corresponding label is picked.

$\left(\frac{1}{5}\right)$ $\frac{1}{25}$ $\frac{1}{5}$

prob of 1 label, square, added for all 5

$1 - \left(\frac{1}{5}\right)^2 = 0.8 \rightarrow GI$

$$GINI(t) = 1 - \sum_K \left[p(C_k | t)^2\right]$$

$p(C_k/t)$ = prob of a node t being categorized as C_k.

t → no of data points

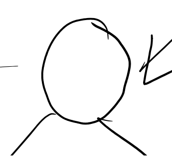
INFORMATION GAIN

→ helps us find Q that reduces uncertainity the most

↳ how much a Q helps unmix the labels

$\frac{1}{4}$ $\frac{1}{16}$ $\frac{6}{16}$

$\frac{1}{4}$ $\frac{1}{16}$



0.64

is color = green

2 →

2 →

A $2/5$ $4/25$

G $2/5$ $4/25$

$\frac{1}{4}$ $\frac{1}{16}$ 16

$\frac{2}{4}$ $\frac{4}{16}$

$1 - \frac{6}{16}$

$\frac{10}{16}$

Y 3 A
R 1 G
R 6 G
Y 3 L

Imp = 0.62

is color = green

T G 3 A

Imp = 0

$2 \rightarrow$
$1 \rightarrow$

5

$\frac{06}{25}$

$$\text{Avg} ~~\text{imp}~~ = \frac{4}{5} \times 0.62 + \frac{1}{5} \times 0$$

$$= 0.5$$

INFORmation Gain = Starting GI − A...

$$= 0.64 - 0.5 = 0.\ldots$$

Dry Run

In [33]:

```python
def build_tree(rows):
    """Builds the tree.

    Rules of recursion: 1) Believe that it works. 2) Start by checking
    for the base case (no further information gain). 3) Prepare for
    giant stack traces.
    """

    # Try partitioing the dataset on each of the unique attribute,
    # calculate the information gain,
    # and return the question that produces the highest gain.
    gain, question = find_best_split(rows)

    # Base case: no further info gain
    # Since we can ask no further questions,
    # we'll return a leaf.
    if gain == 0:
        return Leaf(rows)

    # If we reach here, we have found a useful feature / value
    # to partition on.
    true_rows, false_rows = partition(rows, question)
```

G $^2/5$ $4/25$

L. $1/5$ $1/25$

$1 - \frac{9}{25}$

avg GI of
next step

$14 \rightarrow$ max info gain

$\frac{2}{3}$ $\frac{1}{3}$

$\frac{4}{9}$ $\frac{1}{9}$ 1 $\frac{5}{9}$

$\frac{4}{9}$

```python
    # to partition on.
    true_rows, false_rows = partition(rows, question)

    # Recursively build the true branch.
    true_branch = build_tree(true_rows)

    # Recursively build the false branch.
    false_branch = build_tree(false_rows)

    # Return a Question node.
    # This records the best feature / value to ask at this point,
    # as well as the branches to follow
    # dependingo on the answer.
    return Decision_Node(question, true_branch, false_branch)
```

```python
training_data = [
    ['Green', 3, 'Apple'],
    ['Yellow', 3, 'Apple'],
    ['Red', 1, 'Grape'],
    ['Red', 1, 'Grape'],
    ['Yellow', 3, 'Lemon'],
]
```

$\rightarrow$ 0.64

G 3

G 3 A

T

~~Y 3~~ A

F Y 3 A

~~Y~~ 3 L

Y 3 L

0.5

F R I G

R I G

T = 9 1