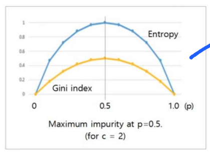
DT Revisited

Saturday 18 May 2024 5:35 AM

Impurity, Entropy and Gini index

- When there are 4 nodes (A, B, C, and D) as follows, the impurity of each node can be measured by the entropy or Gini index. Impurity is a measure of how much things with different properties are mixed within a node.
- Node A has the highest purity (lowest impurity) because it contains all of the same colors. Node C has the lowest purity (highest impurities) because it is a half-and-half mixture.
- Impurity can be used to express the amount of information held by each node.

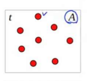


→ Binary Classification

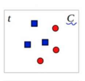
- Entropy
- Gini index

(c : the number of class)

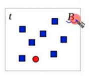
$$H(t)=-\sum_{i=1}^{c} p(i|t)\cdot\log_2 p(i|t) \qquad G(t)=\sum_{i=1}^{c} p(i|t)\cdot(1-p(i|t)) = 1-\sum_{i=1}^{c} p(i|t)^2$$



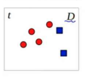
$$H(t)=-(\frac{8}{8}\times\log_2\frac{8}{8}+\frac{0}{8}\times\log_2\frac{0}{8})=0$$
$$G(t)=1-((\frac{8}{8})^2+(\frac{0}{8})^2)=0$$



$$H(t)=-(\frac{3}{6}\times\log_2\frac{3}{6}+\frac{3}{6}\times\log_2\frac{3}{6})=1.0$$
$$G(t)=1-((\frac{3}{6})^2+(\frac{3}{6})^2)=0.5$$



$$H(t)=-(\frac{1}{8}\times\log_2\frac{1}{8}+\frac{7}{8}\times\log_2\frac{7}{8})=0.544$$
$$G(t)=1-((\frac{1}{8})^2+(\frac{7}{8})^2)=0.219$$



$$H(t)=-(\frac{4}{6}\times\log_2\frac{4}{6}+\frac{2}{6}\times\log_2\frac{2}{6})=0.918$$
$$G(t)=1-((\frac{4}{6})^2+(\frac{2}{6})^2)=0.444$$

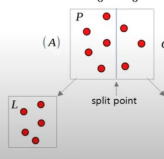
Information Gain (IG)

- The information gain (IG) can be measured through the change in impurities due to tree node splitting. Lowering impurity when splitting nodes is a good thing and provides informational benefits. Information gain is measured as the difference in weighted average impurities before and after splitting.
- For node A, the impurity before splitting is already 0, so there is no benefit from splitting (no need to split). For node B, splitting lowers impurities and produces an information gain of 0.278 (need to split).
- Decision tree algorithms build a tree by finding the split points that provide the greatest information gain.

Range : [0, 1]

- Gini index: $G(t)=1-\sum_{i=1}^{c} p(i|t)^2$
- Information gain: $IG=G(P)-\frac{N_L}{N}\times G(L)-\frac{N_R}{N}\times G(R)$

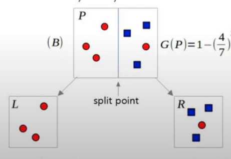
$$IG=0-\frac{5}{8}\times0-\frac{3}{8}\times0=0$$



$$G(P)=1-(\frac{8}{8})^2-(\frac{0}{8})^2=0$$

$$G(L)=1-(\frac{5}{5})^2-(\frac{0}{5})^2=0 \qquad G(R)=1-(\frac{3}{3})^2-(\frac{0}{3})^2=0$$

$$IG=0.49-\frac{3}{7}\times0-\frac{4}{7}\times0.375=0.278 > 0$$



$$G(P)=1-(\frac{4}{7})^2-(\frac{3}{7})^2=0.490$$

$$G(L)=1-(\frac{3}{3})^2-(\frac{0}{3})^2=0 \qquad G(R)=1-(\frac{3}{4})^2-(\frac{1}{4})^2=0.375$$

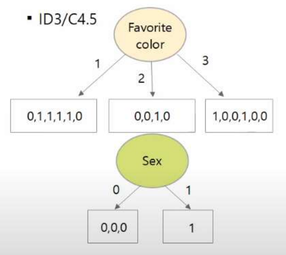
- CART uses binary trees. It uses not only the important feature at the top of the tree, but also the best split point for the feature, allowing the depth to be made smaller. It is suitable for ensemble techniques that use many small trees. However, the CART algorithm may not be suitable for categorical features. In particular, it is not very good for nominal categorical data, such as ['red', 'green', 'blue']. On the other hand, CART can precisely process continuous numerical data. (You can find the exact split point). In order to use CART, categorical data must be converted to numeric type in advance. Overall, it has more advantages than ID3.

- For ordinal features (Pclass) with size order, we can create trees as following.
- Pclass feature is in the following order: [Level 1, Level 2, Level 3]. We can assume that the level 1 is 'high', level 2 is 'medium', and level 3 is 'low'.
- In CART, there is no problem because Pclass is an ordinal feature, but there may be a problem if it is a nominal feature, such as ['red', 'green', 'blue']. Nominal features need to be converted to numbers using one-hot encoding or something like that.

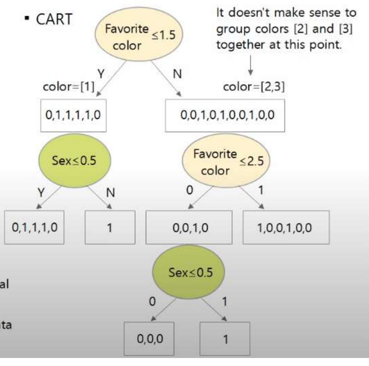
- For nominal feature with no size or order, you might consider the following trees. ID3 tree is fine, but CART tree may have problem.
- The "Favorite color" feature is the passenger's favorite color, marked using label encoding as [1=red, 2=blue, 3=yellow]. Colors have no concept of large or small size. If the colors are split into [1] and [2, 3], It doesn't make sense to group colors [2] and [3] together. In the case of Sex, even though it is nominal feature, there is no problem because there are only two types [0, 1].

No	Favorite color	Sex	Survived
1	1	male	0
2	3	female	1
3	1	male	1
4	3	male	0
5	2	male	0
6	2	male	0
7	1	female	1
8	2	female	1
9	3	female	0
10	1	male	1
11	1	male	1
12	3	male	1
13	3	female	0
14	3	female	0
15	1	male	0
16	2	male	0

- ID3/C4.5



- CART



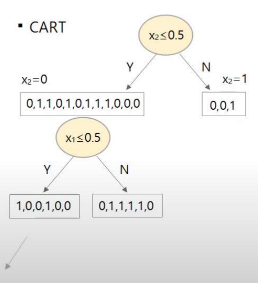
- In ID3/C4.5, there is no problem in processing nominal data because all cases are divided simultaneously.
- CART is a binary tree, so it cannot handle nominal data directly.

- For nominal data, you can consider converting to one-hot encoding as follows. Instead of the "Favorite color" feature, we can create 3 new features (x_1, x_2, x_3). Since x_1, x_2, and x_3 are independent and all equidistant, there is no concept of size or order. Also, since there are only 0 and 1, the problem shown on the previous page does not occur.
- However, as the number of classes of "Favorite color" increases, the number of new features also increases, and the number of 0 also increases (sparsity). This will cause the tree to lean in one direction and increase its depth.
- Pruning deep tree in this way may reduce performance.
- In particular, ensemble models using shallow trees may further degrade performance.
- For reference, ensemble models such as LightGBM is able to process large amounts of data quickly by merging sparse features. It is called Exclusive Feature Bundling. Then x1, x2, and x3 features might be merged back into one feature, the original feature of "Favorite color".
- One-hot encoding can be considered for nominal data, but analysts must be aware of these problems and be prepared in advance.
- When using label encoding, one-hot encoding, or binary encoding methods, data scientists must make careful decisions considering their learning objectives and the characteristics of the dataset.

Favorite color $\rightarrow \begin{cases} 1 \rightarrow (1\ 0\ 0) \\ 2 \rightarrow (0\ 1\ 0) \\ 3 \rightarrow (0\ 0\ 1) \end{cases}$

No	x_1	x_2	x_3	Sex	Survived
1	1	0	0	male	0
2	0	0	1	female	1
3	1	0	0	male	1
4	0	0	1	male	0
5	0	1	0	male	0
6	0	1	0	male	0
7	1	0	0	female	1
8	0	1	0	female	1
9	0	0	1	female	0
10	1	0	0	male	1
11	1	0	0	male	1
12	0	0	1	male	1
13	0	0	1	female	0
14	0	0	1	female	0
15	1	0	0	male	0

- CART



As data increases and the number of one-hot features increases, the tree becomes skewed to the left. The tree grows deeper.

(This is just a reformat of my comment from 2016...it still holds true.)

The accepted answer for this question is misleading.

As it stands, sklearn decision trees do not handle categorical data - see issue #5442.

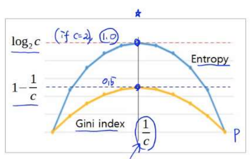
The recommended approach of using Label Encoding converts to integers which the `DecisionTreeClassifier()` will treat **as numeric**. If your categorical data is not ordinal, this is not good - you'll end up with splits that do not make sense.

Using a `OneHotEncoder` is the only current valid way, allowing arbitrary splits not dependent on the label ordering, but is computationally expensive.

Multi-class classification

Binary classification (c=2)

- For multiple classes, the entropy and Gini index can be calculated as follows, so the tree can be created using information gain in the same way as the previous process. (c : the number of class).



- Entropy
- Gini index

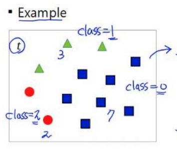
# of C = 3

$$H(t)=-\sum_{i=1}^{c} p(i|t)\log_2 p(i|t) \qquad G(t)=\sum_{i=1}^{c} p(i|t)(1-p(i|t))=1-\sum_{i=1}^{c} p(i|t)^2$$

$i = [0, 1, 2]$

- c = the number of class
- If $c > 2$, the entropy can be greater than 1.0 and the Gini index can also be greater than 0.5

- Example



class=0 class=1 class=2

$$H(t) = -(\frac{7}{12}\times\log_2\frac{7}{12}+\frac{3}{12}\times\log_2\frac{3}{12}+\frac{2}{12}\times\log_2\frac{2}{12}) = \boxed{1.384}$$

max H(t) = 1.0, if c=2

max G(t) = 0.5

$$G(t) = 1-(\frac{7}{12})^2-(\frac{3}{12})^2-(\frac{2}{12})^2 = \boxed{0.569}$$

$\frac{9}{3} = 3$

- Maximum if there are (n/c) pieces of data per class inside a node.

max H, max G

n=9

$$p_1 = p_2 = \dots p_c = \frac{n/c}{n} = \frac{1}{c}$$

$$H(t)_{max} = -\sum_{i=1}^{c}\frac{1}{c}\cdot\log_2\frac{1}{c} = \boxed{\log_2 c}$$

$$G(t)_{max} = 1-\sum_{i=1}^{c}\frac{1}{c^2} = \boxed{1-\frac{1}{c}}$$

Regression tree

test data

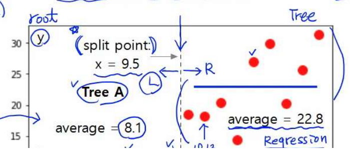
x	y
11.5	?

IG $\hat{y} = \bar{y}$ $MSE = \frac{1}{N}\sum_i (y_i - \hat{y}_i)^2$

- Regression trees use mean square error (MSE) instead of entropy or Gini index. Select the (split point) with the smallest MSE.

(feature) (target value)

x	y
12.0	20.34
3.0	11.6
10.0	18.49
16.0	20.27



$$MSE=\frac{1}{N}\sum_{i=1}^{N}(y_i-\bar{y})^2$$

Mean-squared error for each node
It is the same as Variance

$(x \le 9.5)$

$$MSE_{left}=\frac{1}{9}\times((5.28-8.1)^2+(2.33-8.1)^2+\dots+(11.80-8.1)^2)=9.777$$

$(x > 9.5)$

Regression tree

test data

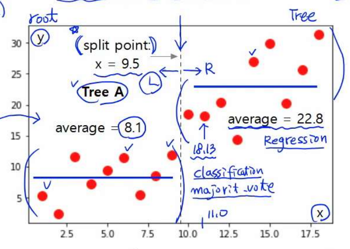
x	y
11.5	?

IG $\hat{y} = \bar{y}$ $MSE = \frac{1}{N}\sum_i (y_i - \hat{y}_i)^2$

- Regression trees use mean square error (MSE) instead of entropy or Gini index. Select the split point with the smallest MSE.

(feature) (target value)

x	y
12.0	20.34
3.0	11.6
10.0	18.49
16.0	20.27
5.0	9.43
18.0	31.34
1.0	5.28
15.0	29.84
2.0	2.33
8.0	8.52
11.0	18.13
7.0	5.40
13.0	14.42
4.0	7.20
9.0	11.80
17.0	25.60
6.0	11.38
14.0	26.96



$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2$$ ← Mean-squared error for each node. It is the same as Variance

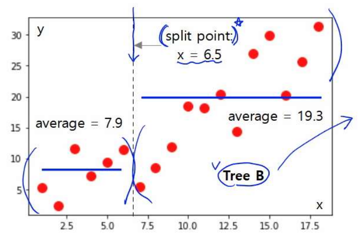
$(x \le 9.5)$

$$MSE_{\text{left}} = \frac{1}{9}\times\left((5.28-8.1)^2+(2.33-8.1)^2+\ldots+(11.80-8.1)^2\right)=9.777$$

$(x > 9.5)$

$$MSE_{\text{right}} = \frac{1}{9}\times\left((18.49-22.8)^2+(18.13-22.8)^2+\ldots+(31.34-22.8)^2\right)=30.077$$

$$MSE_A = 9.777\times\frac{9}{18}+30.077\times\frac{9}{18}=19.93$$ ← weighted average



$$MSE_{\text{left}} = \frac{1}{6}\times\left((5.28-7.9)^2+(2.33-7.9)^2+\ldots+(11.38-7.9)^2\right)=11.086$$

$$MSE_{\text{right}} = \frac{1}{12}\times\left((5.40-19.3)^2+(8.52-19.3)^2+\ldots+(31.34-19.3)^2\right)=62.327$$

$$MSE_B = 11.086\times\frac{6}{18}+62.327\times\frac{12}{18}=45.24$$ ← weighted average

$MSE_A < MSE_B$: Tree A is better.

- It works as a non-linear regression rather than a linear regression.
- The shallower the tree, the more likely it is to be underfitting, and the deeper the tree, the more likely it is to be overfitting.
- As with classification, pruning is necessary to prevent overfitting. Cost Complexity Pruning (CCP) can also be applied for regression.
- It can also be applied when there are multiple features (multiple regression).

$$\rightarrow C(T) = \underline{R(T)} + \alpha |T|$$

MSE SSR for regression

[illegible]