Decision Tree Algorithms

There are algorithms for creating decision trees: 1) ID3
2) C4.5
3) C5.0
4) CART (Classification and Regression Trees)

1) Classification Trees: (When the target variable categorical)

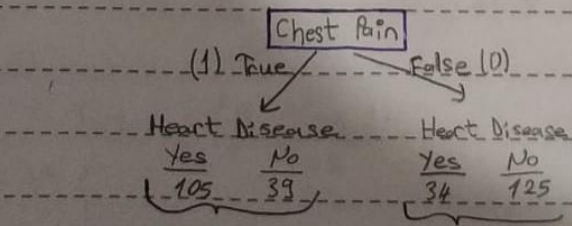
None of the leaf nodes are 100% "Target 1" or 100% "Target 0" they are all considered "impure".

↓

To determine which separation is best, we need a way to measure and compare "impurity".

"impurity" ölçmenin birçok yolu var. Aşağıda en çok kullanılan split'lere ayırmada impurity ölçen metriklerden bahsedildi.

1) Gini Index (Cost function used to evaluate splits)



Gini impurity = Left Node = $1 - (\text{probability of "Yes"})^2 - (\text{prob of "No"})^2$

Gini impurity = Right Node = $1 - (\text{prob of "Yes"})^2 - (\text{prob of "No"})^2$

../..../....

Gini impurity - left node => $1 - \left(\frac{105}{105+39}\right)^2 - \left(\frac{39}{105+39}\right)^2 = 0.395$

Gini impurity - right node => $1 - \left(\frac{34}{125+34}\right)^2 - \left(\frac{125}{125+34}\right)^2 = 0.336$

↳ Total gini impurity for using "Chest Pain" to seperate patients with and without heart disease is the weighted average of the leaf node impurities

* Gini impurity for Chest Pain => $\left(\frac{144}{144+159}\right) \cdot 0.395 + \left(\frac{159}{144+159}\right) \cdot 0.336 = 0.364$

↓

Ben bir split'i bir değişkenle böleceğim zaman Gini impurity scoru en düşük olanı alırım.

Not: Nerede duracağım? leaf node elde etmek için artık o leaf node'un bölünse bile daha saf hali bulunamıyorsa orada Split'e ayırma işlemi bitecek. Yani, impurity azalana dek bölme işlemi yapılır. Artık impurity azalmayacak durumda o node "leaf node" olarak kabul edilir.

Burada Feature'ın kategorik olduğu durumu ele aldık peki ya feature numeric ise, o zaman ne yapacağız?

In order to determine the best split, we need to iterate through all the features and consider the mid points between adjacent (komşu) training samples as a candidate split. We then need to evaluate the cost of the split (Gini) and find the optimal split (Lowest Gini).

Örneğin

X_1 numerik bir feature olsun. İlk olarak X_1'in tüm değerlerini artan sırayla yazıyoruz.

For each row gives us all the possible Gini Scores.

Feature	Value	Gini	
X_1	1.72	0.5	→
X_1	2.77	0.44	→
X_1	2.99	0.37	→
X_1	3.67	0.28	
//	3.96	0.16	→ (6.64+3.96)/2 = 5.30 → Threshold
//	6.64	0.0	→
//	7.44	0.16	
//	7.49	0.28	

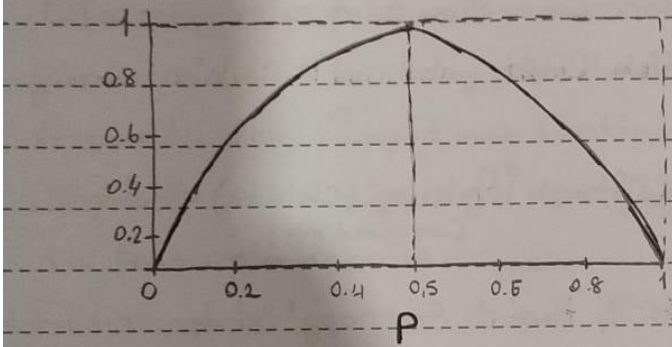
Yani kısacası her 2'li değerin orta noktası threshold (split point) adayıdır. En düşük Gini impurity veren midpoint split value olarak kullanılır.

Weight	Heart Disease	
155	No	→ Gini impurity?
180	Yes	→ //
190	No	→ //
220	Yes	→ //
225	Yes	

2) Entropy

→ If the sample is completely homogeneous, the Entropy = 0.

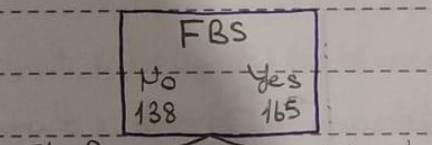
→ if the sample is an equally divided, the Entropy = 1.



$Entropy \Rightarrow -p\log_2 p - q\log_2 q$

$Entropy \Rightarrow -0.5\log_2 0.5 - 0.5\log_2 0.5 = 1$

* Burada da yine en düşük Entropiye sahip değişken split'i ayırır. (Lower the value of Entropy, higher is the purity of the node)



No	Yes	No	Yes
116	22	142	23

$$Entropy - fbs=0 \Rightarrow -\left(\frac{116}{116+22}\right)\cdot\log_2\left(\frac{116}{116+22}\right) - \left(\frac{22}{116+22}\right)\cdot\log_2\left(\frac{22}{116+22}\right)$$

$$Entropy - fbs=1 \Rightarrow -\left(\frac{142}{142+23}\right)\cdot\log_2\left(\frac{142}{142+23}\right) - \left(\frac{23}{142+23}\right)\cdot\log_2\left(\frac{23}{142+23}\right)$$

(Weighted) → $Total\ Entropy = \left(\frac{138}{138+165}\right)\cdot \underbrace{Entropy_{FBS=0}}_{0.632} + \left(\frac{165}{138+165}\right)\cdot \underbrace{Entropy_{FBS=1}}_{0.582}$

$= 0.605$

Entropy ile ilintili bir kavramda Information Gain'dir

Information Gain = Entropy(Parent) − Entropy(Children) (weighted average)

→ If the Entropy decreases due to a split, it will yield an Information Gain

* Örnek: Önceki örnekte, "Entropy(Children) (weighted avg.)" bulundu.

Bu değer (0.605), Parent Entropy'sinden çıkartılır Information Gain bulunur.

(Higher the value of IG; Higher the purity of nodes)

Fbs		
No	Yes	→ Parent
138	165	

Entropy of Parent => $-\left(\frac{138}{138+165}\right)\cdot\log_2\left(\frac{138}{138+165}\right)-\left(\frac{165}{138+165}\right)\cdot\log_2\left(\frac{165}{138+165}\right)$

= 0.994

Information Gain => 0.994 − 0.605 = 0.389

↓

* How much entropy we removed

2) Regression Trees (When target is Continuous.)

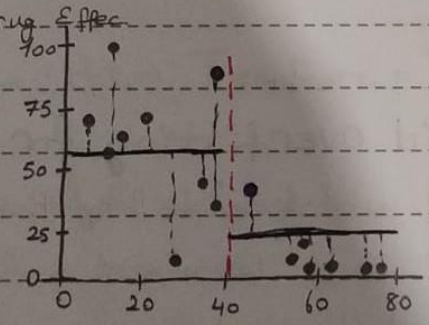
→ Each leaf represents a numeric value.

Eğer numeric feature'larımız var ise daha önce babsettiğimiz midpoint'ler aday threshold'lar olacak tek tek denenir. Ancak bu kez, Target'imiz Continuous olduğu için Gini impurity skora göre değil de RSS'e göre en iyi split'i ve feature'ı seçeriz.

Each leaf node corresponds to the Average Target value from a different cluster of observations.

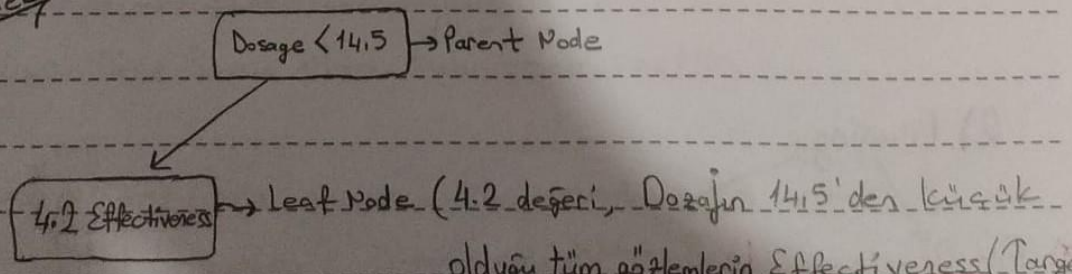
Örnek Target: Drug Effectiveness

Feature: Age (Continuous Feature)



Diyelim ki tüm 2'li komşu gözlemler arası midpointleri denedik ve yandaki ~~Mavi~~ gözlemler arasındaki threshold en iyi RSS'yi verdi. (Kırmızı → Threshold)

Örnek



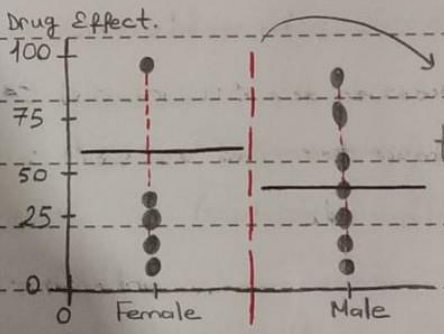
../..../....

Örnek Target: Drug Effectiveness

Feature: Gender (Categorical feature)

Feature Continuous olduğu durumda her 2'li gözlem arasındaki midpoint Aday Threshold (Split Point) olabiliyor ve içlerinde en düşük RSS'yi veren Threshold (split point) oluyordu.

Ancak, feature Categoric olduğu durumda Kategori sayısından 1 eksik sayıda threshold vardır.



We use that threshold to calculate the RSS.

Techniques to avoid overfitting for CART

1) Define some parameter which ends the Recursive splitting process.

for example: * Max tree depth

* Minimum number of samples required in a split

2) Pruning

a) Pre Pruning b) Post Pruning

a) Pre Pruning: (Limiting the tree before split happens)

↳ Sklearn içerisinde pre-pruning yapabilmek için max_depth, min_samples_leaf, min_samples_split'ten yararlanabiliriz.

b) Post Pruning: Decision Tree'de Post pruning, "Cost complexity pruning" metotu ile yapılır.

Aslında; Sklearn pre pruning için içerisinde zaten parametre barındırır (max depth vs.). Cost complexity pruning ise ağaç size'ını kontrol etmek için kullanılan post pruning bir yöntemdir. Bu pruning teknik "cost complexity parameter" ile yönetilir. Bu parametre Sklearn'de ccp_alpha dır.

Örnek Not ⇒ Tree Score = RSS + $\underbrace{\alpha |T|}_{\text{Tree complexity penalty}}$ (|T| → Total number of leaf node)

1) İlk olarak original full sized tree için Tree Score hesaplanır.

2) Daha sonra prune edilen tree'ler için ayrı ayrı Tree score hesaplanır.

3) En düşük Tree Score olan sub-tree, seçilir.

* Not ⇒ α arttıkça prune edilen node sayısı artar. Bu da Total impurity'i artırır.

Calculation of feature Importance

Çok matematiğine girmeyeceğim. Kısaca;

feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature.

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}$$

ni_j ⇒ the importance of node j

w_j ⇒ weighted number of samples reaching node j

C_j ⇒ the impurity value of node j

$left_j$ ⇒ child node from left split on node j

$right_j$ ⇒ child node from right split on node j

$$fi_{(i)} = \frac{\text{feature } i \text{ tarafından split edilen } ni_j \text{'lerin toplamı}}{\sum k \in \text{all nodes } ni_k} \rightarrow \text{(importance of node j)}$$

↓

(importance of feature i)

These can then be normalized to a value between 0 and 1 by dividing by the sum of all feature importance values:

$$norm\,fi_i = \frac{fi_{(i)}}{\sum j \in \text{all features } fi_j}$$

Not: Random Forest'ta, final feature importance, tüm ağaçların ortalaması alınarak bulunur.

↳ The sum of the feature's importance value on each tree is calculated and divided by the total number of trees.

Sklearn Decision Tree Parametreler

1) Criterion, default = "gini"

The function to measure the quality of split.

2) max_depth, default = None

Pre-pruning için kullanılır.

If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

3) min_samples_split, default = 2 (Pre-pruning için kullanılır)

The minimum number of samples required to split an internal node

(Split yapabilmek için gerekli minimum gözlem sayısı)

4) min_samples_leaf, default = 1 (Pre-pruning için kullanılır)

(Leaf node'da olabilecek minimum gözlem sayısı.

↳ It guarantees a minimum number of samples in every leaf.

5) min_weight_fraction_leaf, default = 0

Similar to min_samples_leaf but defined as a fraction of the total number of observations instead of an integer.

(min_samples_leaf kullanılıyorsa, bunu kullanmaya gerek yok)

6) max_features , default = None

The number of features to consider when looking for the best split.

↓

Her split'de, algoritmamız feature'lara bakar ve gini impurity veya entropiye göre 2 dal oluşturur. Her seferinde tüm feature'lara bakmak hem computationally heavy hem de overfit'e neden olur. Bu nedenle bir split için bazı feature'lara bakmak hem varyans kontrolü hem de hız açısından daha iyi dur.

* If None, then max_features = n_features.

7) max_leaf_nodes, default = None

If None, then unlimited number of leaf nodes.

Bu kadar sayıda leaf node varsa ağaç finalize edilir. Best nodes are defined as relative reduction in impurity. (Kaç tane leaf node olacağına karar verir ve bu leaf ler saflıklarına göre seçilir)

8) min_impurity_decrease, default = 0

Child node'a verilen threshold'dur. Yani; Parent node split edildi diyelim. Eğer bu parametreye verilen değerden daha az bir impurity decrease'e sahip ise split gerçekleşmez.

Weighted impurity decrease equation:

$$\left[N_t / N * \left(\text{impurity}_{\text{parent}} - N_{tr} / N_t * \text{right impurity} - N_{tl} / N_t * \text{left impurity} \right) \right]$$

N: Total number of samples, N_t: number of samples at current node (parent node)

N_{tl}: number of samples in left child, N_{tr}: number of samples in right child.

* → if the final impurity decrease is less than the minimum impurity decrease parameter, then the split will not be performed.

9) min_impurity_split, default = 0

Threshold for early stopping in tree growth.

Impurity, verilen threshold'dan yüksek ise node bölünür; aksi takdirde leaf node olur.

10) ccp_alpha, default = 0

Complexity parameter used for Minimal Cost-Complexity Pruning (Bir çeşit post pruning yöntemidir)

* By default, no post pruning performed.

→ Greater values of "ccp_alpha" increase the number of nodes pruned. As alpha increases, more of the tree is pruned, which increases the total impurity of its leaves.