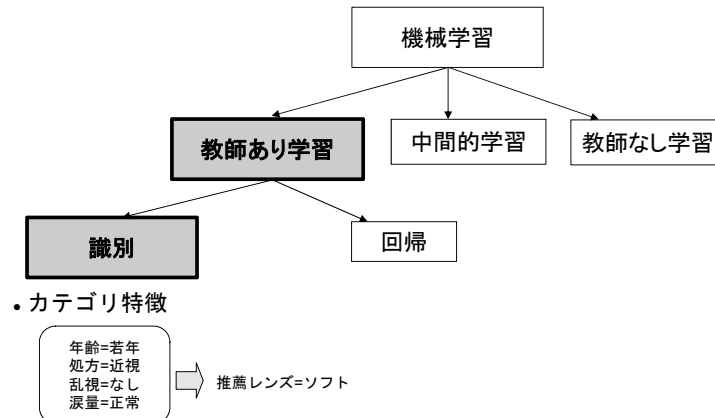


### 3. 識別 ―概念学習―

#### 3.1 カテゴリ特徴に対する「教師あり・識別」問題の定義

- 問題設定

- 教師あり学習  $\{(x_i, y_i)\}, \quad i = 1 \dots N$
- カテゴリ入力  $\rightarrow$  カテゴリ出力
- クラスの概念を得ることが目的



この章では、各次元がカテゴリデータである特徴ベクトル $x$ と、その正解クラスの情報 $y$ からなる学習データを用いて、「クラスの概念を得る方法」について説明します。  
データからクラスの概念を得ることを概念学習とよびます。  
クラスの概念を得ることができれば、未知の入力に対して、その概念に当てはまるかどうかを判定することで、当該クラスの事例であるかどうかを識別することができます。

# contact-lensesデータ

表 3.1 コンタクトレンズデータ (contact-lens.arff)

No.	age	spectacle-prescrip	astigmatism	tear-prod	contact-lenses
1	young	myope	no	reduced	none
2	young	myope	no	normal	soft
3	young	myope	yes	reduced	none
4	young	myope	yes	normal	hard
5	young	hypermetrope	no	reduced	none

表 3.2 コンタクトレンズデータの特徴値

特徴	値
age(年齢)	{young, pre-presbyopic, presbyopic} (若年, 老眼前期, 老眼)
spectacle-prescrip(眼鏡)	{myope, hypermetrope} (近視, 遠視)
astigmatism(乱視)	{no, yes} (なし, あり)
tear-prod-rate(涙量)	{reduced, normal} (減少, 正常)
contact-lenses(クラス)	{soft, hard, none} (ソフト, ハード, なし)


説明に用いるコンタクトレンズデータは、目の年齢、眼鏡の種類、乱視の有無、ドライアイの状況という特徴を用いて、「ソフトコンタクトレンズの使用を勧める」・「ハードコンタクトレンズの使用を勧める」・「コンタクトレンズの使用を勧めない」という概念を獲得するための架空のデータです。

## 3.2 概念学習とバイアス

- 概念学習とは

- 正解の概念を説明する特徴ベクトルの性質（論理式）を求めること
- 「softレンズを勧める」という概念の例  
(乱視=あり)  $\wedge$  (ドライアイ=なし)  $\Rightarrow$  soft

- 学習の方法

- 可能な論理式が少数の場合
  - 正解概念の候補を絞り込んでゆく（初期の概念学習）
- 可能な論理式が多数の場合
  - バイアス（偏見）をかけて探索する  **決定木**

カテゴリ特徴の場合、概念は論理式で表すことができます。

たとえば、「softレンズを勧める」という概念を、乱視があって、かつ、ドライアイがない、ならばsoftレンズを勧める、のように表すのが論理式による表現です。

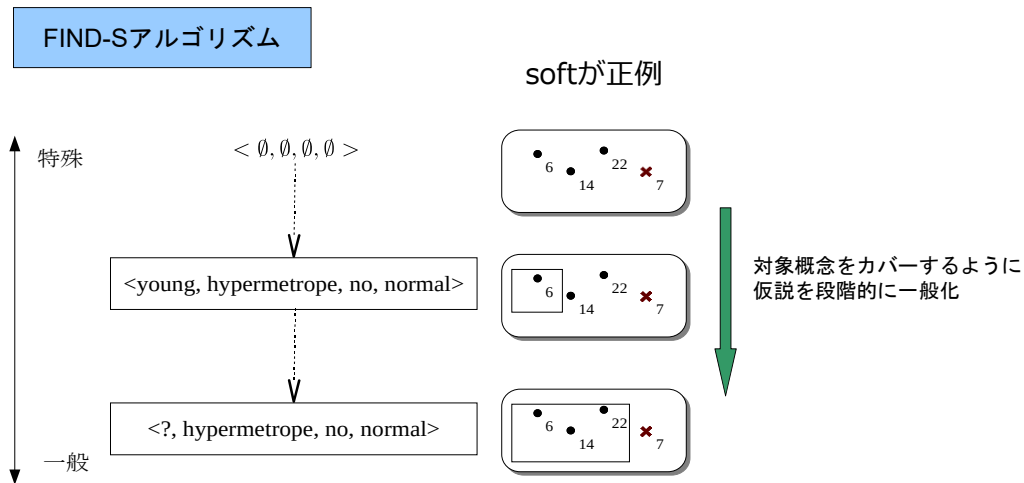
「特徴名 = 値」の表現をリテラルとよび、リテラルをAND（論理積）やOR（論理和）で結合したものが論理式です。

概念を学習する方法は大きく分けて二通りあります。

一つ目は、論理式の形式を制限することで正解候補を少数に限定して、データを用いて候補を絞り込んでゆく方法です。

二つ目は、形式の制限のない多数の正解候補に対して、特定の偏見をもって探索を行い、もっとも早く見つかったものを正解とする方法です。

### 3.2.1 初期の概念学習



概念学習の研究における初期の段階では、論理式の形式を制限することで正解候補を少数に限定し、データを用いて候補を絞り込んでゆく方法が考案されました。

FIND-S アルゴリズムは、正解仮説をリテラルの論理積(AND)結合に制限します。

このように、仮説に対して課す制約をバイアスとよびます。

FIND-S アルゴリズムの探索の最初は、もっとも特殊な仮説（いかなる事例も正ではない）からスタートします。

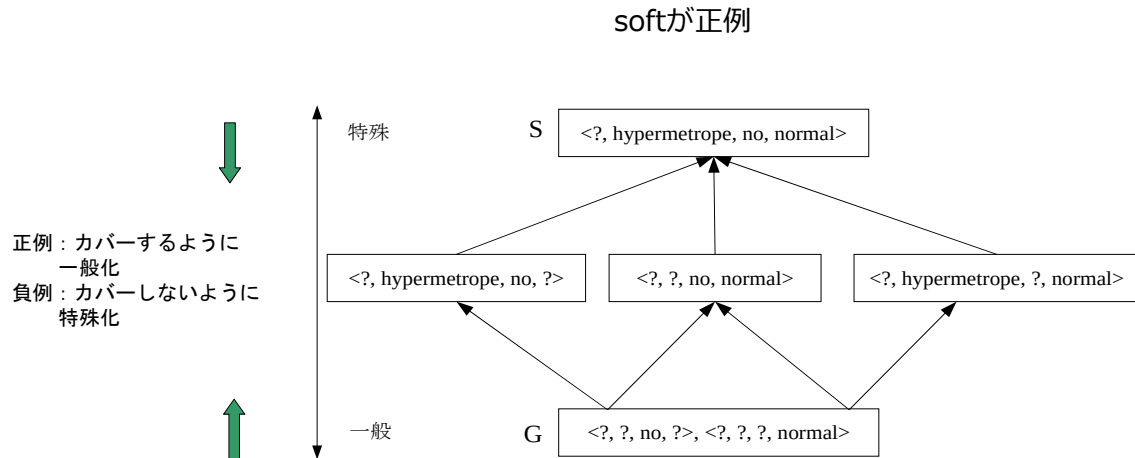
そして、学習には正例のみを用います。

正例を一つずつ読み込んで、その事例の値を受け入れるように仮説を最低限一般化することを繰り返します。

一般化とは、あるリテラルを取り除き、仮説がカバーする概念を広げることです。

### 3.2.1 初期の概念学習

#### 候補削除アルゴリズム



同様のアプローチである候補削除アルゴリズムは、負例も学習に用います。  
負例の場合は、もっとも一般的な仮説（いかなる事例も正である）からスタートし、負例をカバーしないように最低限特殊化します。  
特殊化とは、あるリテラルを追加して、ある仮説がカバーする概念を狭めることです。

## 3.2.2 概念学習のバイアスを考える

- 初期の概念学習
  - 学習対象の概念にバイアス（偏見）をかけて絞り込み
    - 求める論理式を「リテラル（特徴名＝値）のAND結合」に限定
  - 正解概念の候補数： $4 \times 3 \times 3 \times 3 + 1 = 109$
- 問題点
  - リテラルのOR結合が表現できないので、正解概念が仮説空間内に存在しないことが多い
    - 例）「age=young or age=pre-presbyopic」が表現できない

FIND-S アルゴリズムや候補削除アルゴリズムは、求める論理式を「リテラルのAND結合」に限定することで、正解概念の候補数を絞り込んでいます。

このように仮説を限定した場合でも、学習データを説明できる仮説が見つかった場合は、その条件を満たす未知のデータも正しく分類できる可能性は高くなります。

つまり、仮説の表現に関するバイアスが強い状況にも関わらず見つかった概念は、信頼してもよいだろうということがいえます。

しかし、これらのアルゴリズムは、小規模かつ特殊なデータでない限りは、このような概念が見つかる可能性は低くなります。

具体的には、ある特徴のとりえる値が複数でもよいような、リテラルのOR結合が表現できません。

## 3.2.2 概念学習のバイアスを考える

- 単純にリテラルのOR結合を許す場合
  - 正解概念の候補数が増大する
    - 2の事例数乗  $2^{24}=16777216$
  - 正例のOR結合が自明な解となり、未知事例に対して判定する根拠を持たない
- 解決策
  - 探索空間はリテラルのOR結合を許し、探索方法にバイアスをかけて候補を限定する
  - 見つかった候補が未知データに対しても信頼できるようなバイアスとは何か

それでは、特徴値のOR結合を仮説表現として認めればどうなるでしょう。

これは論理式の形式に関するバイアスをなくしてしまうことになり、仮説空間は原理的にすべての特徴値がとりうる組合せを要素としたべき集合となります。

バイアスをなくすることによる弊害は、候補数の多さだけには止まりません。

正例のOR結合、およびそれに対して負例に含まれない事例をORで結合したものなど、自明な解が多数存在します。

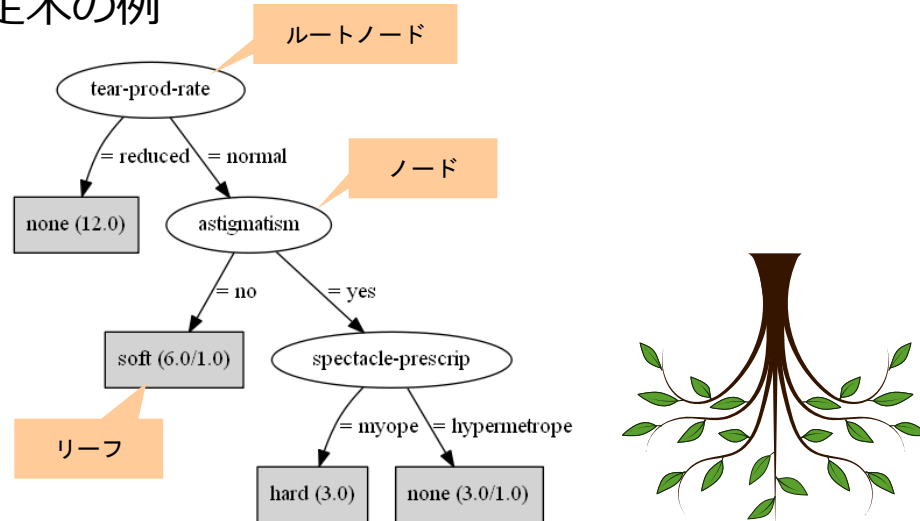
これらは未知事例に対して判定する根拠を持ちません。

そこで解決策として、探索空間はリテラルのOR結合を許し、探索方法にバイアスをかけて候補を限定します。

そして、見つかった候補が未知データに対しても信頼できるようなバイアスとは何かを考えてゆきます。

## 3.3 決定木の学習

### ・ 決定木の例



決定木とは、データを分類する質問をノード（節）とし、分類結果をリーフ（葉）とする木構造の概念表現です。

ルート（根）から、分類結果が正であるリーフに至るノードの分岐の値を AND 条件で結合し、すべての正のリーフに関して、そのようにして得られた論理式を OR 条件で結合することで、木構造と等価な論理式を得ることができます。

すなわち、論理式のOR結合を許した概念表現ということになります。

この例では特徴 tear-prod-rate（涙量）が最初の質問で、この値が reduced（減少）であると、コンタクトレンズは勧められない、という結論になります。

一方、この値が normal（正常）であれば、次の特徴 astigmatism（乱視）を調べる、という手順になります。



## 3.3.1 決定木とは

### ・決定木学習の考え方

- ・ ノードはデータを分割する特徴を表す
  - 分割後のデータができるだけ同一クラスが偏るように特徴を選択する
- ・ 特徴の値に基づいて学習データを分割する
- ・ 分割後のデータ集合に対して、同様の操作を行う
- ・ すべてのリーフが単一クラスの集合になれば終了

決定木学習の手順を説明します。

まず、学習データ全体を、その値によって分割する特徴を選びます。

選択の基準は、分割後のデータができるだけ同一クラスが偏るようになるものとします。

そして、特徴の値に基づいて学習データを分割します。

さらに分割後のデータ集合に対して、同様の操作を再帰的行います。

すべてのリーフが単一クラスの集合になれば終了です。

## 3.3.2 ID3アルゴリズム

---

**Algorithm 3.1** ID3 アルゴリズム

---

入力: 正解付き学習データ  $D$ , クラス特徴  $y$ , 特徴集合  $A$

出力: 決定木  $T$

```
root ノードを作成
if  $D$  が全て正例 then
    return ラベル Yes
else if  $D$  が全て負例 then
    return ラベル No
else if 特徴集合  $A == \emptyset$  (空集合) then
    return  $D$  中の最頻値のクラス
else
```

このような決定木を作成するもっとも基本的な手順が、ID3アルゴリズムです。  
ここでは、2値分類問題で決定木の作成手順を説明します。  
最初のところは再帰呼び出しの終了条件です。

### 3.3.2 ID3アルゴリズム

```
 $a \leftarrow A$  中で最も分類能力の高い特徴  
root ノードの決定特徴  $\leftarrow a$   
for all  $a$  の取りうる値  $v$  do  
   $a = v$  に対応する枝を作成  
  データの中から値  $v$  を取る部分集合  $D_v$  を作成  
  if  $D_v == \emptyset$  then  
    return  $D$  中の最頻値のクラス  
  else  
    ID3(部分集合  $D_v$ , クラス特徴  $y$ , 特徴集合  $A - a$ )  
  end if  
end for  
end if  
return root ノード
```

---

ID3アルゴリズムでは、すべての特徴の中から、得られる情報がもっとも多そうなものを最初の質問として選びます。

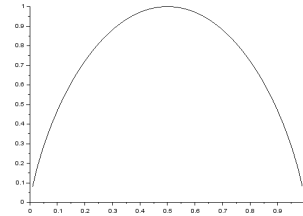
そして、その質問の答えによって、学習データがいくつかの部分集合に分割されます。

同じクラスのデータのみからなる部分集合は、もうそれ以上質問を続ける必要はありません。

一方、異なるクラスのデータが混在する部分集合は、得られる情報が多そうな質問を選んで、データをさらに分割します。

## 3.3.2 ID3アルゴリズム

- 分類能力の高い属性を決定する方法
  - その属性を使った分類を行うことによって、なるべくきれいにクラスが分かれるように
- エントロピー
  - データ集合 $D$ の乱雑さを表現
  - 正例の割合: $P_+$  , 負例の割合:  $P_-$
  - エントロピーの定義



$$E(D) = -P_+ \log_2 P_+ - P_- \log_2 P_-$$

分類能力が高いとは、「その特徴を使った分類を行うことによって、なるべくきれいに正例と負例に分かれる」ということです。

いいかえると、乱雑さが少なくなるように分類を行うということです。

乱雑さの尺度として、エントロピーを用いる場合の計算法を説明します。

データ集合 $D$ のエントロピーの値は $P_+ = 1$ または $P_- = 1$ のとき最小値0となり、 $P_+ = P_- = 0.5$ のとき、最大値1となります。

エントロピーの値が小さいほど、集合が乱雑でない、すなわち整っている（同じクラスのものが多い）ということになります。

## 3.3.2 ID3アルゴリズム

- 情報獲得量
  - 特徴aを用いた分割後のエントロピーの減少量
  - 特徴aで値vを取るデータの集合:  $D_v$
  - $D_v$ の要素数:  $|D_v|$
  - 情報獲得量の定義

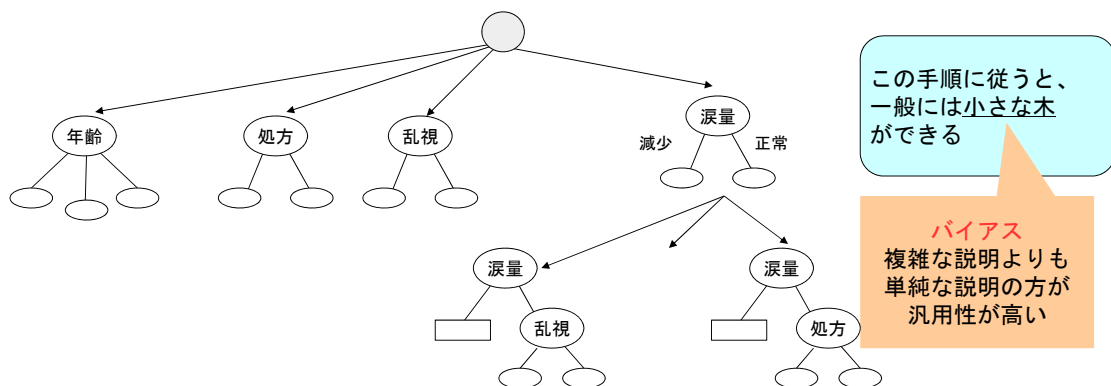
$$\text{Gain}(D, a) \equiv E(D) - \sum_{v \in \text{Values}(a)} \frac{|D_v|}{|D|} E(D_v)$$

データの分割によるエントロピーの減少量を情報獲得量と定義します。  
E(D)が分割前のエントロピーです。  
データDをある特徴aの値に基づいて分割したとします。  
分類後のエントロピーは、集合の要素数の割合で重みを付けて計算します。

## 3.3.2 ID3アルゴリズム

### ・ ID3アルゴリズムのバイアス

- ・ 分類能力の高いノードをなるべく根の近くにもつ
- ・ 欲張り法で探索を行い、すべてのリーフが単一クラスの集合になれば終了



ID3 アルゴリズムは、探索において「分類能力の高いノードをなるべく根の近くにもち、その結果として得られるなるべく小さな木を結果とする」というバイアスを与えます。また、探索空間が大きく全探索が難しいので、分岐ごとにより結果を積み重ねてゆく欲張り法で目標概念を探します。ただし、この方法で作成された決定木は、最小であることが保証されないで、このことが解の不安定性をもたらしていることになります。

## 3.3.2 ID3アルゴリズム

- なぜ小さな木の方がよいか

- オッカムの剃刀

「データに適合する最も単純な仮説を選べ」

- 複雑な仮説

→表現能力が高い

→偶然にデータを説明できるかもしれない

- 単純な仮説

→表現能力が低い

→偶然にデータを説明できる確率は低い

→でも説明できた！

→**必然**

ID3 アルゴリズムのバイアスは単純に表現すると、「小さい木を好む」となります。  
なぜ小さな木を結果とするのでしょうか。

これはオッカムの剃刀とよばれる「データに適合するもっとも単純な仮説を選べ」という原理に基づいています。

複雑な仮説なら表現能力が高いので、偶然に学習データを説明できるかもしれません。

一方、単純な仮説の場合は表現能力が低いので、偶然にデータを説明できる確率は低くなります。

もし、ID3 アルゴリズムによって、小さな木で学習データが説明できたとすると、これは偶然である確率は相当低くなります。

すなわち偶然でなければ必然である、というわけです。

### 3.3.3 過学習を避ける

- 決定木学習における過学習の避け方
  - 学習過程で木の成長を止める
    - リーフに所属することができる最小データ数を多くする
    - 木の段階の最大数を決めておく
  - 十分に成長させた後に枝刈り
    1. 学習用データを用いてできるだけ成長した木を作成する
    2. 各枝について検証用データを用いて分類能力を測定し、不要な枝を刈り取る

ここで、ID3 アルゴリズムにおける過学習について考えてみます。

過学習とは、文字通りの意味は学習しすぎることです。

機械学習においては、モデルが学習データに特化しすぎたために、未知データに対して性能が下がる現象を指します。

小さな木として学習を終わらせることは、一般的には難しいことになります。

正解率を上げるために、最後の1例までエラーがなくなるように決定木を作成してしまうと、その決定木は成長しすぎて、学習データに適応しすぎた過学習になりがちです。

そこで、過学習への対処として、適当なところで決定木の成長を止める方法や、完全に学習させたあと、枝刈りするという方法があります。



### 3.3.3 過学習を避ける

---

**Algorithm 3.2** 決定木の枝刈りアルゴリズム

---

入力: 学習済みの決定木  $T$ , 検証用データ  $D'$

出力: 枝刈り後の決定木  $T'$

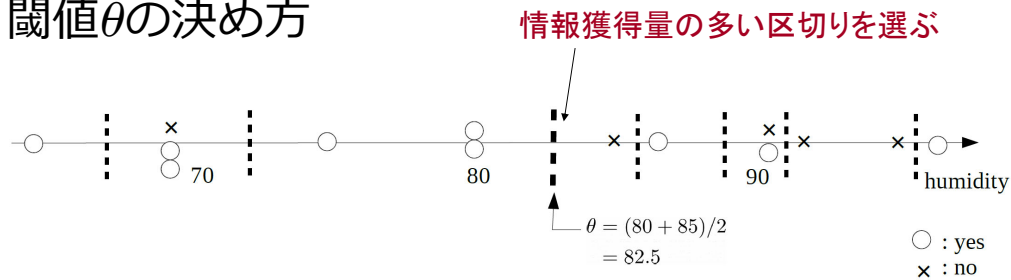
```
for all  $T$  のノード  $N$ , ルートから遠いものから順に do
     $T_N \leftarrow N$  をルートとする木
     $D_N \leftarrow D'$  の中で,  $T_N$  によってカバーされるデータ
    if  $\text{accuracy}(T_N, D_N) < \text{majority}(D_N)$  then
         $T_N$  を, リーフ  $\text{majority\_class}(D_N)$  に置き換え
    end if
end for
return 枝刈り後の決定木  $T'$ 
```

---

枝刈りではまず, 学習用データを用いてできるだけ成長した木を作成します.  
次に, 検証用データを用いて、あるノード以降の分類性能が、そのノードに属するデータの多数決を用いた分類よりも劣れば、その多数クラスをラベルとするリーフに置き換えます.

## 3.4 数値特徴に対する決定木

- ノードにおけるデータ分割
  - カテゴリ特徴：値で分割
  - 数値特徴：閾値との比較で分割
- 閾値 $\theta$ の決め方



$$\begin{aligned}\text{Gain}(D, \text{humidity}_{82.5}) &= 0.94 - \frac{7}{14} \text{Entropy}(D, < 82.5) - \frac{7}{14} \text{Entropy}(D, \geq 82.5) \\ &= 0.152\end{aligned}$$

ここでは特徴を数値データとしたときの決定木の作成について考えます。  
できるだけきれいにデータを分割する特徴を木の上位に配置するという基本的なアイディアは変わりません。  
カテゴリ特徴の場合は、値でデータを分割しましたが、数値特徴の場合は、設定した閾値との比較でデータを分割します。

## 3.4 数値特徴に対する決定木

- CART (Classification and Regression Trees)
  - 必ず2つの子ノードを持つ2分木構造
  - 数値特徴はノードとして何度でも出現可能
  - 識別と回帰のいずれにも使える
  - scikit-learnでの識別のための決定木の実装は `DecisionTreeClassifier` クラス

scikit-learnでの決定木の実装はCARTとよばれるアルゴリズムを採用しています。CARTは必ず2つの子ノードを持つ2分木に構造が限定されています。ここで、数値特徴はノードとして何度でも出現可能です。CARTは識別と回帰のいずれにも使えます。

## 3.4 数値特徴に対する決定木

- DecisionTreeClassifierのパラメータ
  - criterion : 情報獲得量の計算法
    - ジニ不純度  $\text{GiniImpurity}(D) \equiv 2 \cdot P_+ \cdot P_-$
    - エントロピー  $E(D) = -P_+ \log_2 P_+ - P_- \log_2 P_-$
    - ジニ不純度のほうが最頻クラスの分離に少し偏る
  - max\_depth : 木の最大の深さ
  - min\_samples\_split : ノードが持つ事例数の最小値

scikit-learnのDecisionTreeClassifierはインスタンス作成時にいくつかのパラメータを設定できます。

情報獲得量の計算法については、デフォルトがジニ不純度で、エントロピーを指定することもできます。

これらはたいていの場合違いはありませんが、ジニ不純度のほうが最頻クラスの分離に少し偏るといわれています。

max\_depthは木の最大の深さを指定し、min\_samples\_splitはノードが持つ事例数の最小値を指定します。

いずれも過学習を防ぐためのパラメータです。