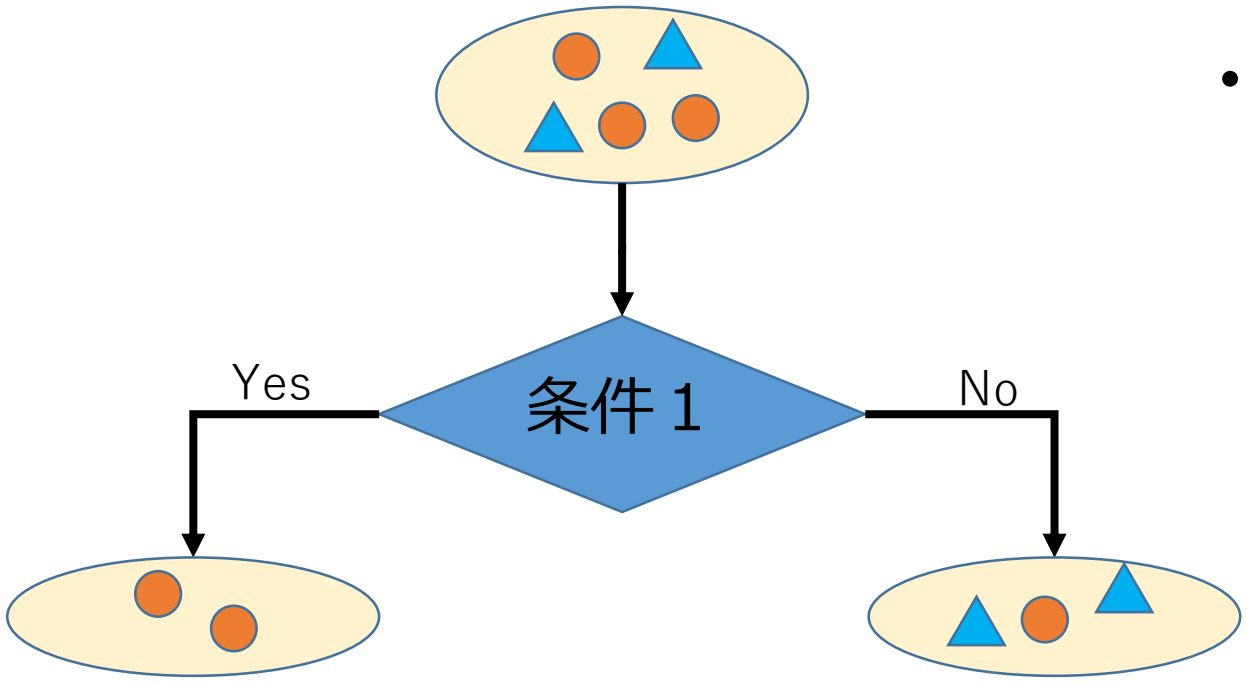
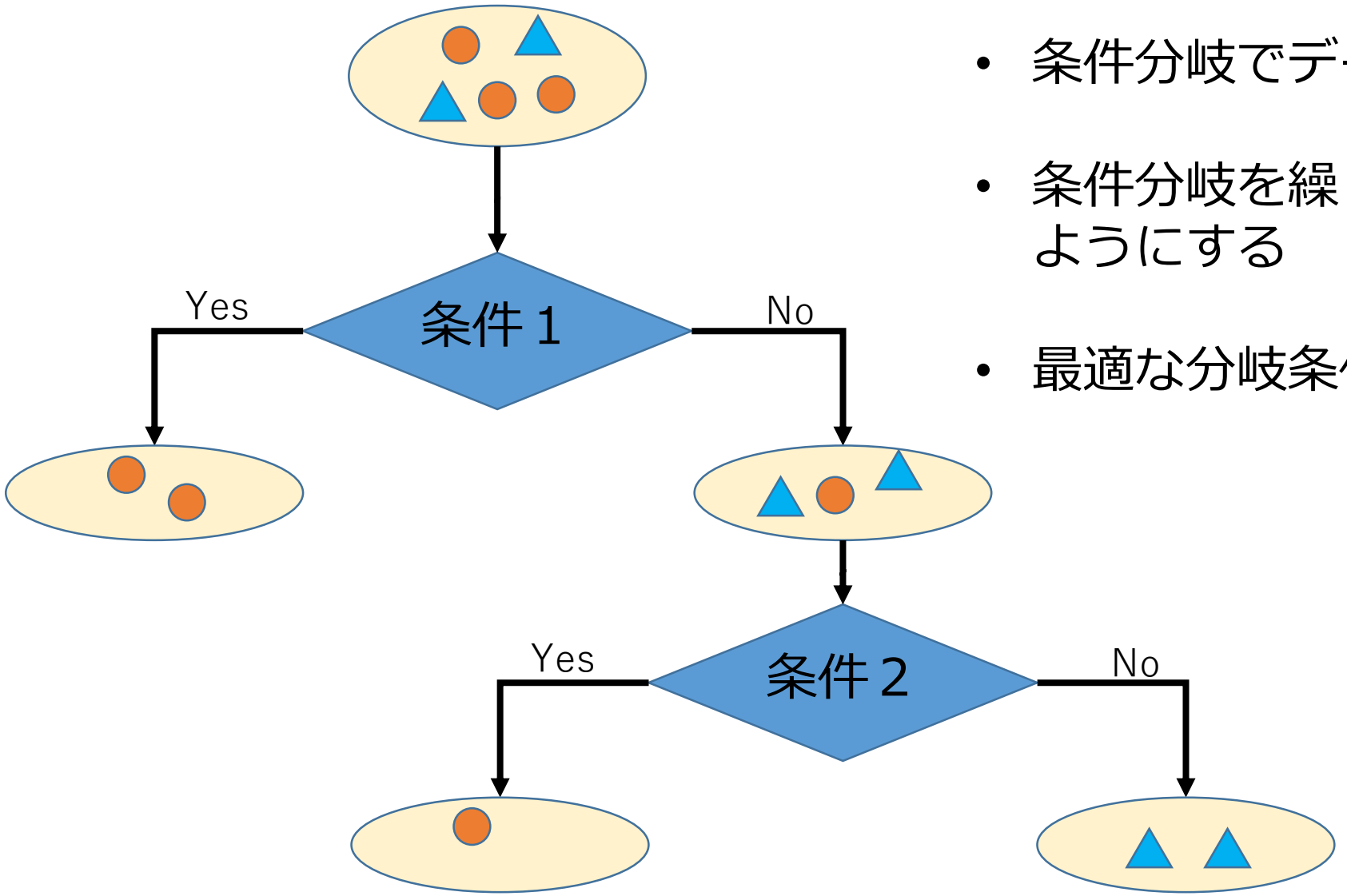
決定木（分類）

決定木モデル



- 条件分岐でデータを分類する

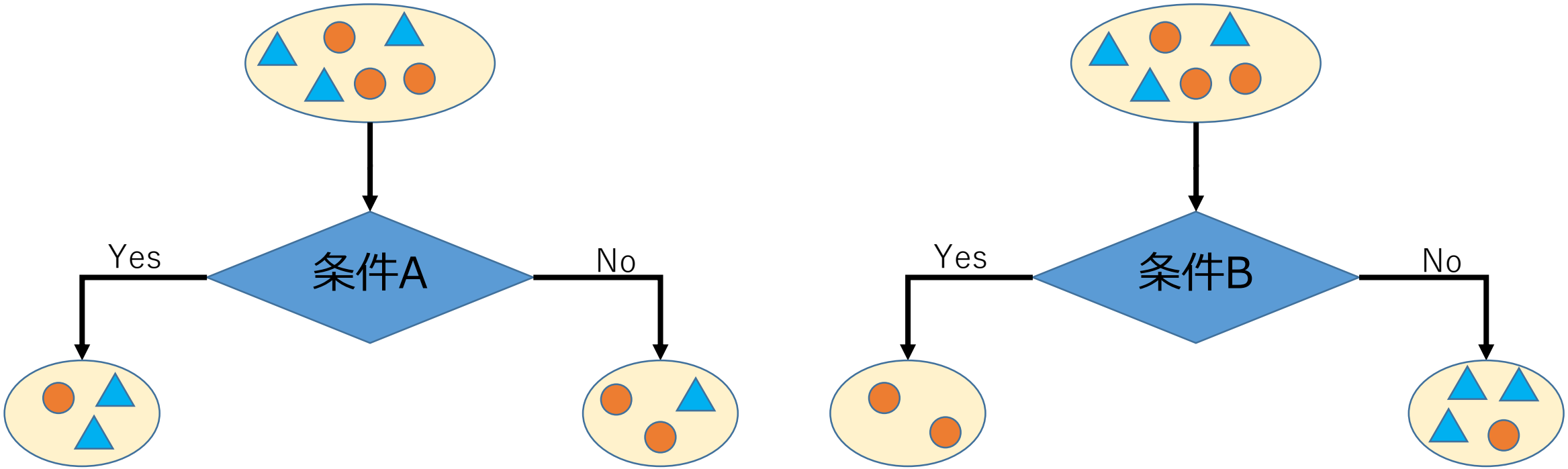
決定木モデル



- 条件分岐でデータを分類する
- 条件分岐を繰り返し、正しく分類できるようにする
- 最適な分岐条件を機械学習で求める

決定木モデル

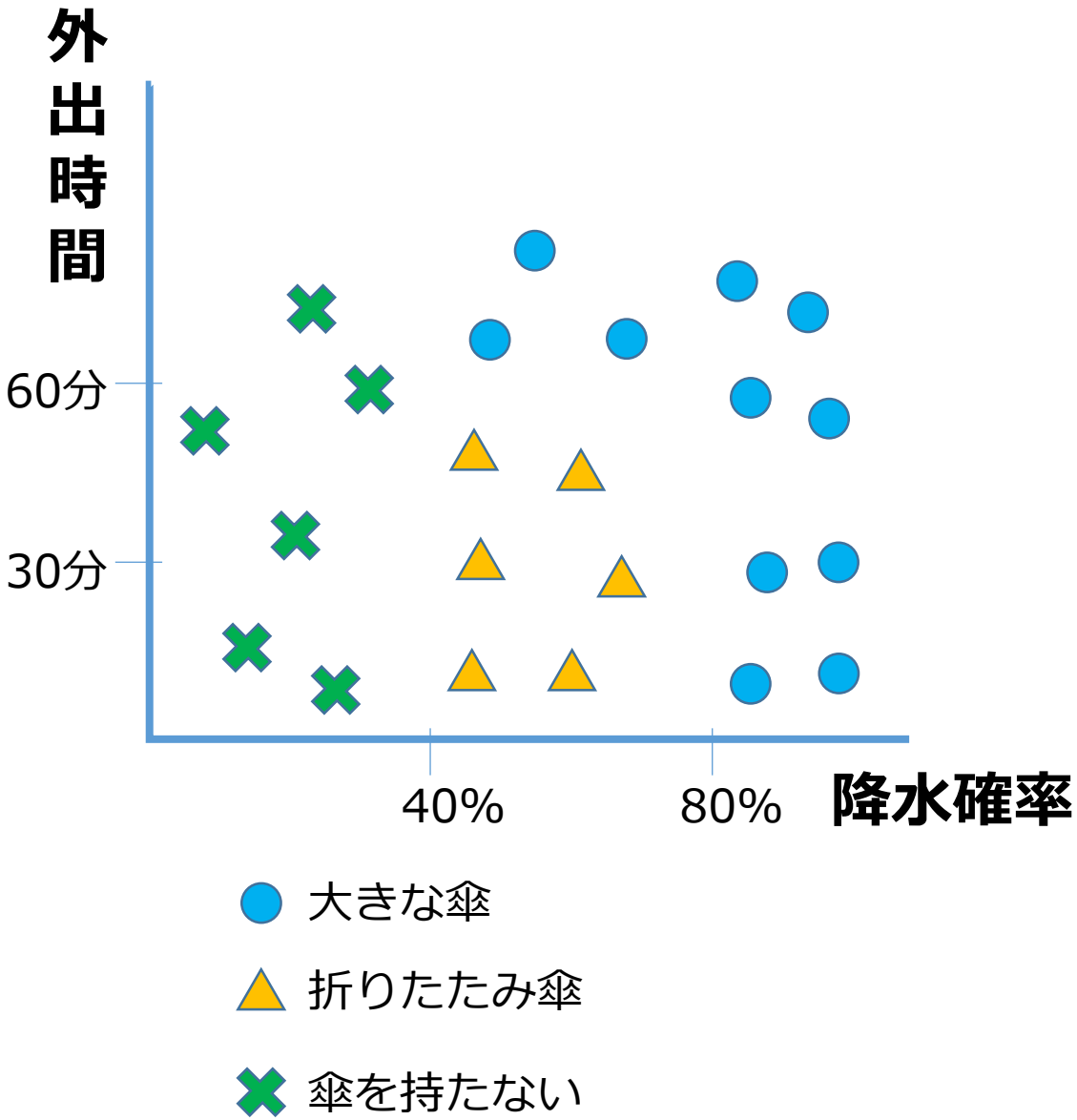
P157、158

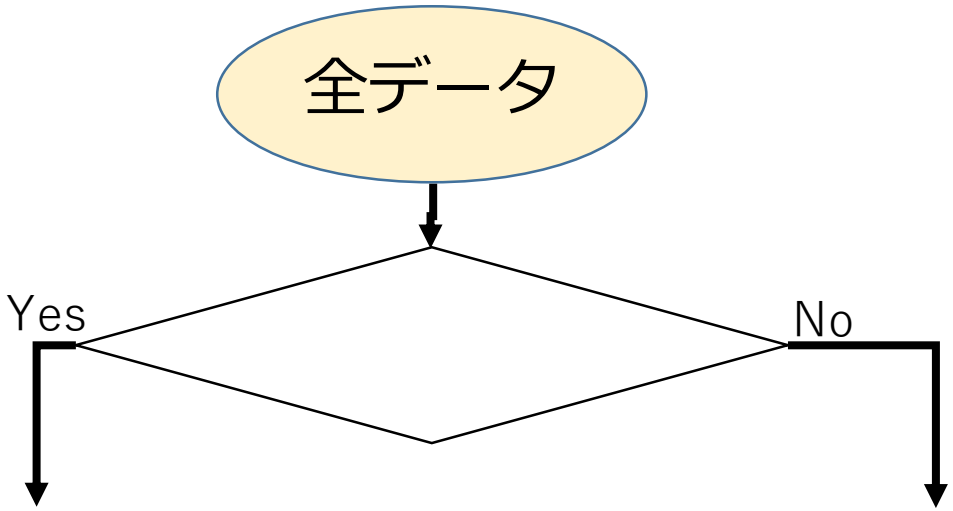


分類したデータの**混ざり具合**が最も小さくなる分岐条件を探す

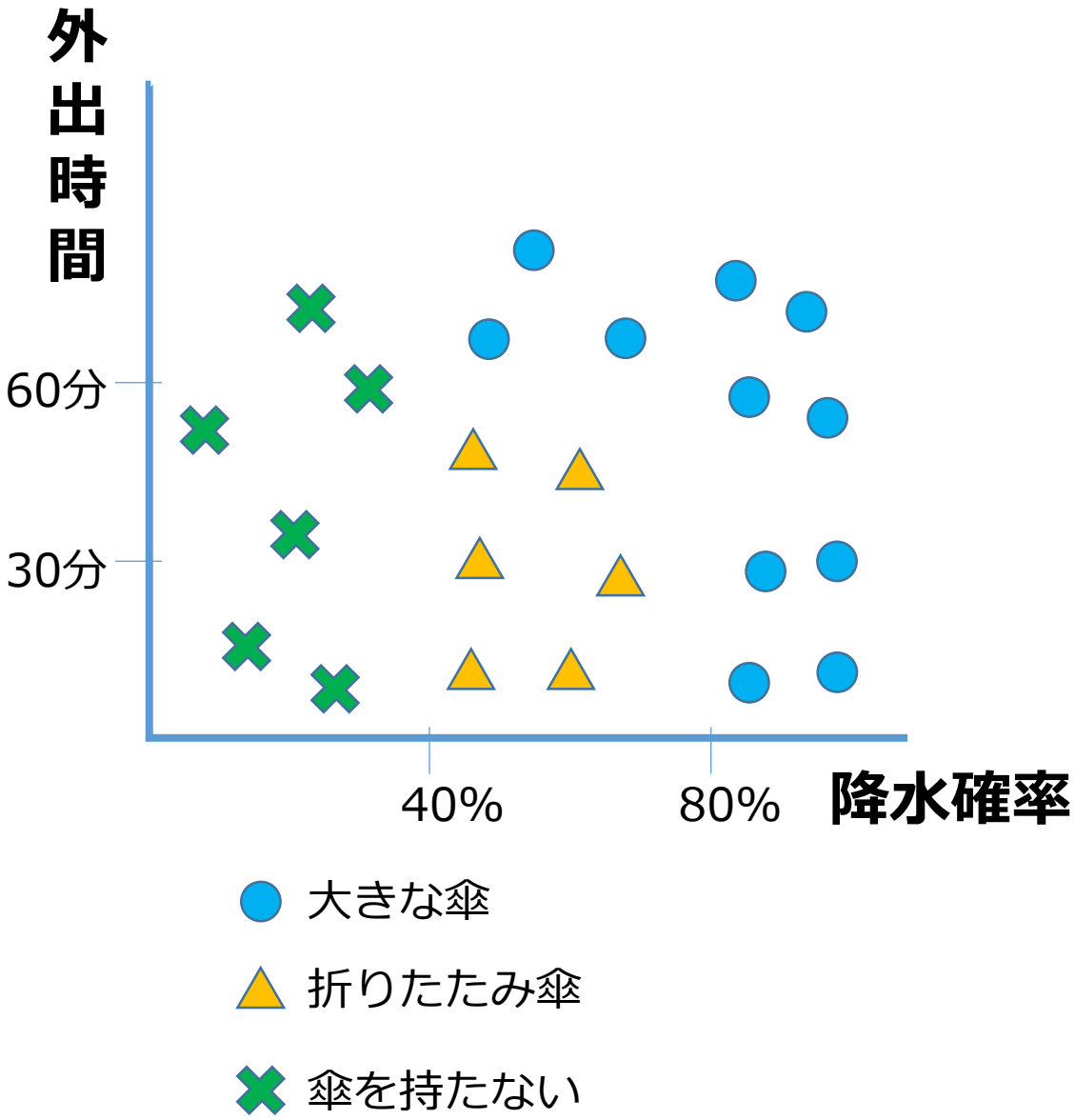
不純度

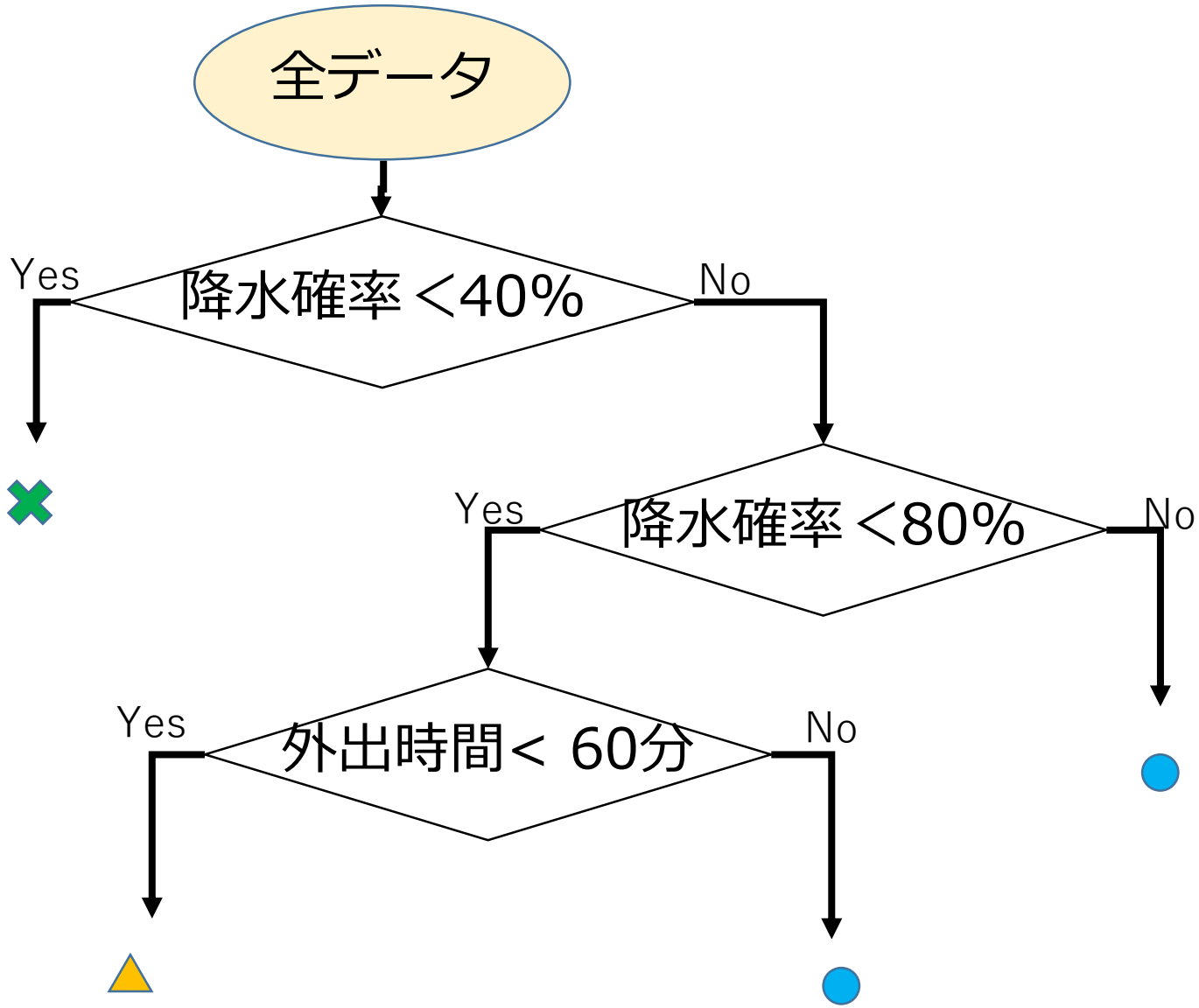
例）外出時に持って行きそうな傘は？





外出時に持って行くと思われる傘は？





分岐条件の確認

コード5-24 分岐条件の列を決める

```
model.tree_.feature
array([ 3, -2,  3, -2, -2], dtype=int64)
```

各ノードの分岐条件に使われている特徴量の列番号

コード5-25 分岐条件のしきい値を含む配列を返すtree_.threshold

```
model.tree_.threshold
array([ 0.275, -2.   ,  0.69 , -2.   , -2.   ])
```

分岐条件の閾値

特徴量xの列番号

0	1	2	3
がく片長さ	がく片幅	花弁長さ	花弁幅

分岐条件の確認

コード5-24 分岐条件の列を決める

model.tree_.feature

```
array([ 3, -2,  3, -2, -2], dtype=int64)
```

最初は　条件「花弁幅 ≦ 0.275」

コード5-25 分岐条件のしきい値を含む配列を返すtree_.threshold

model.tree_.threshold

```
array([ 0.275, -2.   ,  0.69 , -2.   , -2.   ])
```

特徴量xの列番号

0	1	2	3
がく片長さ	がく片幅	花弁長さ	花弁幅

分岐条件の確認

コード5-24 分岐条件の列を決める

```
model.tree_.feature
array([ 3, -2,  3, -2, -2], dtype=int64)
```

次は　分類結果

コード5-25 分岐条件のしきい値を含む配列を返すtree_.threshold

```
model.tree_.threshold
array([ 0.275, -2.   ,  0.69 , -2.   , -2.   ])
```

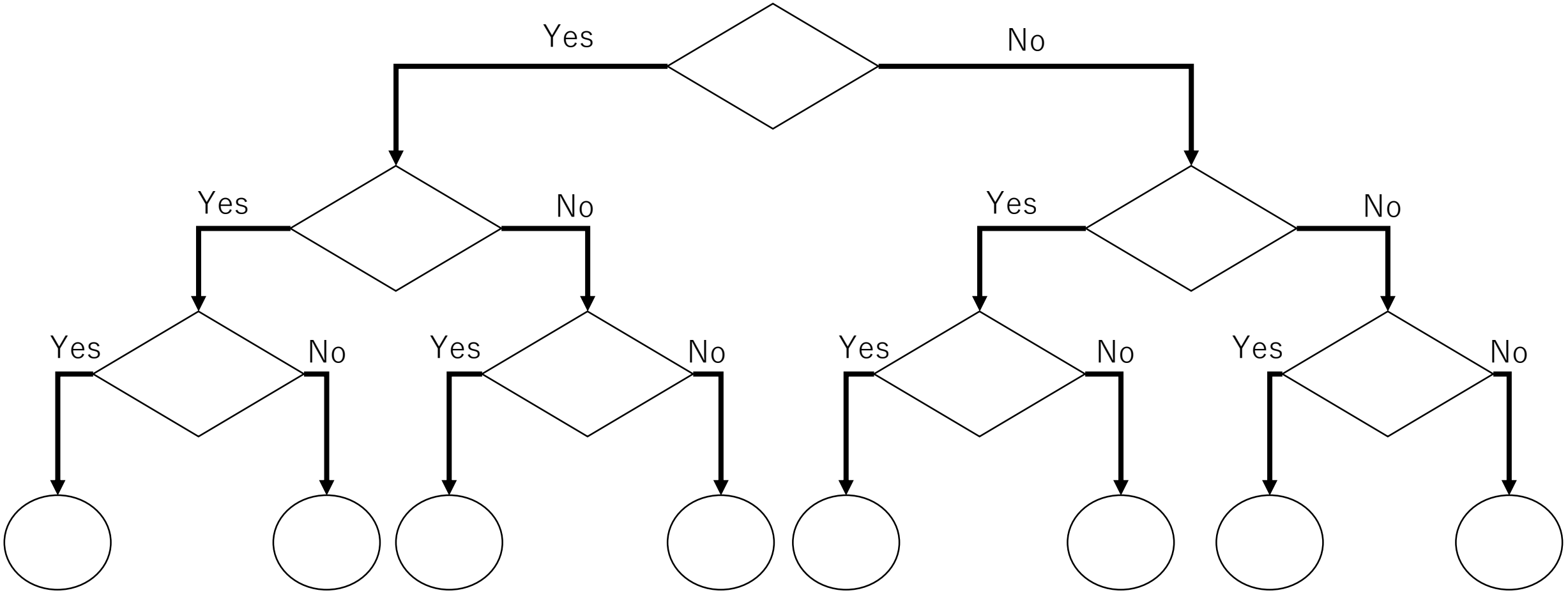
特徴量xの列番号

0	1	2	3
がく片長さ	がく片幅	花弁長さ	花弁幅

決定木の図示

P171、172

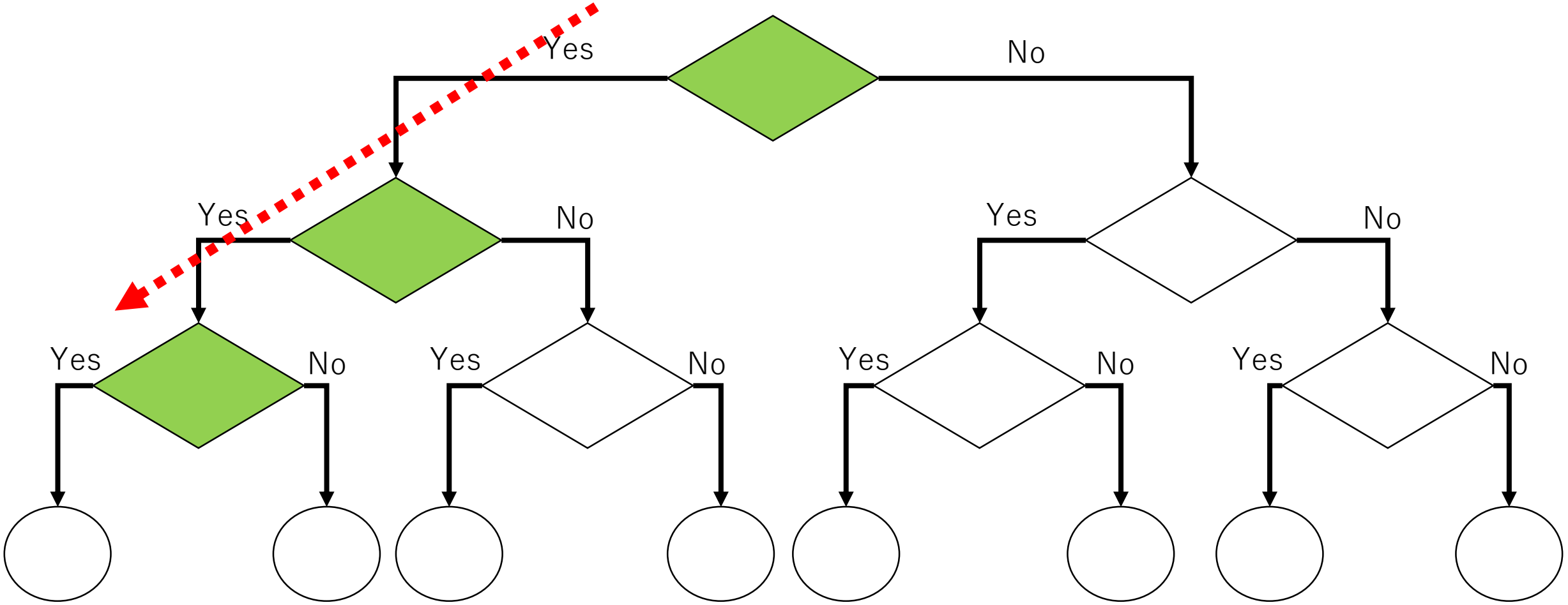
①深さ分の木を書く
例）深さ3の場合



決定木の図示

P171、172

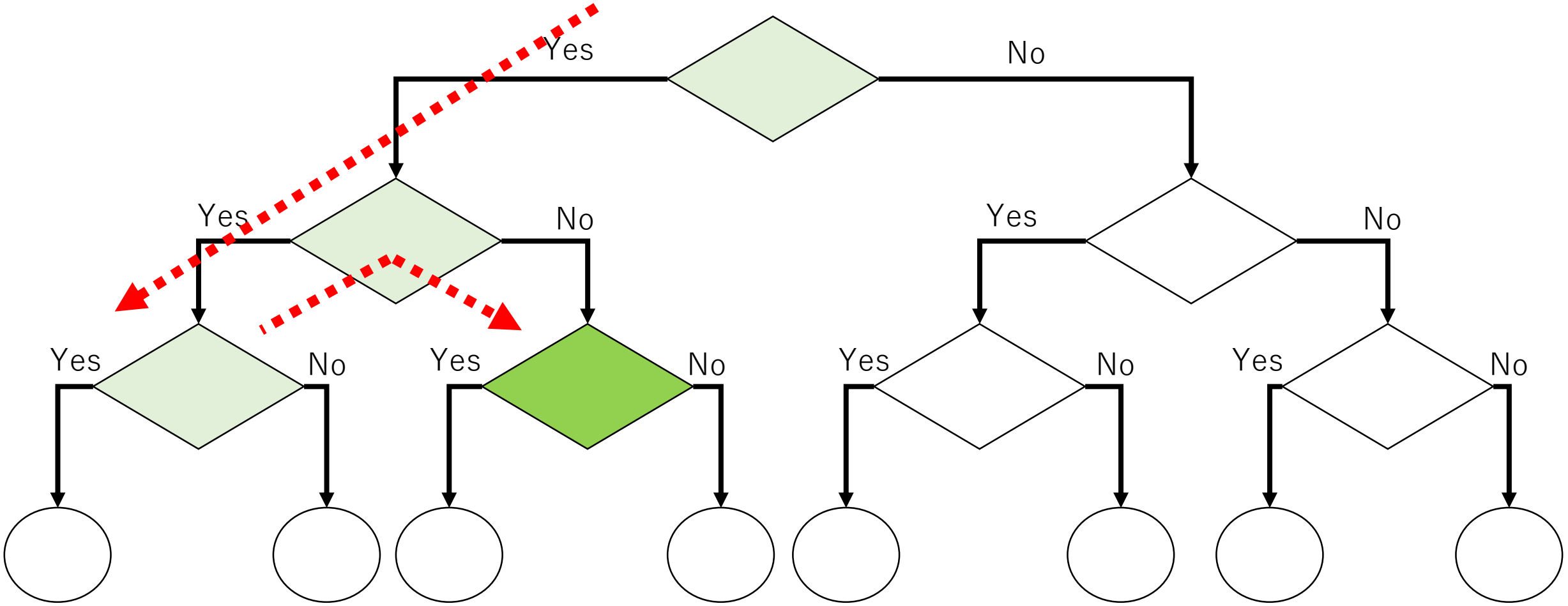
②分岐条件の列を順に当てはめていく



決定木の図示

P171、172

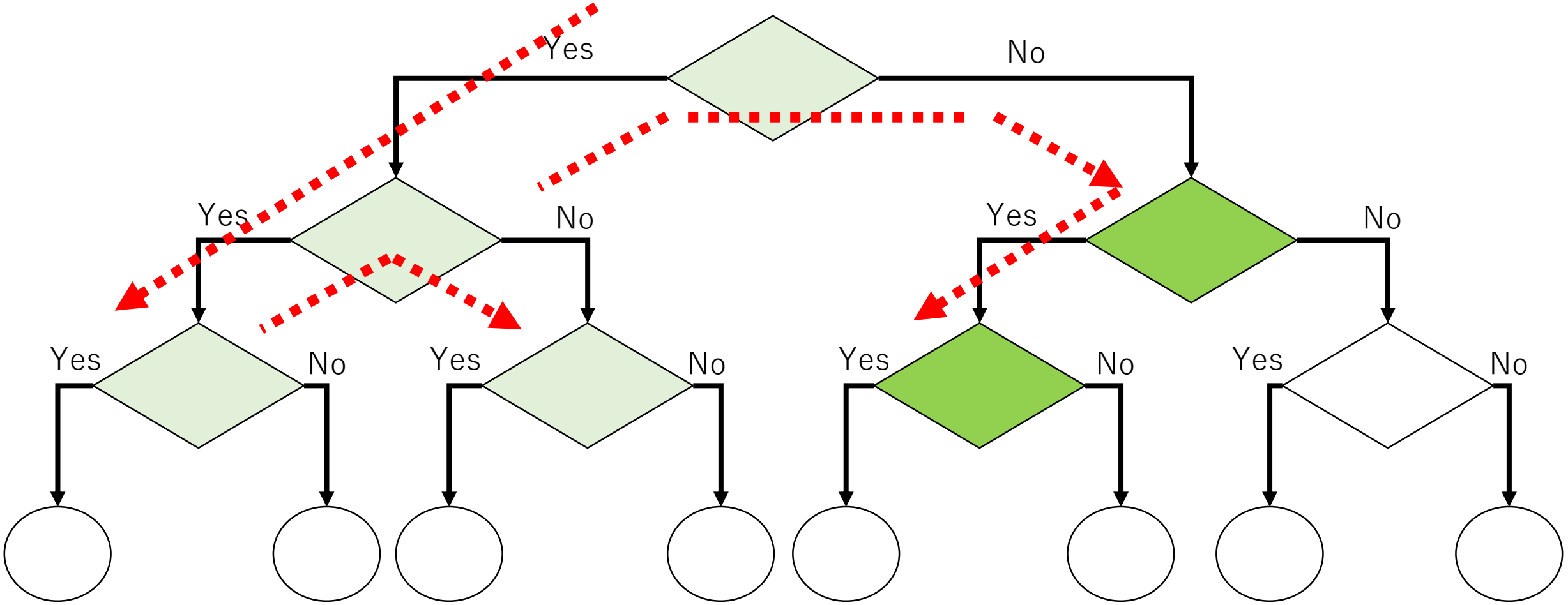
②分岐条件の列を順に当てはめていく



決定木の図示

P171、172

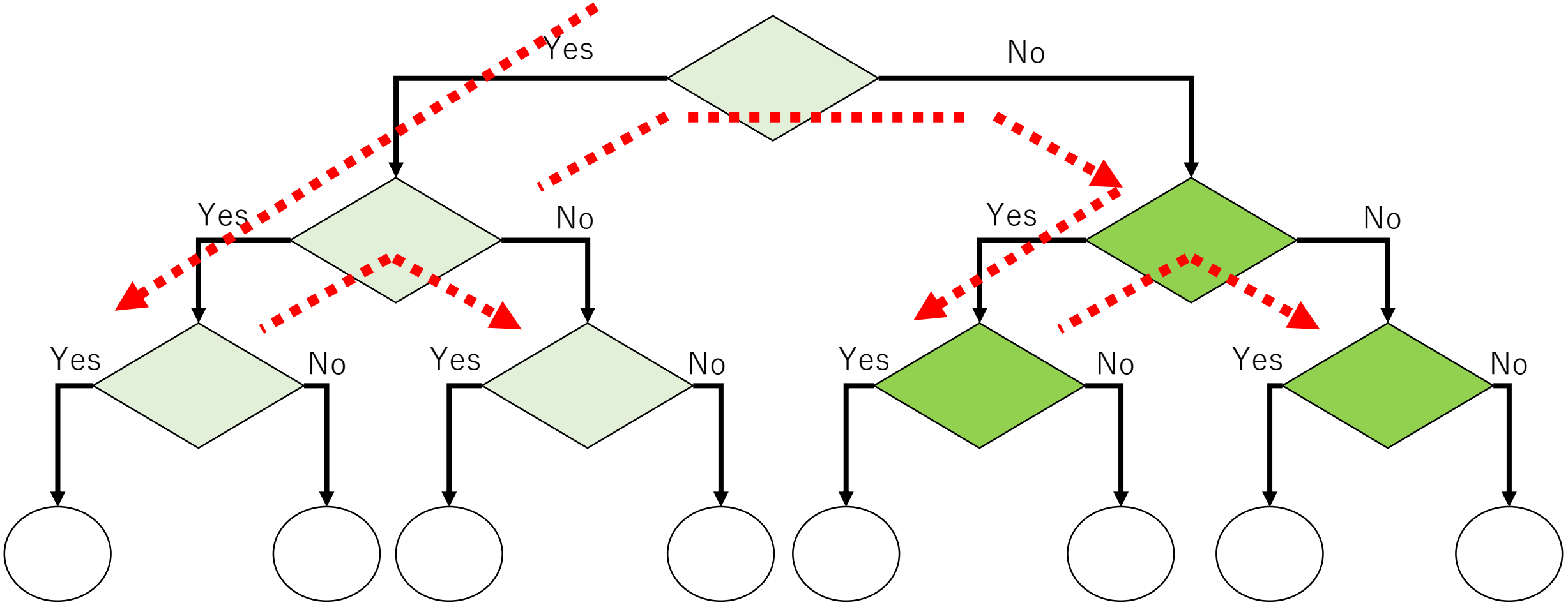
②分岐条件の列を順に当てはめていく



決定木の図示

P171、172

②分岐条件の列を順に当てはめていく

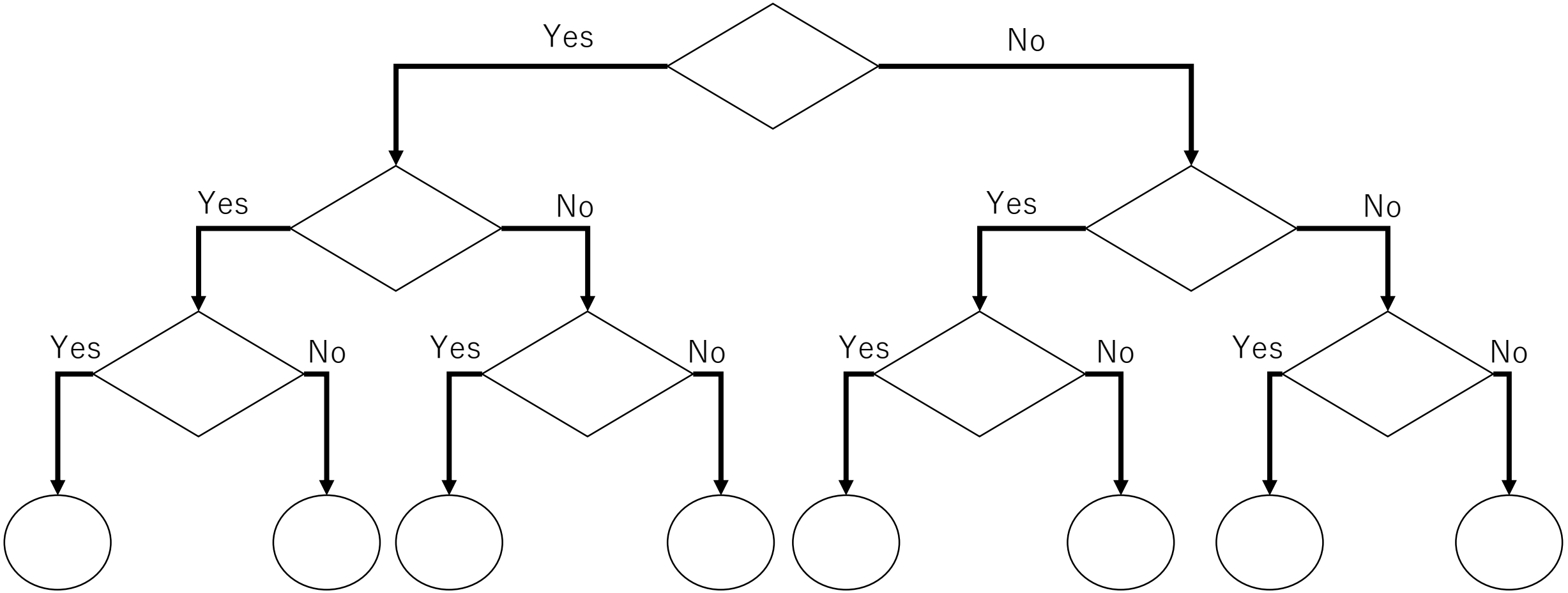


決定木の図示

P171、172

②分岐条件の列を順に当てはめていく

例）model.tree_.feature
[3, 2, 0, -2, -2, -2, 1,-2, -2]

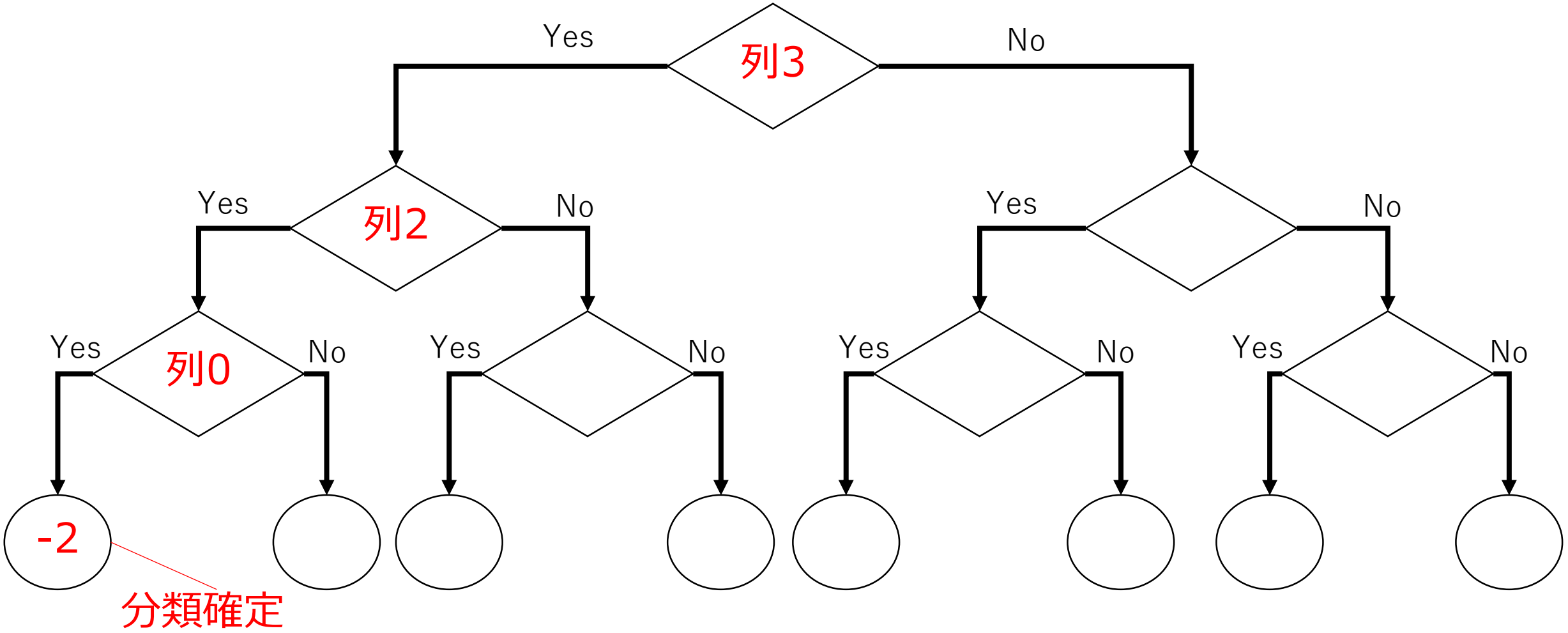


決定木の図示

P171、172

②分岐条件の列を順に当てはめていく

例） model.tree_.feature
[**3, 2, 0, -2**, -2, -2, 1,-2, -2]

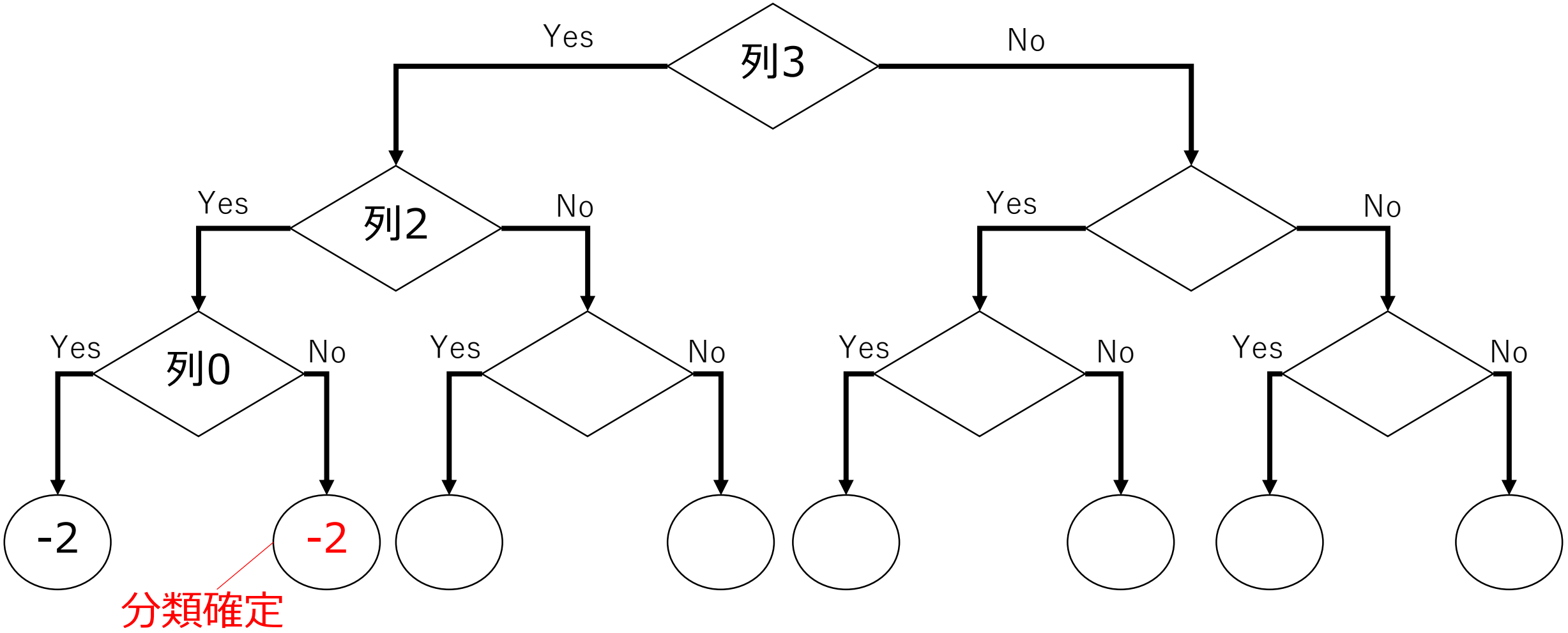


決定木の図示

P171、172

②分岐条件の列を順に当てはめていく

例） model.tree_.feature
[3, 2, 0, -2, **-2**, -2, 1,-2, -2]

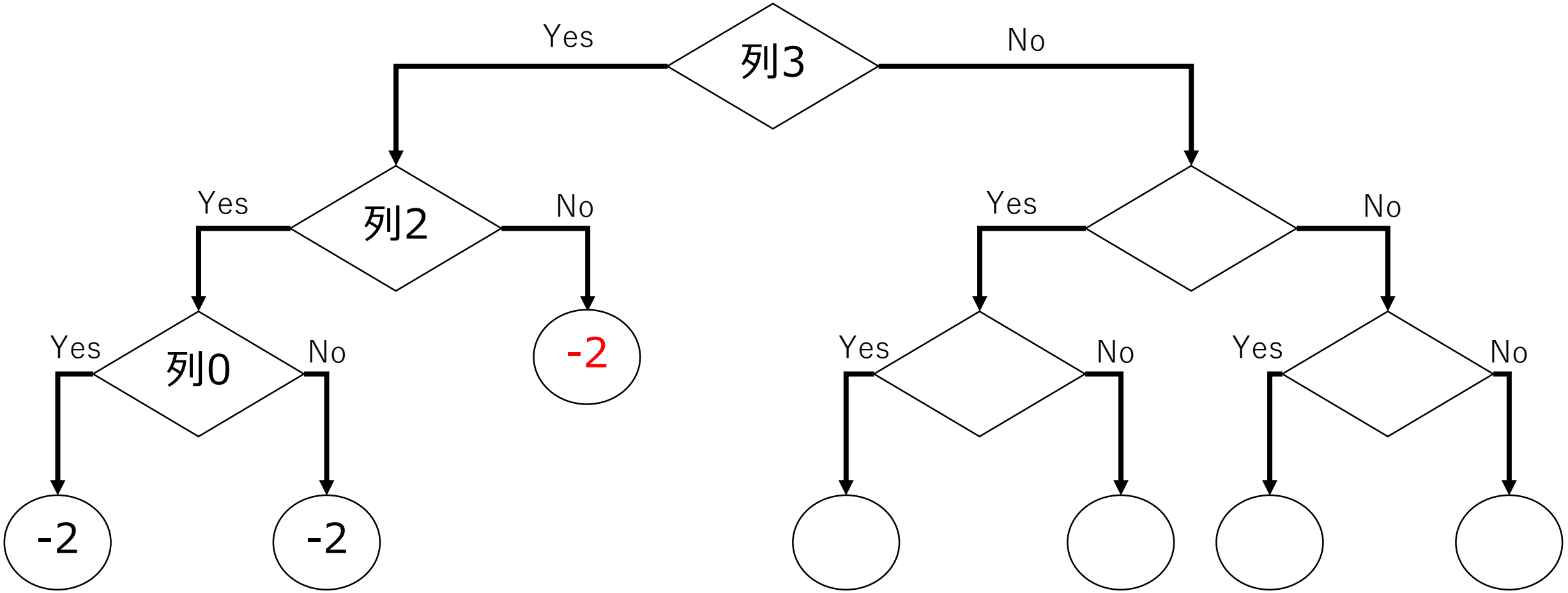


決定木の図示

P171、172

②分岐条件の列を順に当てはめていく

例） model.tree_.feature
[3, 2, 0, -2, -2, **-2**, 1,-2, -2]

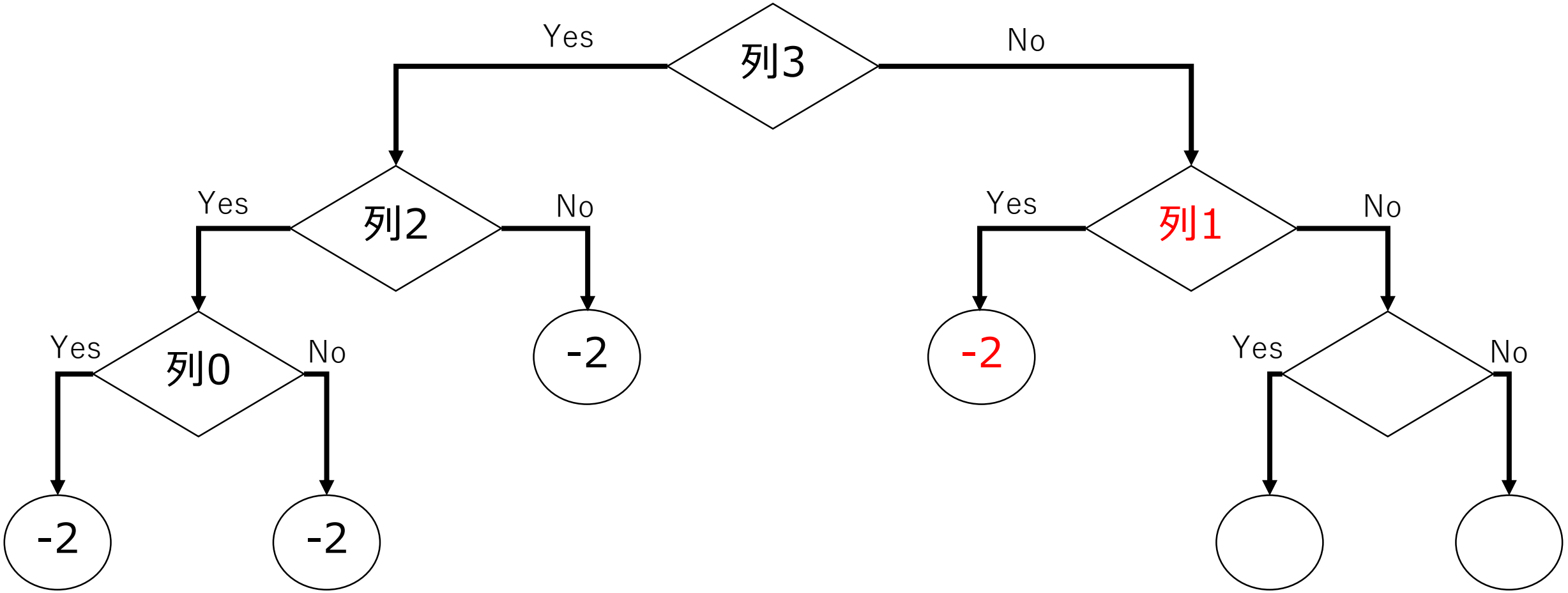


決定木の図示

P171、172

②分岐条件の列を順に当てはめていく

例） model.tree_.feature
[3, 2, 0, -2, -2, -2, 1,-2, -2]

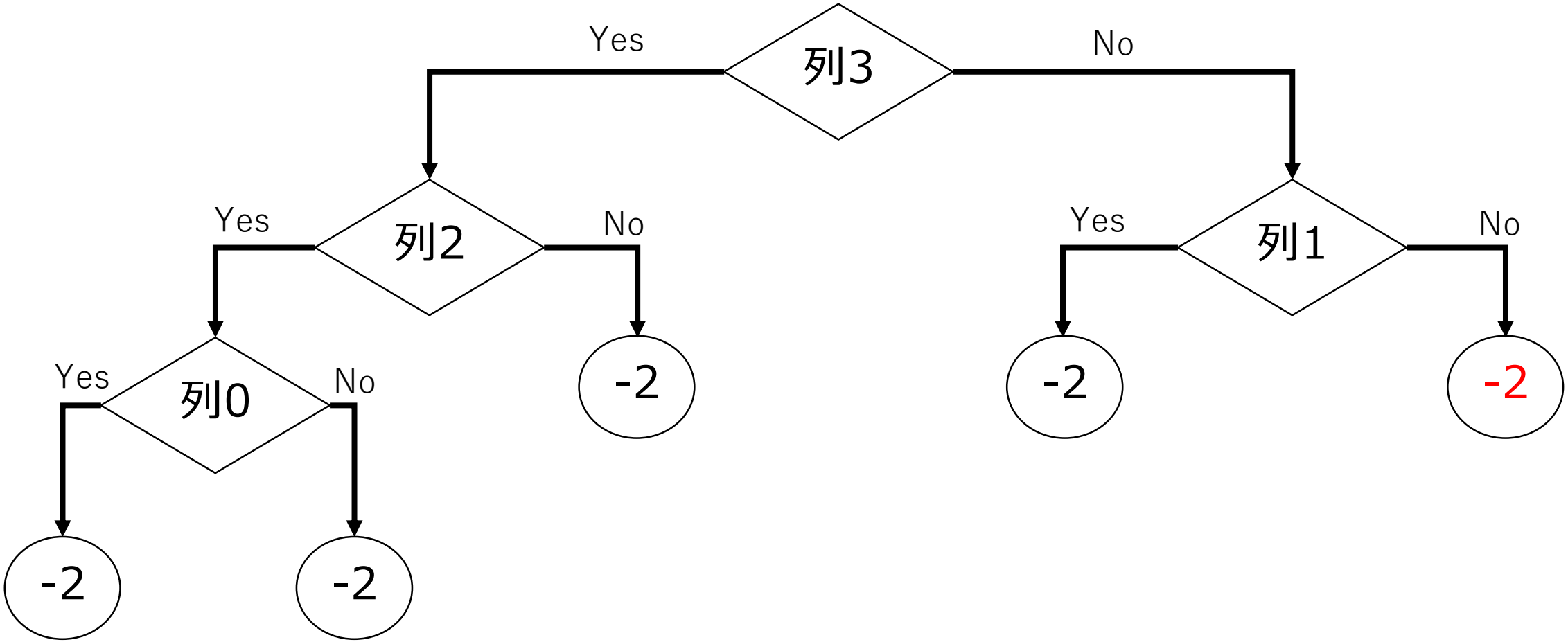


決定木の図示

P171、172

②分岐条件の列を順に当てはめていく

例） model.tree_.feature
[3, 2, 0, -2, -2, -2, 1,-2, -2]

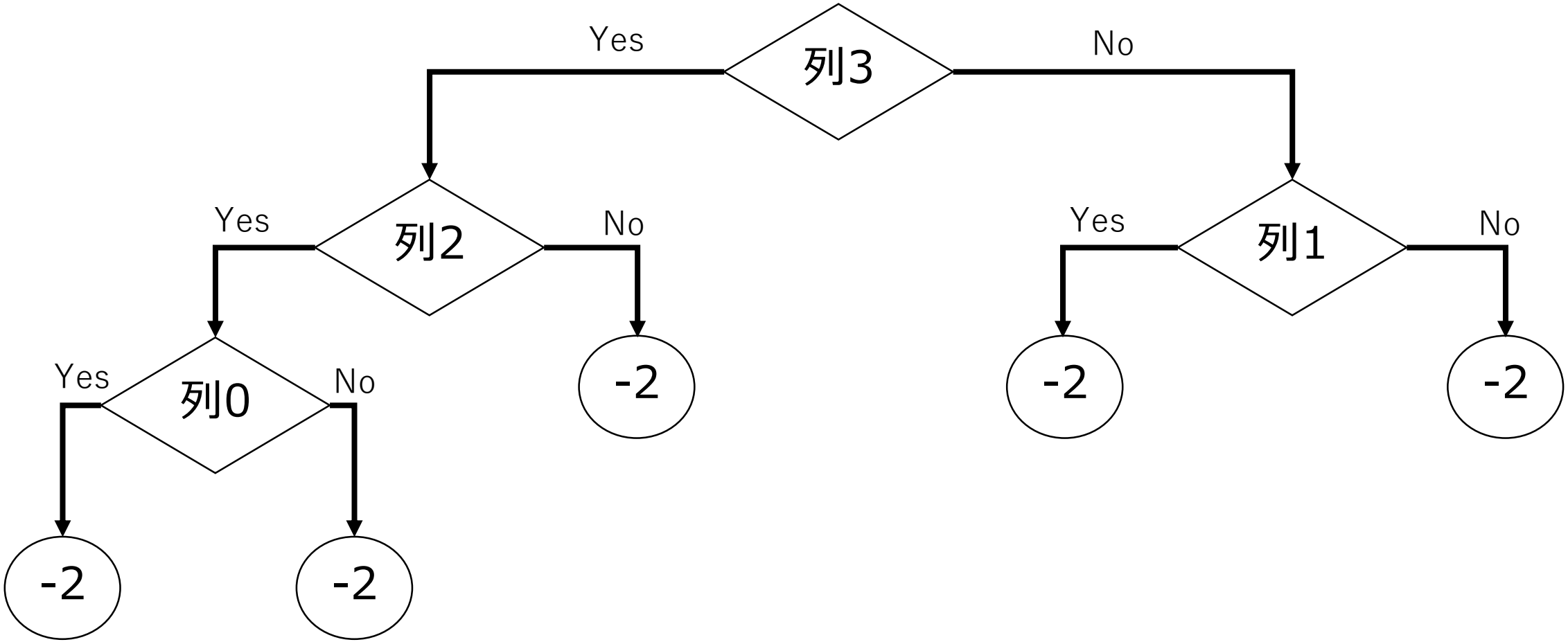


決定木の図示

P171、172

③閾値を当てはめる

例） model.tree_.threshold
[0.2, 0.1, 0.5, -2. , -2. , -2. , 0.3, -2. , -2.]

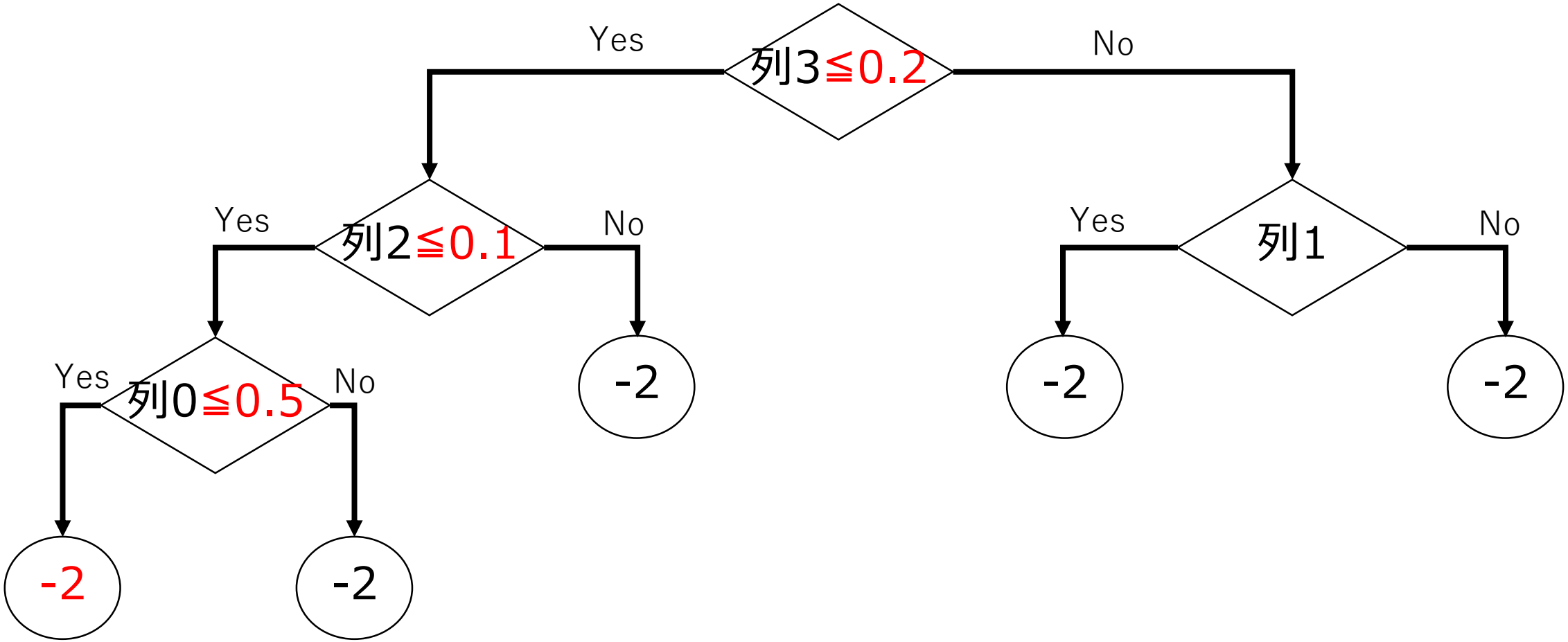


決定木の図示

P171、172

③閾値を当てはめる

例） model.tree_.threshold
[0.2, 0.1, 0.5, -2. , -2. , -2. , 0.3, -2. , -2.]

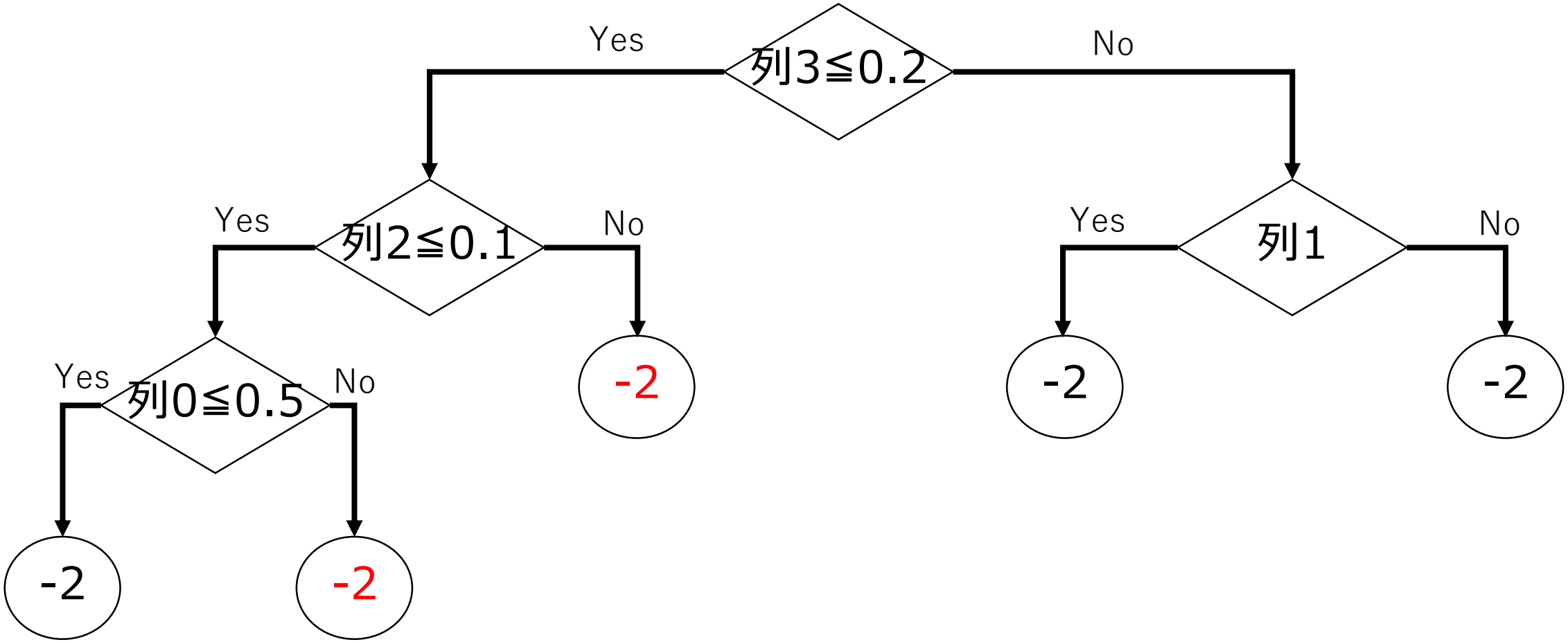


決定木の図示

P171、172

③閾値を当てはめる

例） model.tree_.threshold
[0.2, 0.1, 0.5, -2. , -2. , -2. , 0.3, -2. , -2.]

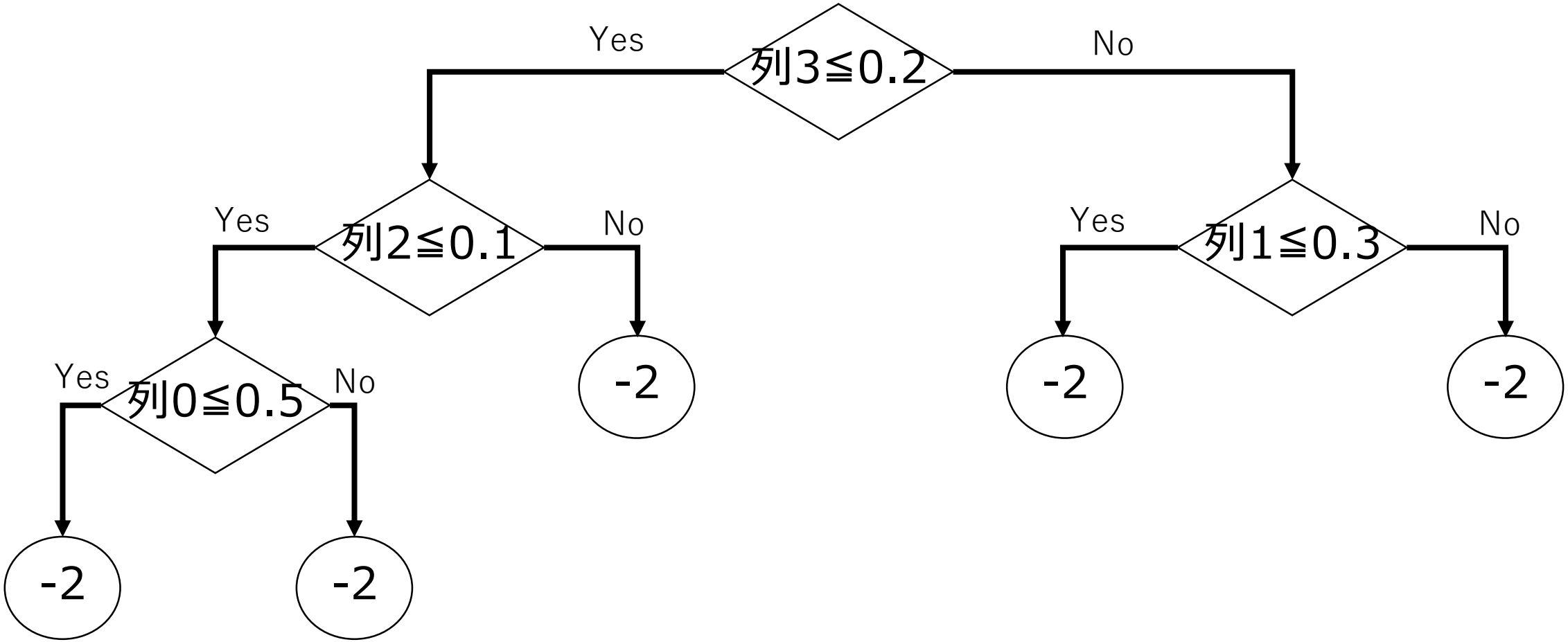


決定木の図示

P171、172

③閾値を当てはめる

例） model.tree_.threshold
[0.2, 0.1, 0.5, -2. , -2. , -2. , 0.3, -2. , -2.]



決定木のグループと分類の対応

P173、174

コード5-26
リーフに到達したデータの数を返す

```
print(model.tree_.value[1])
print(model.tree_.value[3])
print(model.tree_.value[4])
```

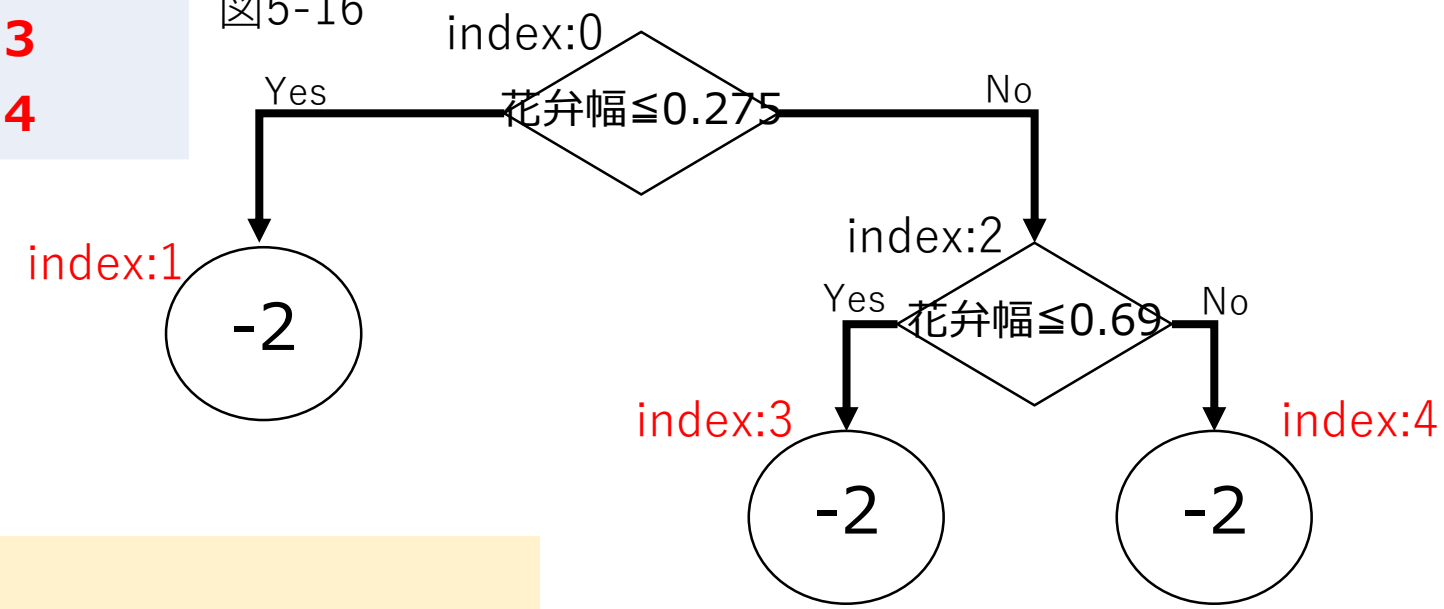
```
[[34.  0.  0.]] ←index:1
[[ 0. 31.  6.]] ←index:3
[[ 0.  1. 33.]] ←index:4
```

コード5-27
アヤメの種類とグループ番号の対応を調べる

```
model.classes_
```

```
array(['Iris-setosa', 'Iris-versicolor',
       'Iris-virginica'], dtype=object)
```

図5-16



※注意
先程のツリーとは別の形の例です

決定木のグループと分類の対応

P173、174

コード5-26
リーフに到達したデータの数を返す

```
print(model.tree_.value[1])
print(model.tree_.value[3])
print(model.tree_.value[4])
```

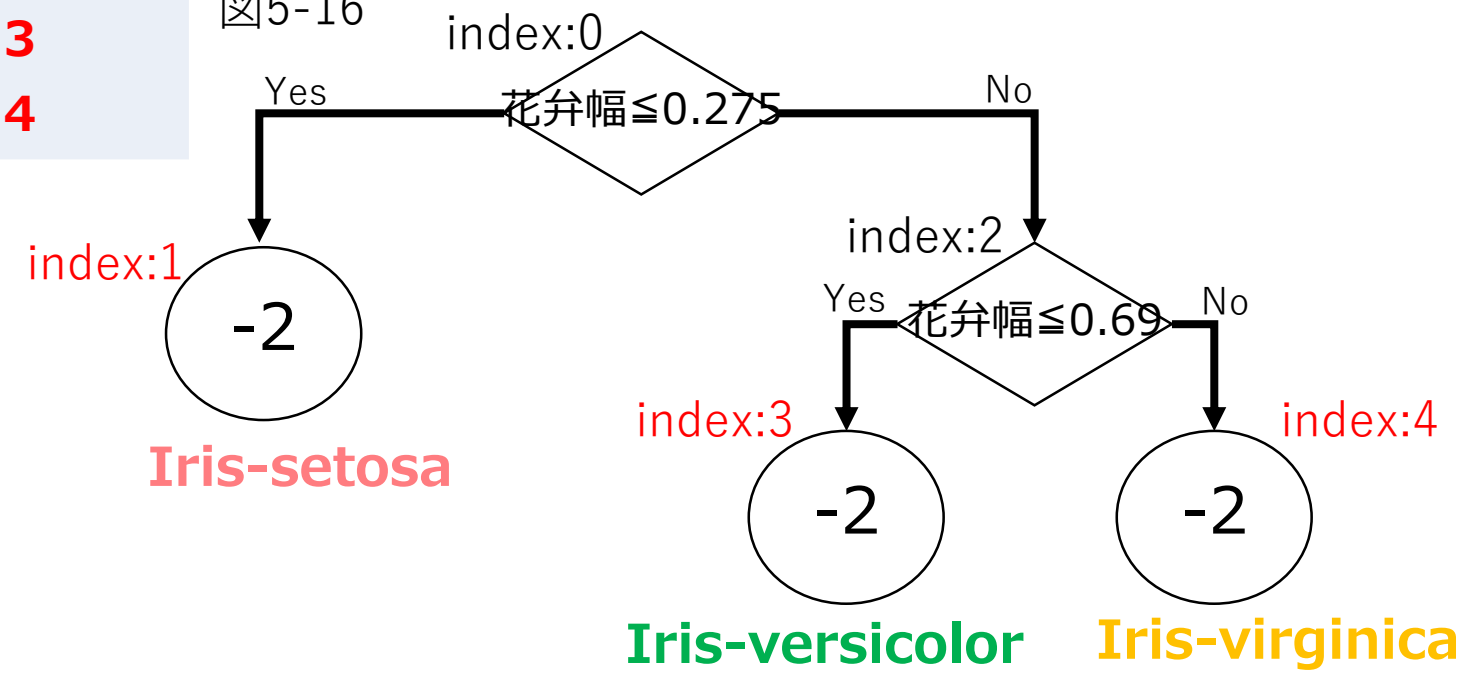
```
[[34.  0.  0.]] ←index:1
[[ 0. 31.  6.]] ←index:3
[[ 0.  1. 33.]] ←index:4
```

コード5-27
アヤメの種類とグループ番号の対応を調べる

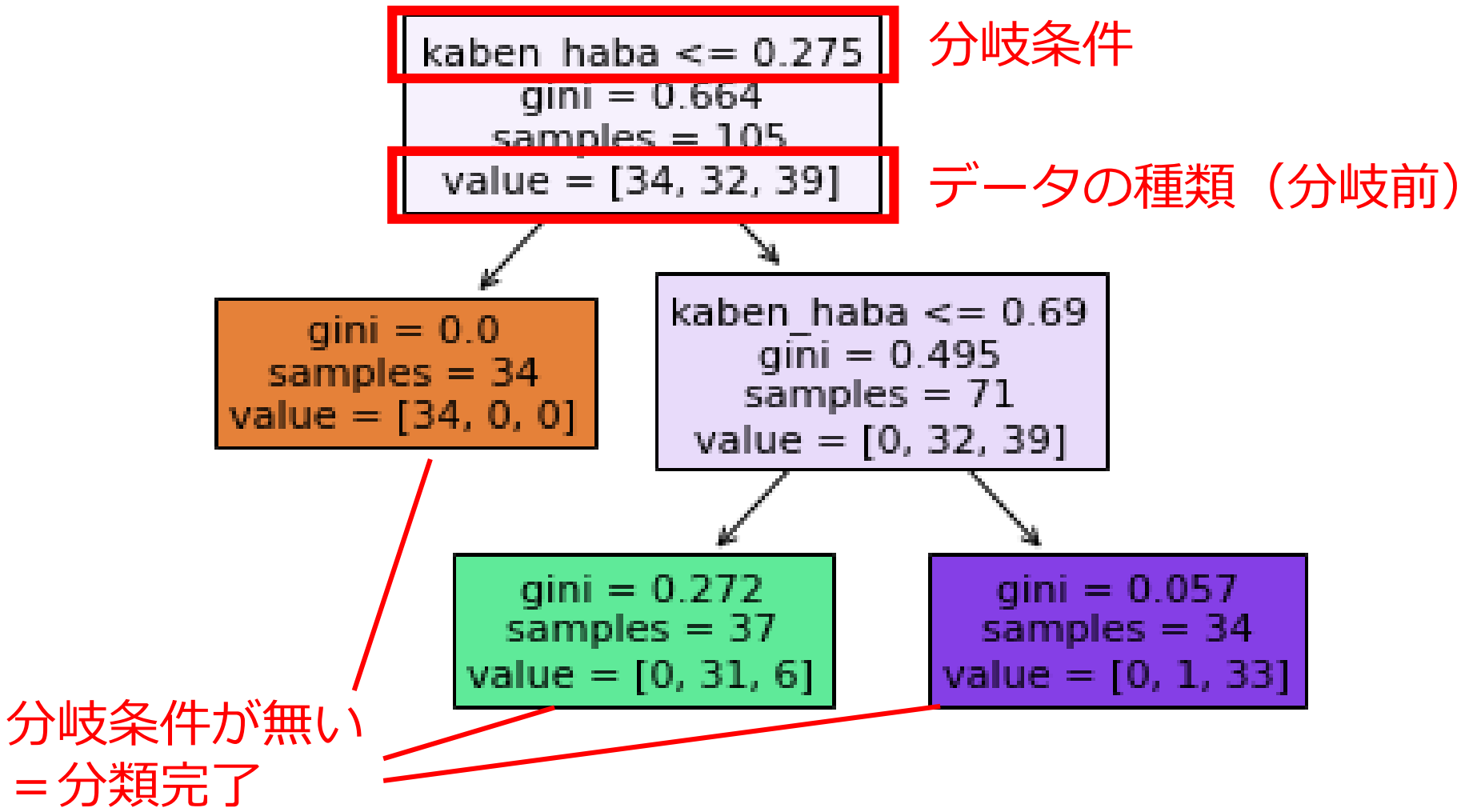
```
model.classes_
```

```
array(['Iris-setosa', 'Iris-versicolor',
       'Iris-virginica'], dtype=object)
```

図5-16



決定木の描画関数


kaben_haba <= 0.275
gini = 0.664
samples = 105
value = [34, 32, 39]
分岐条件
データの種類（分岐前）
gini = 0.0
samples = 34
value = [34, 0, 0]
kaben_haba <= 0.69
gini = 0.495
samples = 71
value = [0, 32, 39]
gini = 0.272
samples = 37
value = [0, 31, 6]
gini = 0.057
samples = 34
value = [0, 1, 33]
分岐条件が無い
＝分類完了