

## B-3) 決定木 (Decision Tree)

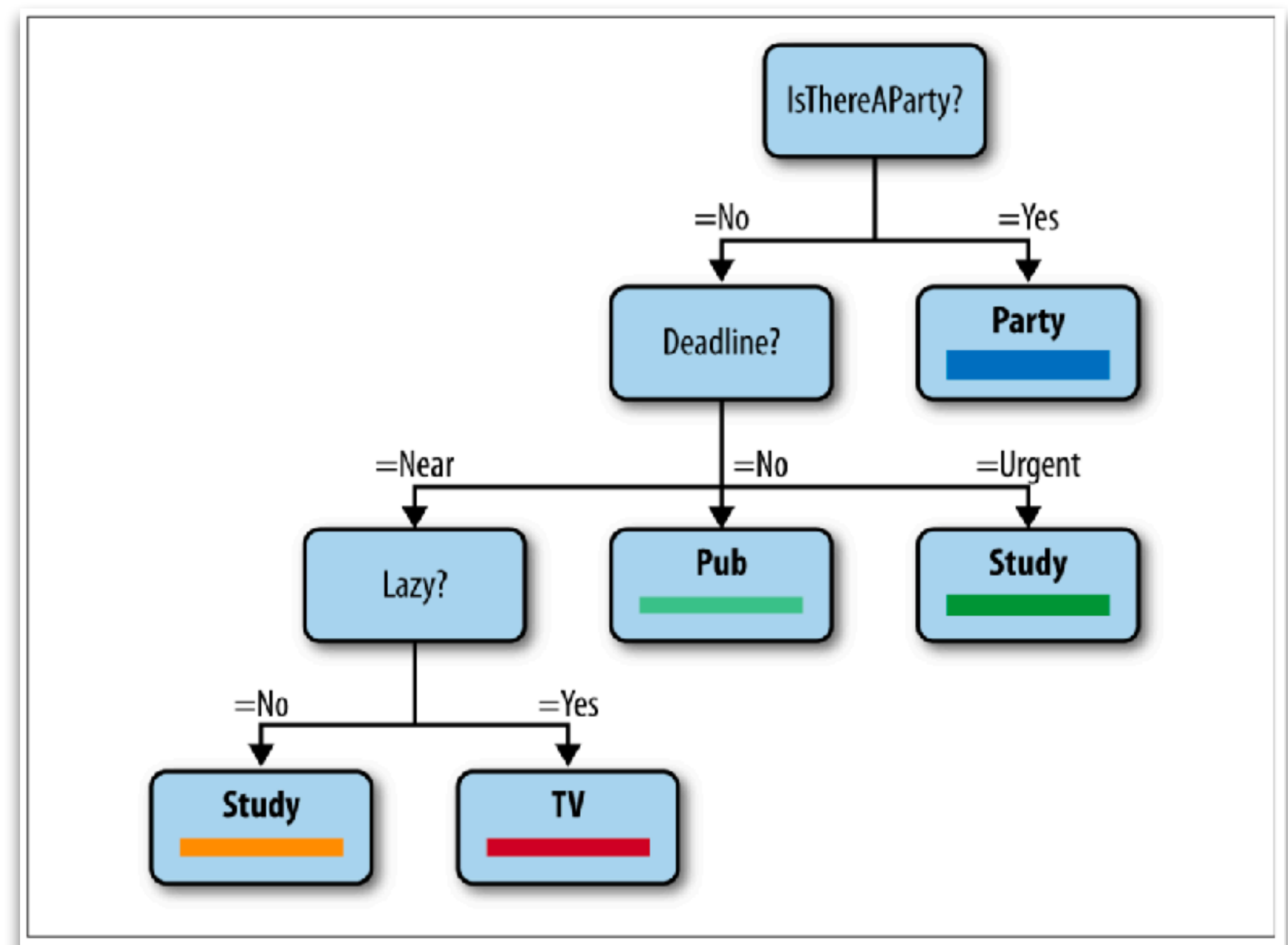
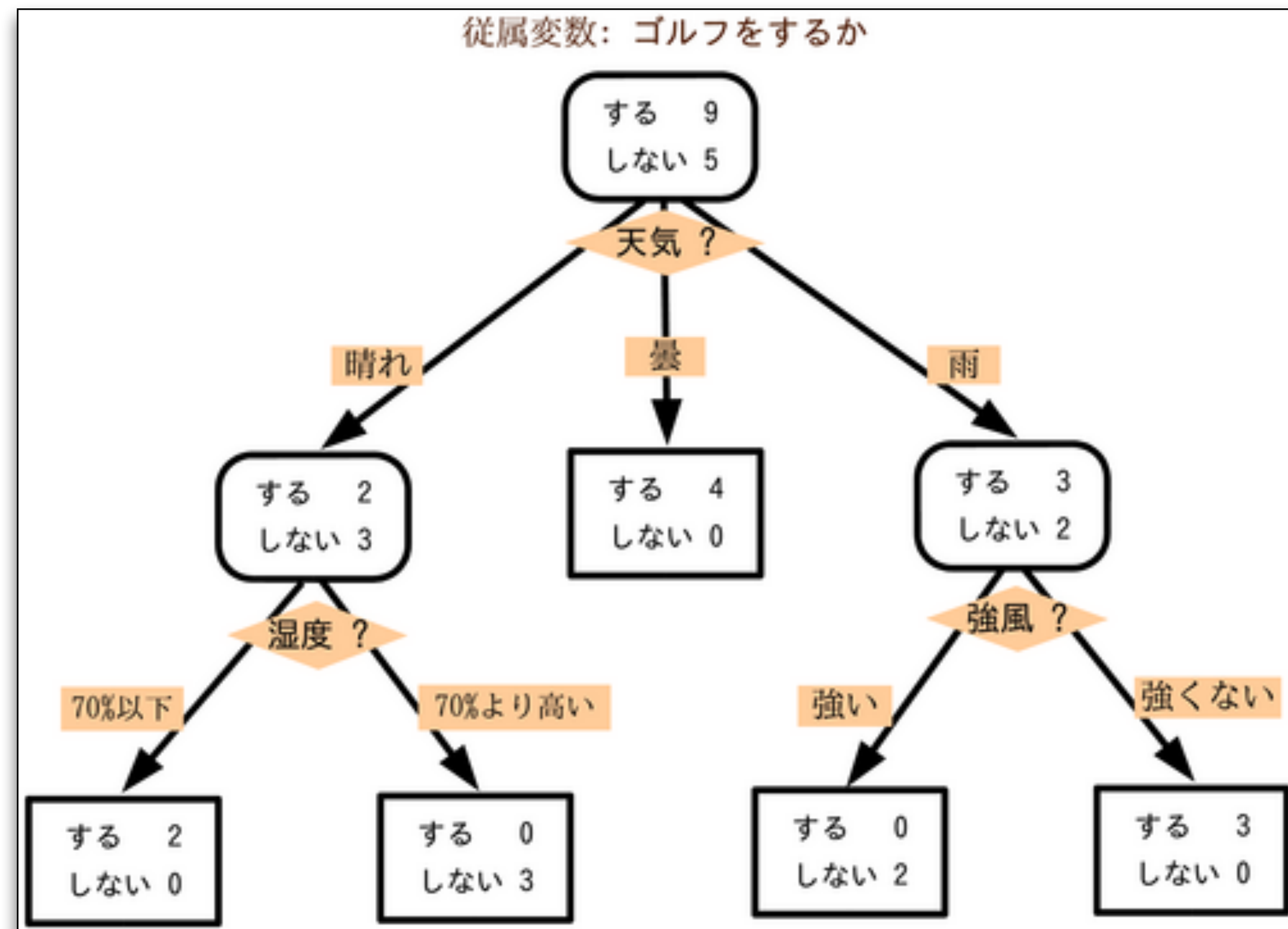
---

社会システム科学

# 決定木とは何か

# 決定木

- 説明変数に基づいて目的変数を決定
  - 例えば下の図では「天気」や「湿度」「パーティの有無」「締切の近さ」が説明変数（または独立変数）
  - 「ゴルフをする or しない」（左）または「当日の行動」（右）が目的変数（または従属変数）
- 木構造に繋がったルール群から構成される。



# 決定木の学習

---

- ルールの優先順位
  - 分割後の情報利得がより多くなるように，順にルールを決定していく。
  - つまり上のルール（先に適用されるルール）が情報利得の面では重要。
  - 情報利得 = ルールによる分割前の不純度 - 分割後の不純度
- 不純度 = 乱雑さ（異なる目的変数を持つ事例の混ざり具合）
  - エントロピー（交差エントロピー）

$$H(p) = - \sum_{\omega \in \Omega} P(\omega) \log P(\omega)$$

- ジニ係数

$$\begin{aligned} G(P) &= \sum_{i \neq j} P(i)P(j) = \sum_i P(i)(1 - P(i)) \\ &= 1 - \sum_i P(i)^2 \end{aligned}$$

Google Colabで決定木

# Google Colabで決定木

---

- 以下の手順で決定木を実際に使ってみます。
  1. Google Colabの計算機上に必要なソフトウェアをインストール
  2. 演習用のデータ作成
  3. Google Colabで決定木の演習

決定木の可視化に必要なパッケージ類のインストール

# 必要なソフトウェア類

---

- Google Colabの仮想コンピュータ上のソフトウェア
  - graphviz : DOTファイルで書かれたグラフを可視化するソフトウェア
- Pythonのパッケージ
  - scikit-learn : 機械学習用パッケージ
  - pandas : データ処理支援パッケージ
  - pydotplus : graphvizを使ってDOTファイルを可視化するパッケージ

この2つはインストールされていない場合があるので次の手順に従ってインストールする。



# Google Colabのコンピュータへのソフトウェアのインストール

1. コードセル上で以下のように入力

```
!apt install graphviz
```

—— サーバ上でコマンドを実行する

2. 実行されてインストールされる（下はインストール済みの場合の出力例）

```
1 !apt install graphviz
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
graphviz is already the newest version (2.40.1-2).
The following package was automatically installed and is no longer required:
  libnvidia-common-430
Use 'apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.
```

# Pythonパッケージのインストール

## 1. Python用の追加パッケージもサーバ側でインストール

```
!pip install pydotplus
```

—— サーバ上でコマンドを実行する

## 2. 実行されてインストールされる（下はインストール済みの場合の出力例）



```
1 !pip install pydotplus
```



```
Requirement already satisfied: pydotplus in /usr/local/lib/python3.6/dist-packages (2.0.2)
```

```
Requirement already satisfied: pyparsing>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from pydotplus) (2.4.5)
```

# 演習用データの準備

# 演習用のデータ

- 下記のようなデータを分析する（一部抜粋）
- Webサイトにエクセルファイルがあるのでそれをダウンロードする

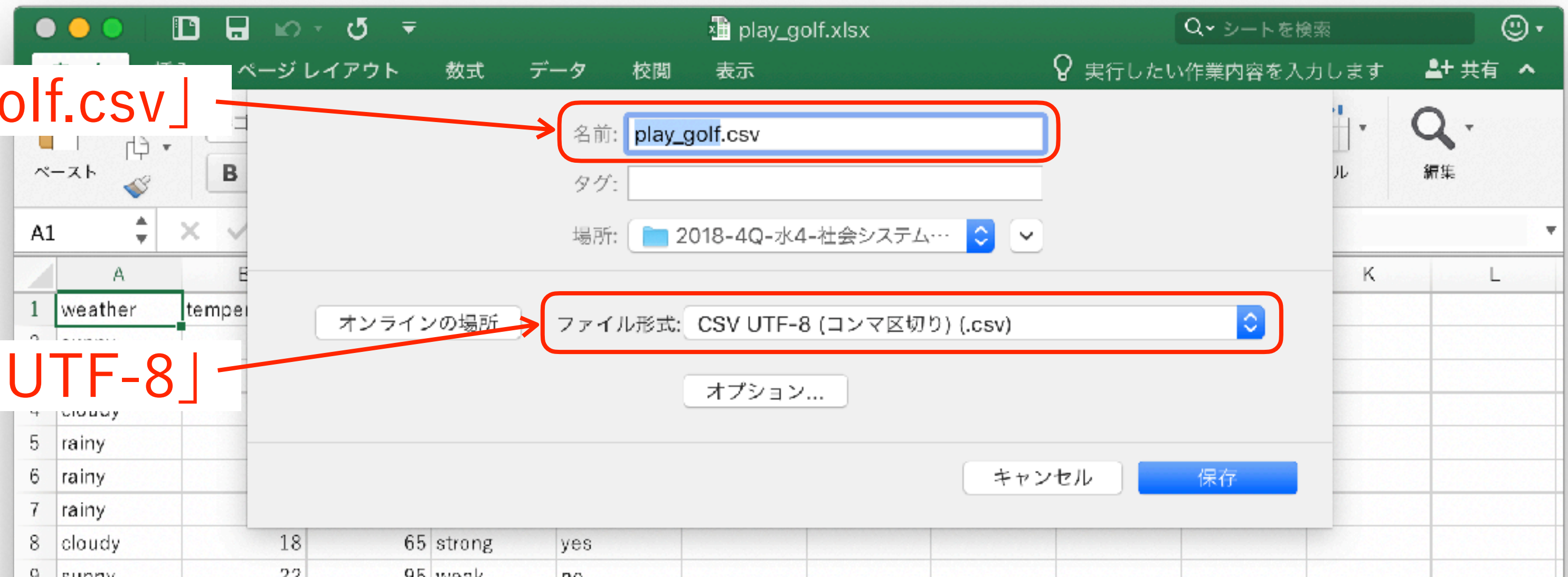
weather	temperature	humidity	wind	play_golf
sunny	29	85	weak	no
sunny	27	90	strong	no
cloudy	28	78	weak	yes
rainy	21	96	weak	yes
rainy	20	80	weak	yes
rainy	18	70	strong	no

# データの準備

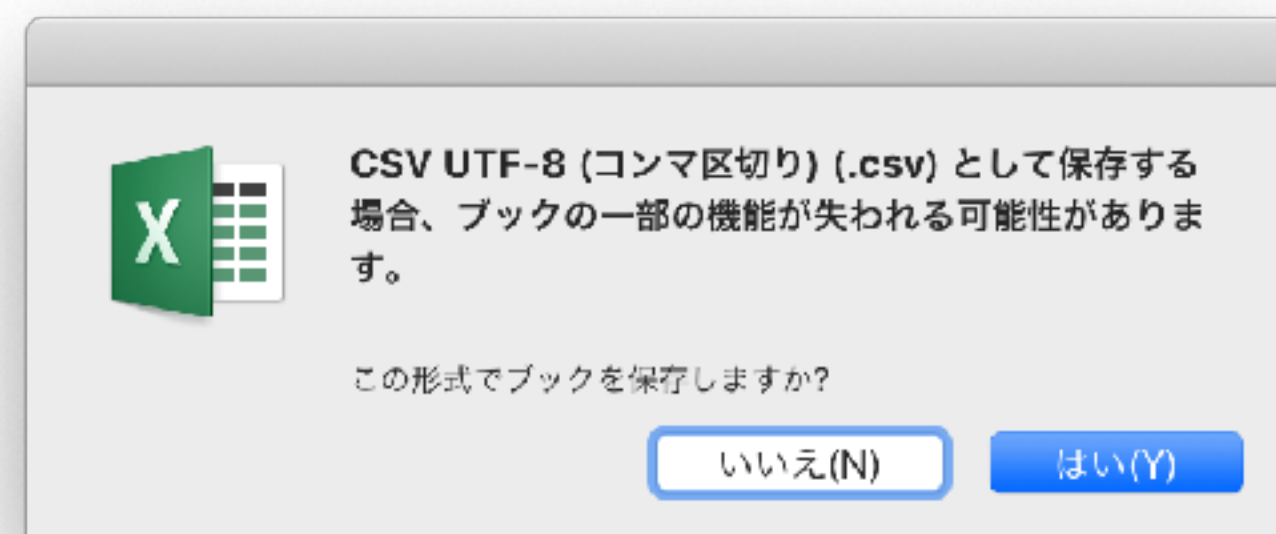
1. ダウンロードしたExcelファイルを開く
2. これをCSV形式で保存する（ファイル → 名前をつけて保存）

2-1. ファイル名は「play\_golf.csv」

2-2. ファイル形式は「CSV UTF-8」



3. 保存するときに下のようなダイアログが出たら「はい」を選択



# Google Colaboratoryへのデータのアップロード

---

- 方法1) ノートブックに直接アップロード
  - google.colab.files パッケージの upload() メソッドを利用して直接ファイルをアップロードする
- 方法2) セッションストレージの利用

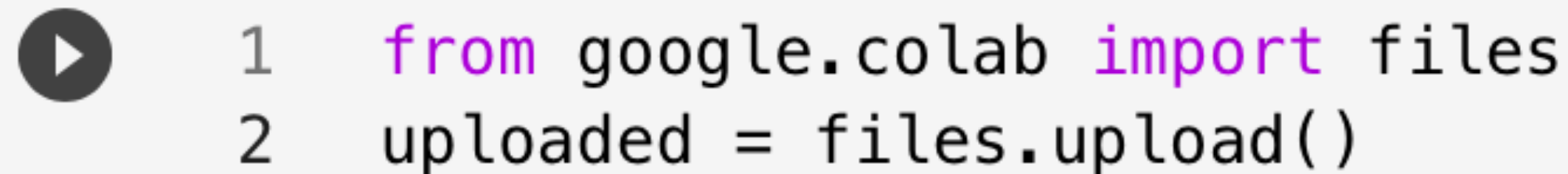


## 方法1) ノートブックに直接アップロード

1. 以下を実行するとファイルアップロード用のダイアログが開く

```
from google.colab import files  
uploaded = files.upload()
```

2. play\_golf.csv をアップロード



```
1 from google.colab import files  
2 uploaded = files.upload()
```



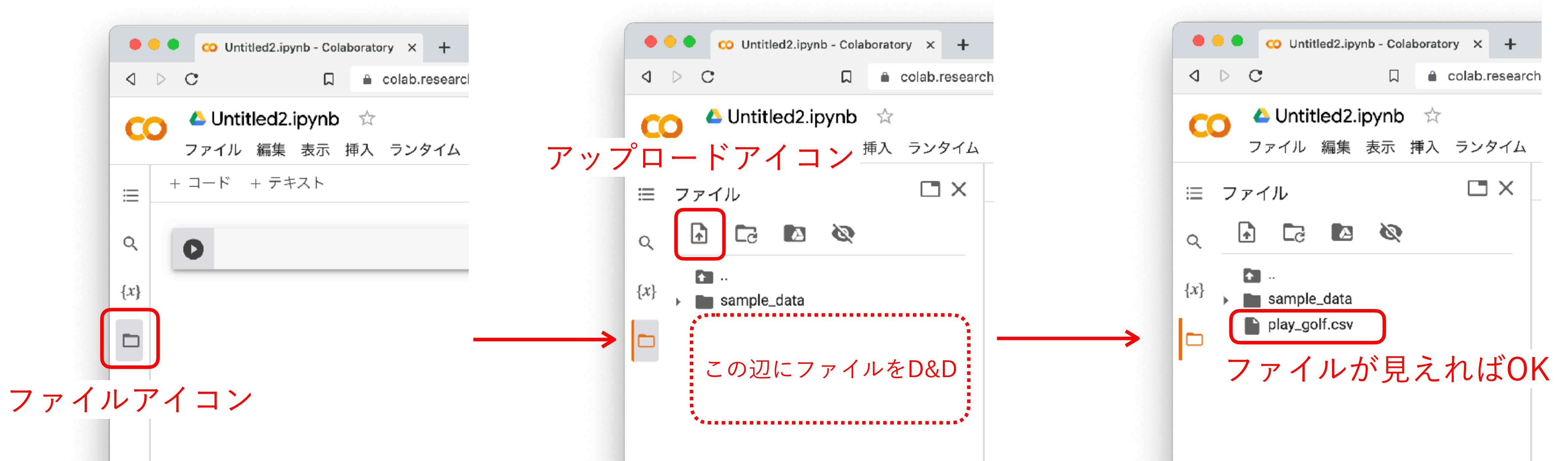
ファイルを選択 play\_golf.csv

• **play\_golf.csv**(text/csv) - 365 bytes, last modified: n/a - 100% done  
Saving play\_golf.csv to **play\_golf.csv**

これが実際に保存されたファイル名なので確認すること

## 方法2) セッションストレージの利用

1. Google Colabのウィンドウ左端のファイルアイコンをクリックしてストレージペインを開く
2. 以下のどちらかの方法でアップロード
  - ・ アップロードアイコンをクリックしてファイルを選択
  - ・ ストレージペインにファイルをドラッグ&ドロップ





**Google Colabで決定木**

# 必要なパッケージの読み込み

---

## 1. まず必要なパッケージを読み込む

```
import pandas as pd  
from sklearn.tree import DecisionTreeClassifier
```

# CSVファイルの読み込み

## 2. pandasを使ってCSVファイルを読み込む

```
data = pd.read_csv("play_golf.csv", index_col=None, header=0)
```

## 3. データの先頭を表示して確認

```
data.head()
```



```
1 data.head()
```



	<b>weather</b>	<b>temperature</b>	<b>humidity</b>	<b>wind</b>	<b>play_golf</b>
<b>0</b>	sunny	29	85	weak	no
<b>1</b>	sunny	27	90	strong	no
<b>2</b>	cloudy	28	78	weak	yes
<b>3</b>	rainy	21	96	weak	yes
<b>4</b>	rainy	20	80	weak	yes

# 文字列データを数値データに変換

## 4. 文字列データを数値データに変換

```
data['weather_val'] = data['weather'].map({'sunny':0, 'cloudy':1, 'rainy':2})  
data['wind_val'] = data['wind'].map({'strong':1, 'weak':0})  
data['play_golf_val'] = data['play_golf'].map({'yes':1, 'no':0})
```

「cloudy」を1に変換

## 5. データの確認

新しい列に数値化されたデータが追加されている

```
data.head()
```

	weather	temperature	humidity	wind	play_golf	weather_val	wind_val	play_golf_val
0	sunny	29	85	weak	no	0	0	0
1	sunny	27	90	strong	no	0	1	0
2	cloudy	28	78	weak	yes	1	0	1
3	rainy	21	96	weak	yes	2	0	1
4	rainy	20	80	weak	yes	2	0	1

# 学習用データとクラスラベルの分離

---

## 6. 学習用データ X とクラスラベル Y に分ける

```
X = data[['weather_val', 'temperature', 'humidity', 'wind_val']]
Y = data['play_golf_val']
```

# 決定木の学習

## 7. DecisionTreeClassifierオブジェクトの生成

```
clf = DecisionTreeClassifier()
```

## 8. DecisionTreeClassifierの学習

```
clf.fit(X, Y)
```



```
1 clf.fit(X,Y)
```

[illegible]

# [文法] DecisionTreeClassifierオブジェクトの生成

## [書式]

```
clf = sklearn.tree.DecisionTreeClassifier(criterion = 'gini' or 'entropy',  
                                           splitter = 'best' or 'random',  
                                           max_depth = None or 整数,  
                                           random_state = None or 整数)
```

## [主なオプション]

- criterion : 不純度
  - 'gini' : ジニ係数 / 'entropy' : 交差エントロピー
- splitter : 各ノードにおける分割方法
  - 'best' : 最適 / 'random' : ランダム最適
- max\_depth : 木の最大深さを指定
- random\_state : ランダムの種は None か整数で指定

# [文法] DecisionTreeClassifierの実行

---

## [書式]

```
clf.fit(X, Y)
```

## [入力データ]

- ・ X：学習に用いる変数セット
- ・ Y：教師ラベル（Xに含まれる変数の組に対する正解）



# 決定木の可視化

# 可視化用のパッケージの読み込み

---

## 9. 可視化用のパッケージを import

```
from sklearn import tree
import pydotplus.graphviz as gv
from IPython.display import Image
```

# 決定木の可視化用ファイルの保存

---

## 10. 決定木をDOT形式で保存

※DOT形式：graphvizでグラフを記述するためのテキストファイル形式

```
tree.export_graphviz(clf, out_file="play_golf.dot",  
                     feature_names = ['weather', 'temeprature', 'humidity', 'wind'],  
                     class_names = ['no', 'yes'])
```

- 以下のように学習データの列名を利用してもよい

```
tree.export_graphviz(clf, out_file="play_golf.dot",  
                     feature_names = X.columns,  
                     class_names = ['no', 'yes'])
```

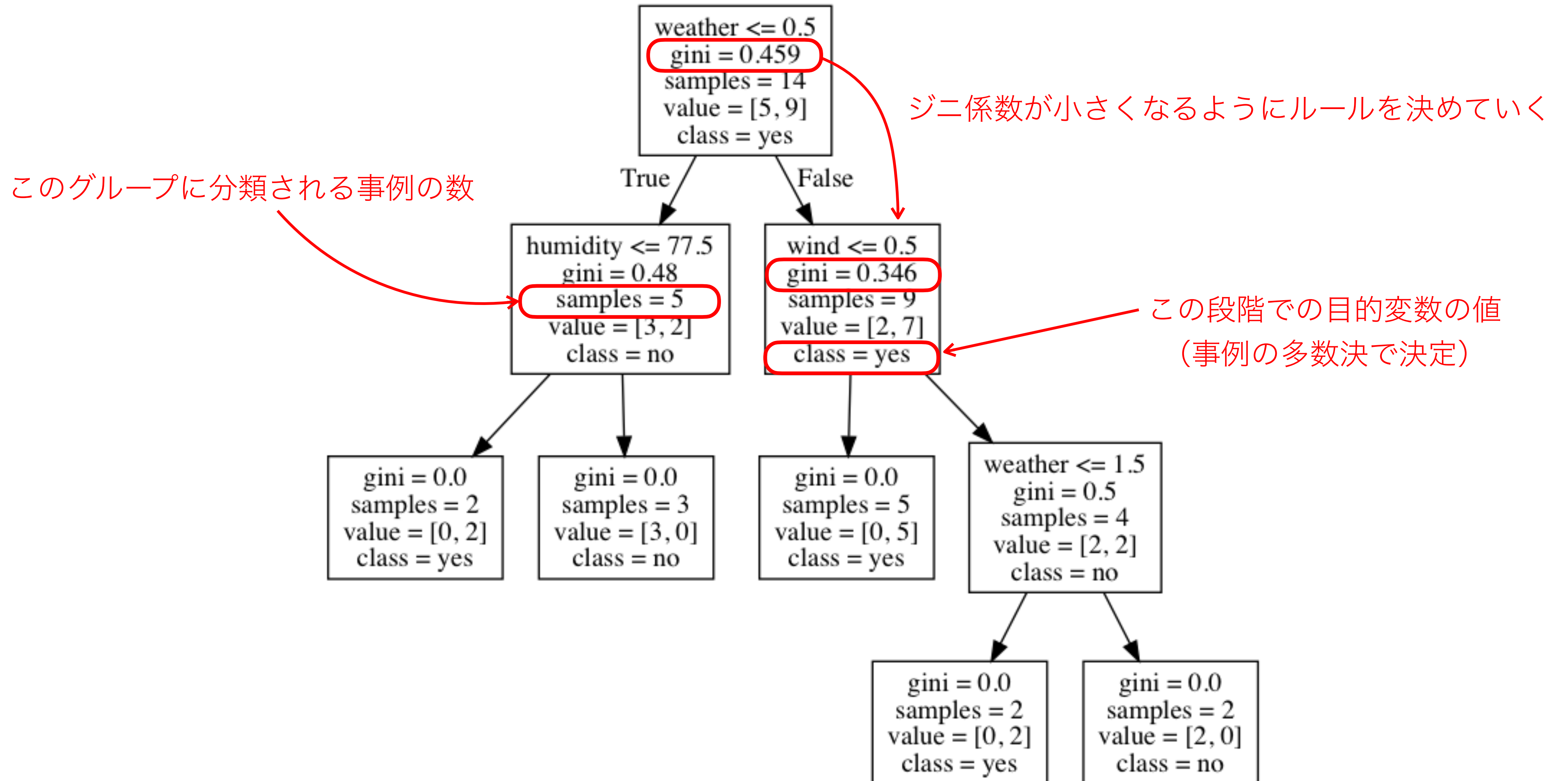
# 決定木の可視化

---

## 11. 決定木を graphviz を利用して可視化

```
graph = gv.graph_from_dot_file("play_golf.dot")  
Image(graph.create_png())
```

# 決定木の構造



# 決定木の利用（データの分類）

# 学習済みのデータを分類してみる

- 学習済みのデータ  $X$  に対して分類を試みる

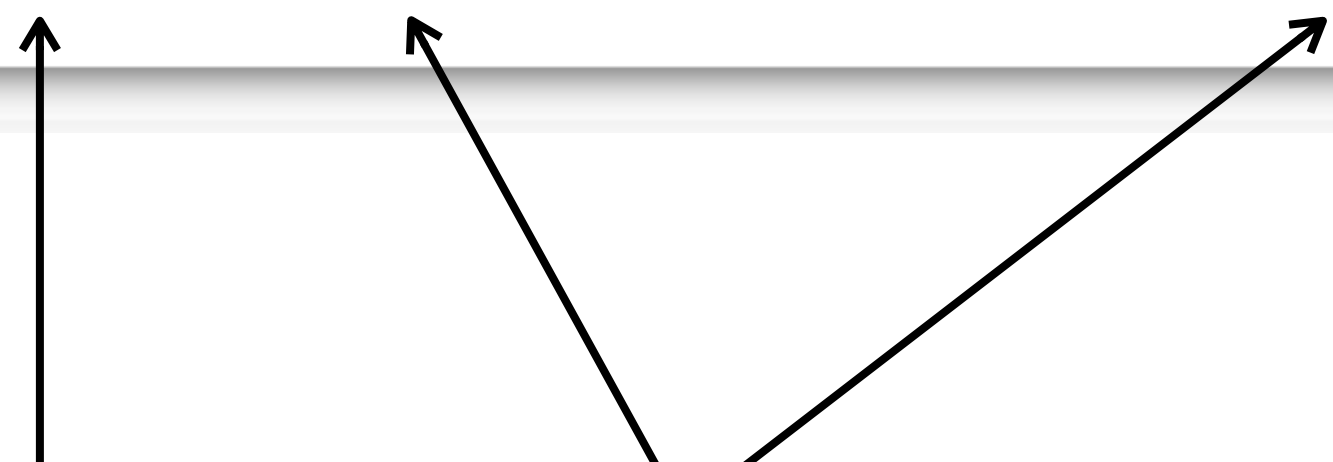
```
clf.predict(X)
```

```
1 clf.predict(X)
```

```
[> array([0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0])
```

no

yes



# 未学習のデータを入力してみる（予測）

- 未学習のデータ

ひとまとまりのデータ

```
clf.predict([ [ 1, 19, 40, 1 ] ])
```

天候 気温 湿度 風の強さ

```
1 clf.predict([[1,19,40,1]])
```

```
array([1])
```

ゴルフをするかどうか