# データ科学基礎演習B（7）

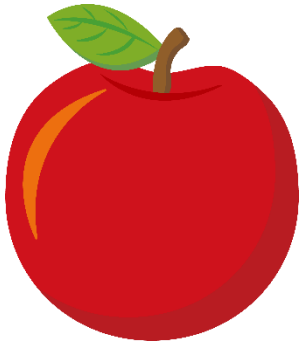
データ科学科目部会

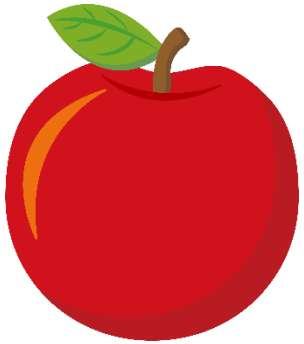
# 機械学習の基礎(分類)

## 機械学習って何？

# まずはじめに・・・

- 自動で「りんご」を分類／認識する機械を作りたい！

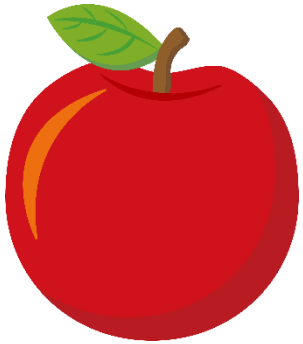




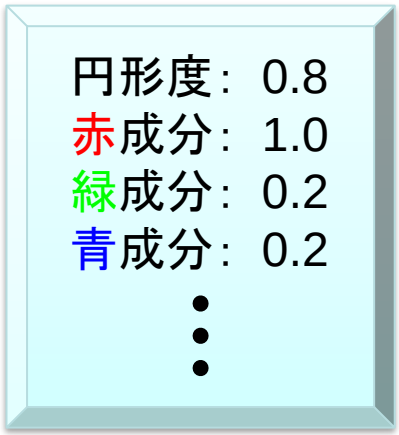
入力（画像）データ

# まずはじめに・・・

- 自動で「りんご」を分類／認識する機械を作りたい！
- 何らかの特徴（丸さ，色，…）を測れば分類できる？



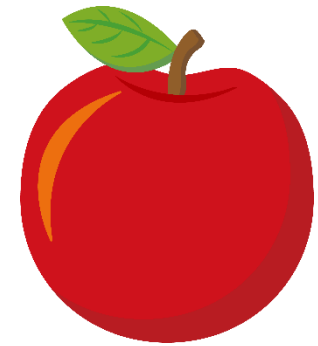
入力（画像）データ



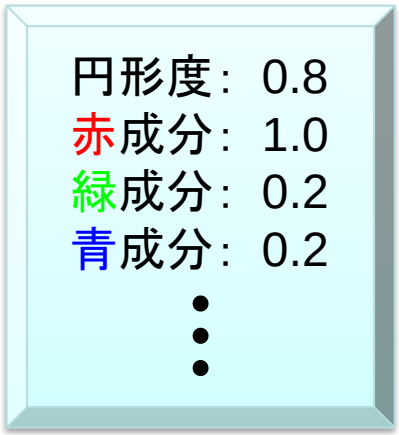

特徴量の記述

# まずはじめに・・・

- 自動で「りんご」を分類／認識する機械を作りたい！
- 何らかの特徴（丸さ，色，…）を測れば分類できる？
  - 特徴を数値の並び（ベクトル）として扱うことを考える



入力（画像）データ



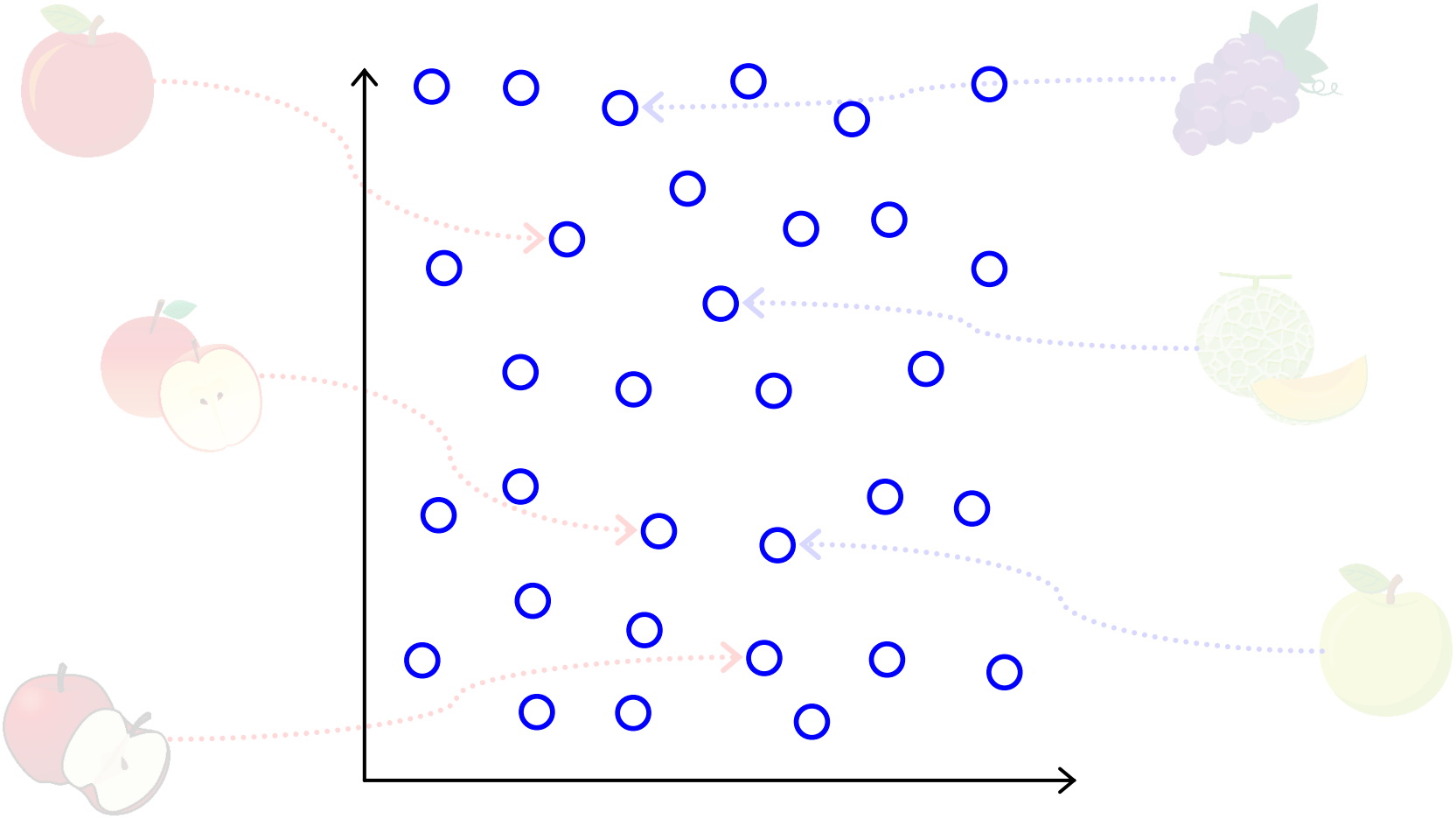

特徴量の記述

$$\begin{pmatrix} 0.8 \\ 1.0 \\ 0.2 \\ 0.2 \\ \vdots \end{pmatrix}$$

ベクトル化

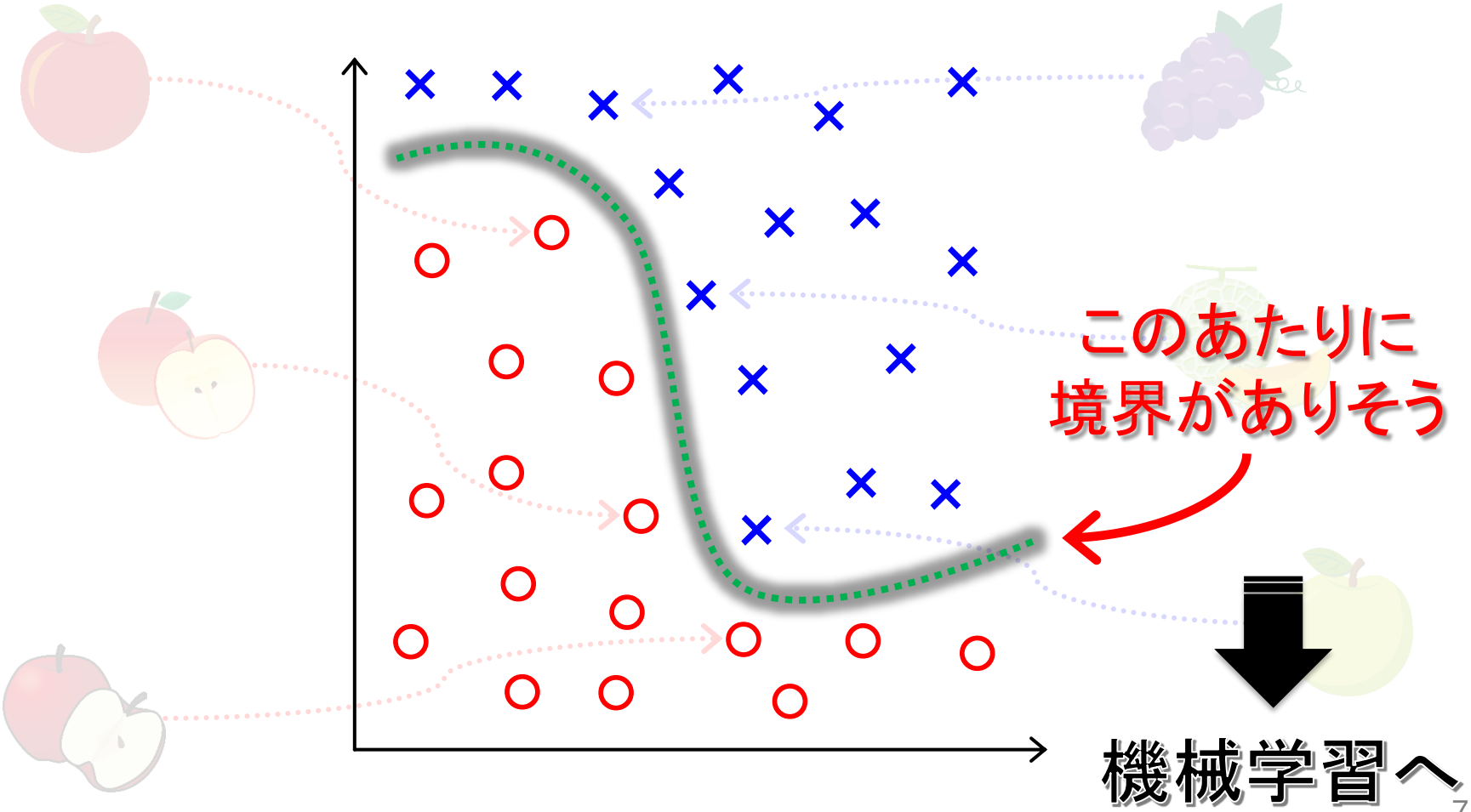
# まずはじめに・・・

- いろいろな果物の特徴（ベクトル）をプロットすると？



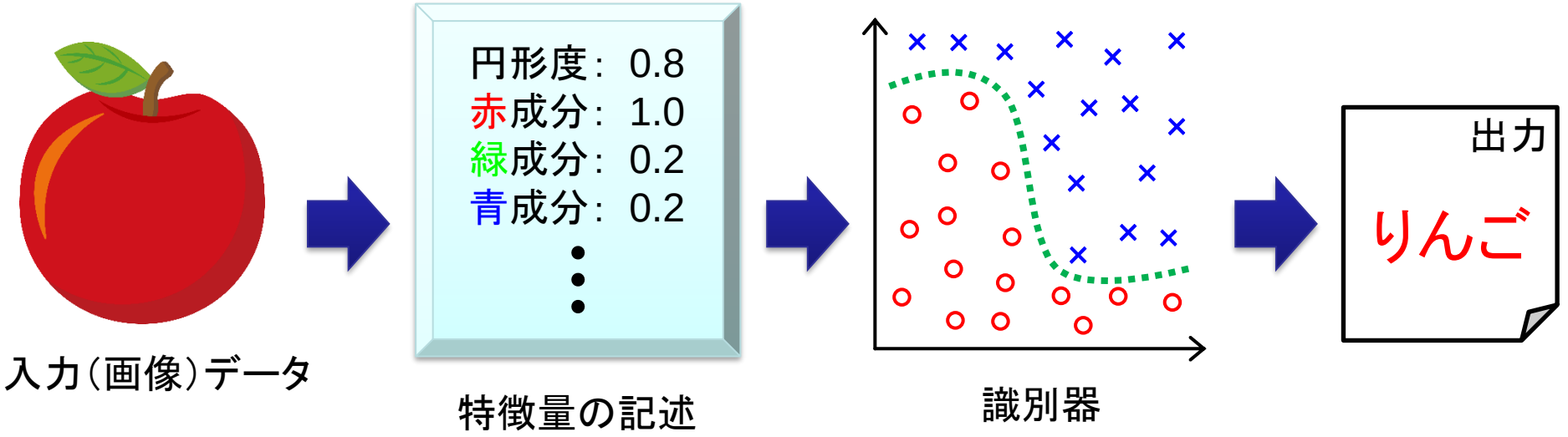
# まずはじめに・・・

- いろいろな果物の特徴（ベクトル）をプロットすると？
  - 「りんご」かどうかのラベル（＝教師データ）があれば・・・

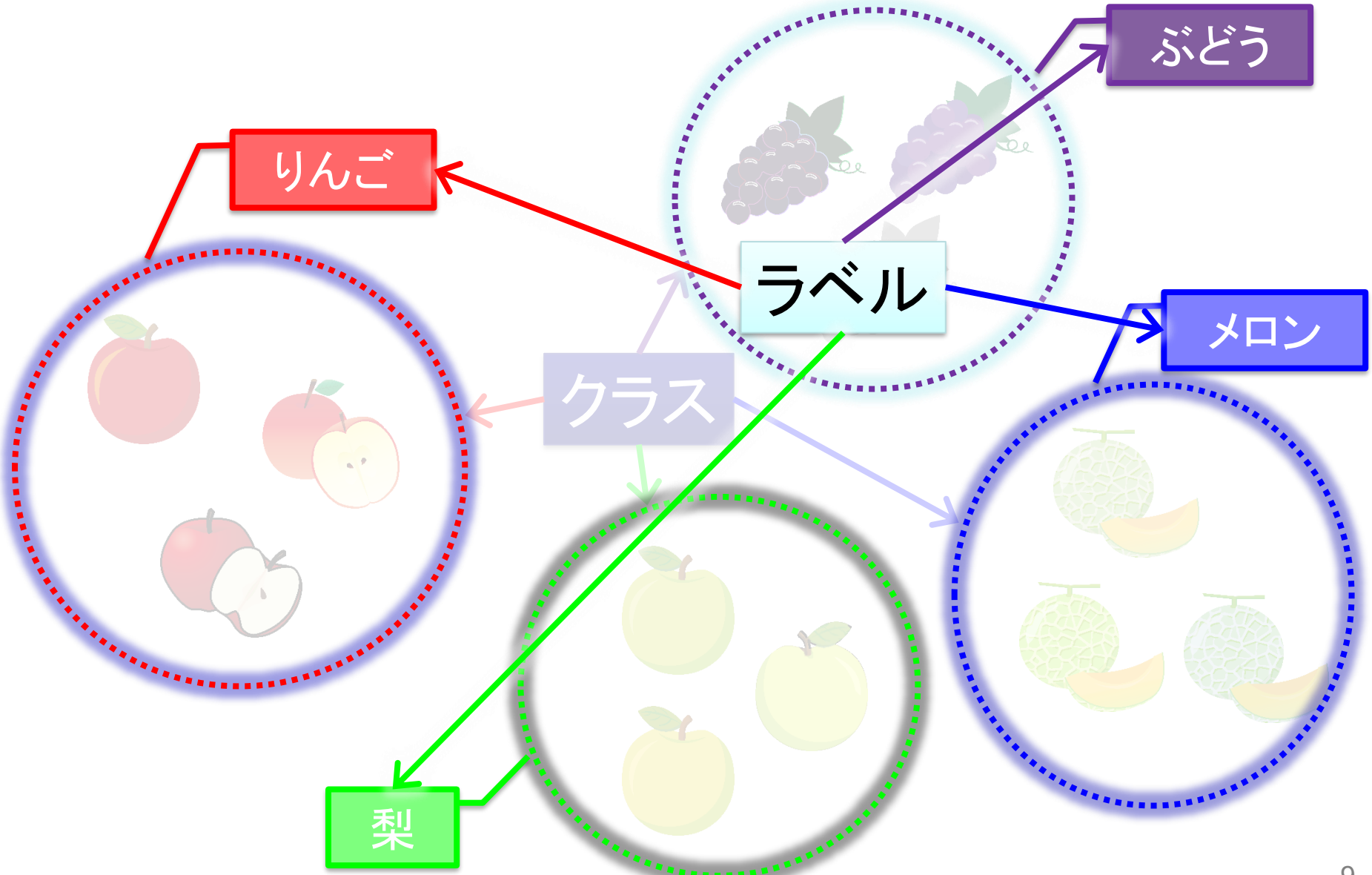


# 機械学習とは？

- 人間が自然に行っている学習能力と同様の機能をコンピュータで実現しようとする技術[Wikipediaより引用]
  - データの集合から何らかのルールや法則を計算機に学習させる技術

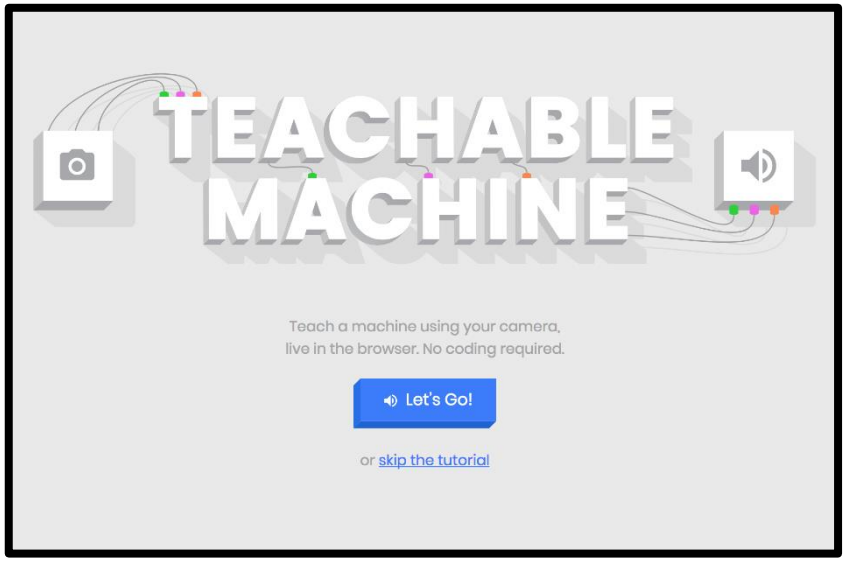


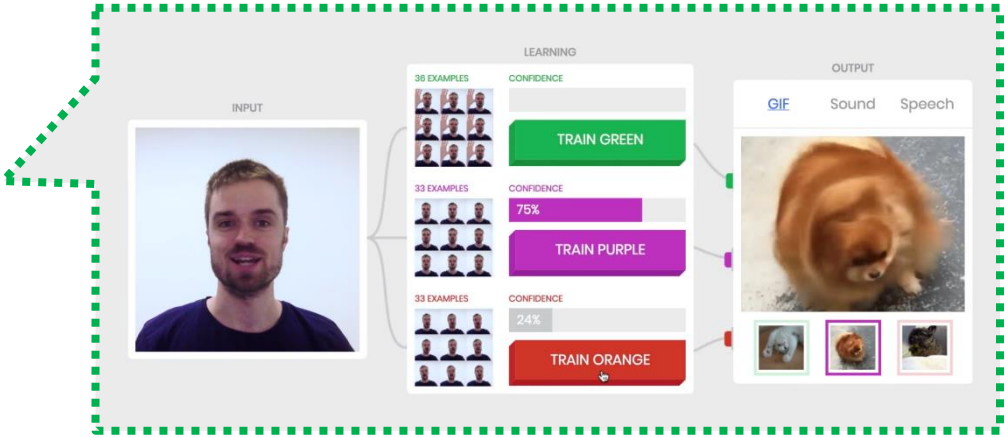
# 機械学習の用語（クラスとラベル）



# 機械学習でどんなことができるの？

- Teachable Machine で試してみよう！
  - WEBブラウザーを使って機械学習を体験できるサイト
  - 2017年にGoogleが公開




https://teachablemachine.withgoogle.com/

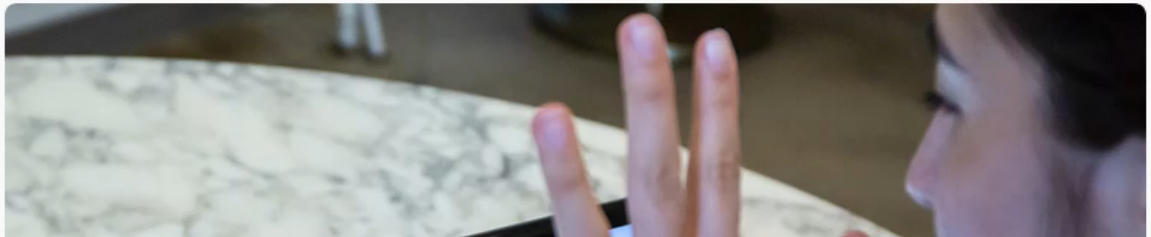
# Teachable Machine

## Train a computer to recognize your own images, sounds, & poses.

A fast, easy way to create machine learning models for your sites, apps, and more – no expertise or coding required.

Get Started

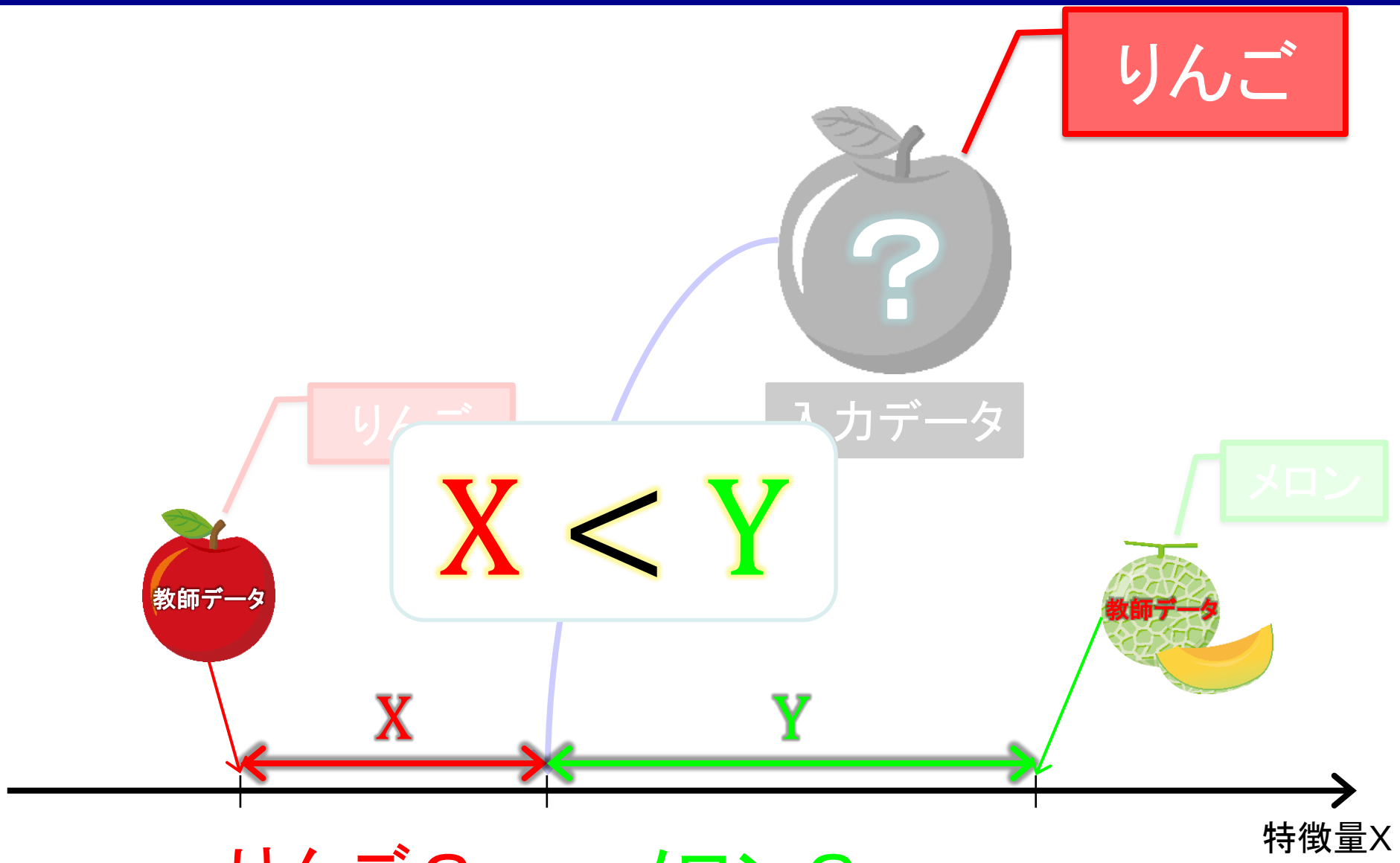

## What is Teachable Machine?



# K最近傍法

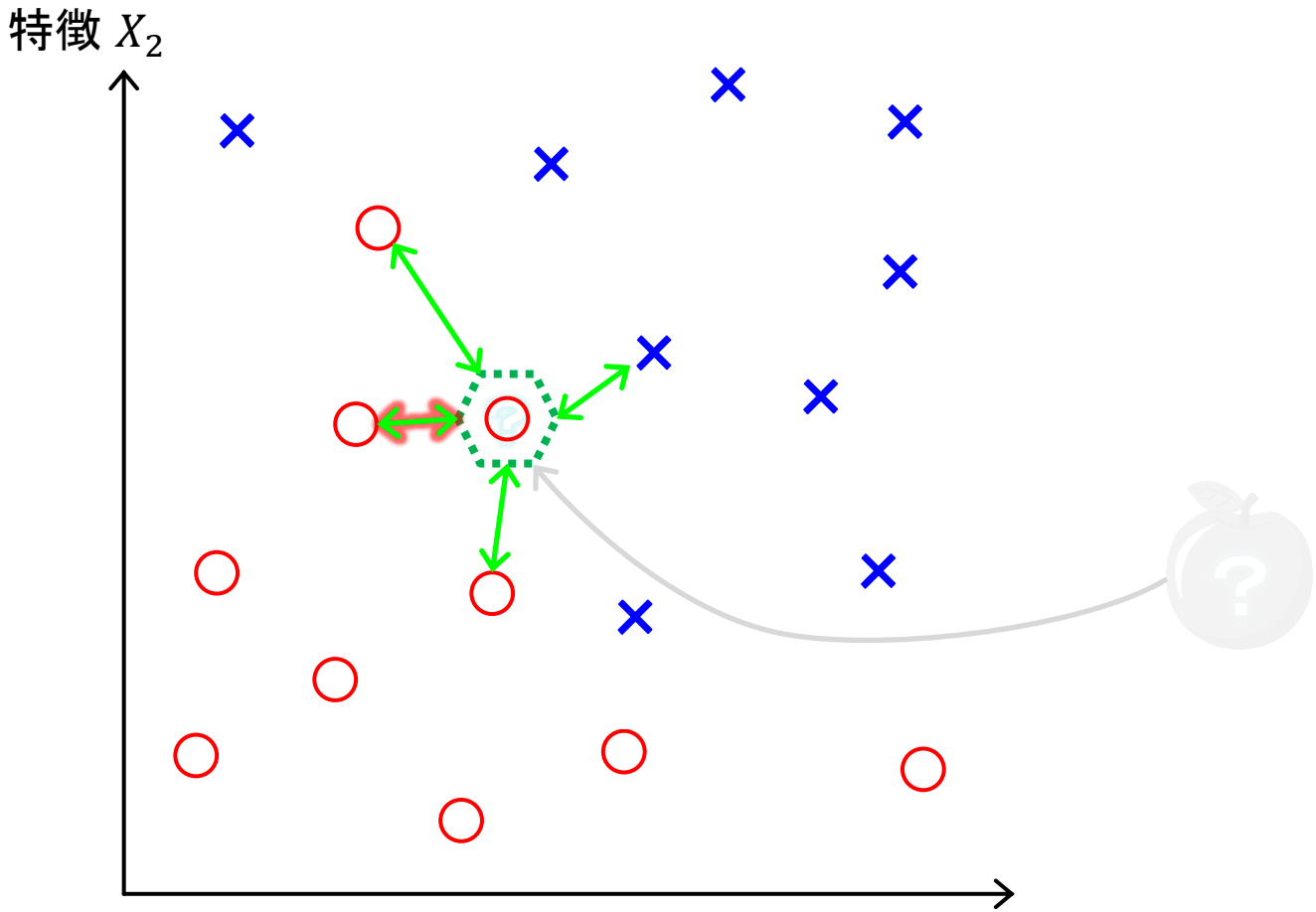
scikit-learnをマスターしよう（その1）

# 最近傍法（2クラスの場合）



# 最近傍法（2クラスの場合）

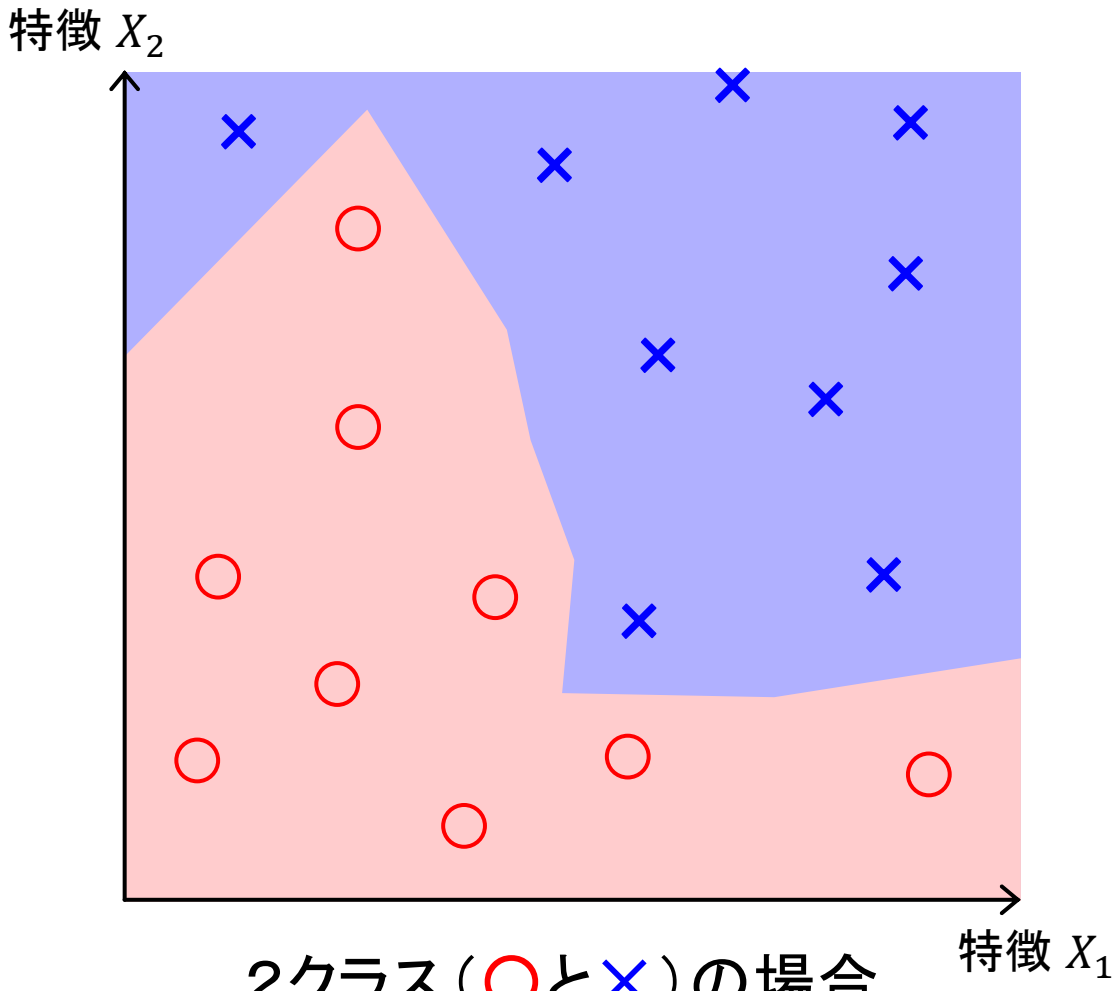
- 入力データに最も近い教師データを出力する手法




2クラス（○と×）の場合

# 最近傍法（2クラスの場合）

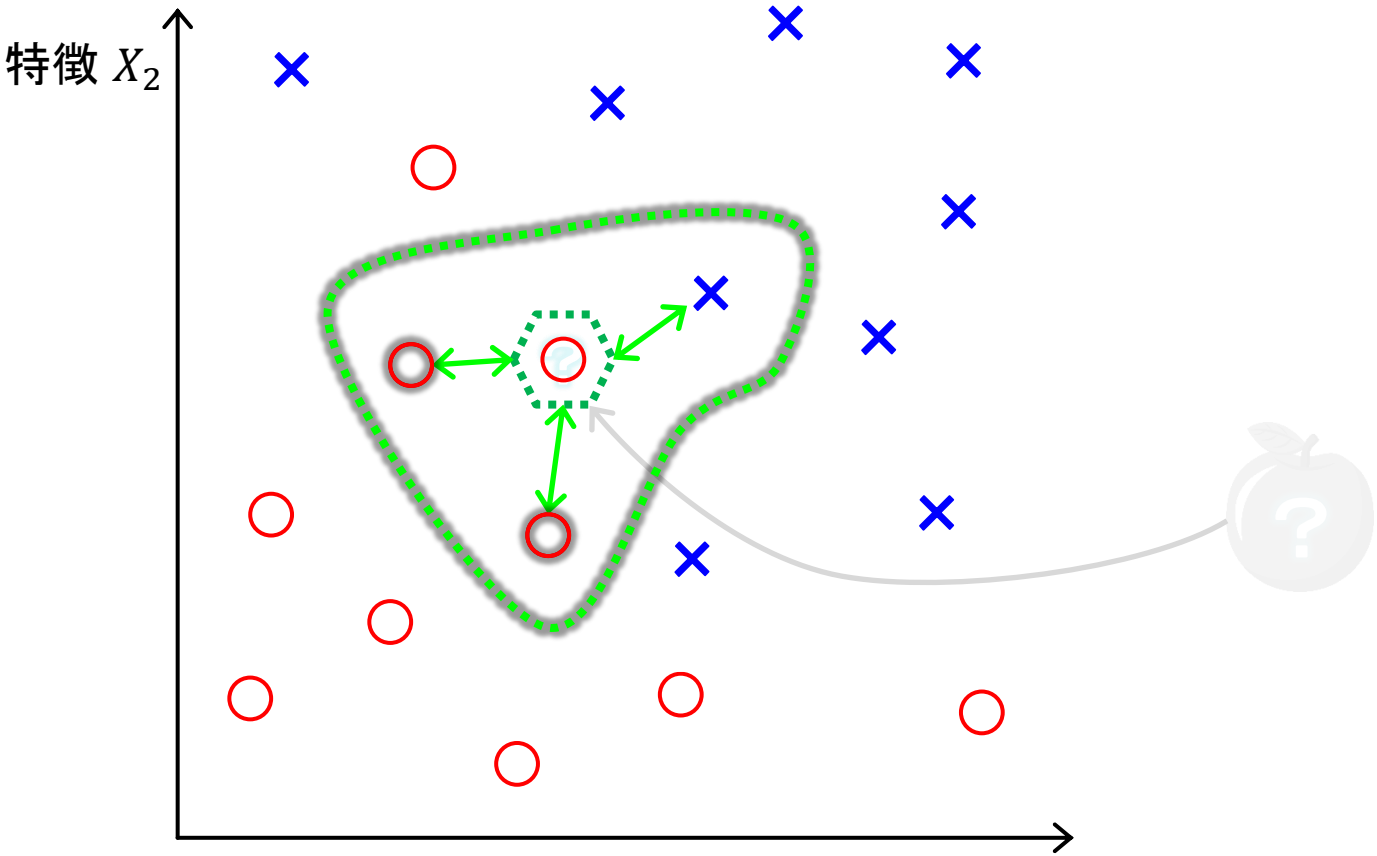
- 入力データに最も近い教師データを出力する手法




2クラス（○と×）の場合

# K最近傍法（2クラスの場合）

- 入力データのK近傍内で最も多く現れる教師データを出力する手法




3近傍（K=3）の場合

# scikit-learnの近傍モジュールをインポート

- sklearn モジュールの neighbors から KNeighborsClassifier をインポート

```
from sklearn.neighbors import KNeighborsClassifier
```

# K最近傍法を使うための準備

- KNeighborsClassifier を初期化

```
from sklearn.neighbors import KNeighborsClassifier
nn = KNeighborsClassifier(n_neighbors=5)
```

最近傍法に関する処理は
この変数に対して行う

近傍の大きさは
この値で指定

# scikit-learnを用いたK最近傍法

- 覚える関数は2つ
  - fit 関数

データからモデルを学習

    - 入力 $x$ と正解ラベル y のペアを $N$ 組与えてモデルを学習
  - predict 関数

モデルを使って出力を予測

    - 入力 $x$ から対応するラベル y を予測

      ※ fit関数で学習したモデルを用いてラベルを推定

# fit 関数の使い方

- 入力 $x$ と正解ラベル y のペアを引数に指定

```
nn.fit(入力X, 正解ラベルY)
```

最近傍法の変数
（2ページ前のnn）

# 入力 $x$ と正解ラベル y のペアの作り方(1)

- 入力 $x$ と正解ラベル y は別々の変数として用意
  - N個の入力を $x_1, x_2, \cdots, x_N$, 出力を $y_1, y_2, \cdots, y_N$
    ※添字が同じであれば入力と出力が対応
- 入力 $X$ の作り方
  - 各行が $x_1, x_2 \cdots, x_N$ に対応するよう並べた行列を作る
    - $i$ 個目のデータを $x_i = (x_{i1}, x_{i2}, \ldots, x_{iM})^T$ とすると

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1M} \\ x_{21} & x_{22} & \cdots & x_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{NM} \end{pmatrix}$$

$x_1$

$x_N$

線形回帰と同じデータの並び

# 入力 $x$ と正解ラベル y のペアの作り方(2)

- 入力 $x$ と正解ラベル y は別々の変数として用意
  - N個の入力を $x_1, x_2, \cdots, x_N$, 出力を $y_1, y_2, \cdots, y_N$
    ※添字が同じであれば入力と出力が対応

- 正解ラベル $Y$ の作り方
  - $y_1, y_2, \cdots, y_N$ が並んだ行ベクトル(配列)を作る
  - $y_1, y_2, \cdots, y_N$ はクラスラベル(1,2,…)

$$Y = (y_1, y_2, \cdots, y_N)$$

# 分類問題のデータセットを読み込もう（1）

- Iris Dataset
  - 非常に有名な機械学習用データセット
  - 4種類の特徴でアヤメを分類
    - 萼（ガク）の長さ
    - 萼（ガク）の幅
    - 花びらの長さ
    - 花びらの幅

Iris setosa

Iris versicolor

Iris virginica

※ 図はフリー百科事典『ウィキペディア（Wikipedia）』より引用

# 分類問題のデータセットを読み込もう(2)

– read_csv 関数で以下のURLからデータを読み込もう

**https://bit.ly/3rZLTgw**

```
[1]  1  import numpy as np
     2  import pandas as pd

[2]  1  pd.read_csv('https://bit.ly/3rZLTgw')
```

| | sepal length | sepal width | petal length | petal width | class name |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **...** | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

150 rows × 5 columns

# 学習用とテスト用にデータを分割しよう(1)

- 入力 $X$ (特徴量)とラベル $Y$ に分割
  - データの0～3列目を $X$, 4列目を $Y$ に代入する

```
[1]  1  import numpy as np
     2  import pandas as pd
     3  c  = pd.read_csv('https://bit.ly/3rZLTgw')
     4  a  = c.values
```

```
[2]  1  x = a[:,:4]
     2  y = a[:, 4]
```

```
[3]  1  x.shape
```

```
(150, 4)
```

```
[4]  1  y.shape
```

```
(150,)
```

# 学習用とテスト用にデータを分割しよう(2)

- $X$ と $Y$ を学習用データとテスト用データに分割
  - まず初めにデータ分活用の関数をインポート

```
from sklearn.model_selection import train_test_split
```

  - train_test_split関数でデータを分割

```
train_test_split(X, Y, train_size=0.8)
```

入力（特徴量）

正解ラベル

学習用データの割合（0～1）

  - train_test_split関数の戻り値
    - (x_train, x_test, y_train, y_test) のタプル ※順番に注意

# 学習用とテスト用にデータを分割しよう(2)

```
[1]  1  import numpy as np
     2  import pandas as pd
     3  c  = pd.read_csv('https://bit.ly/3rZLTgw')
     4  a  = c.values
```

```
[2]  1  x = a[:,:4]
     2  y = a[:, 4]
```

```
[3]  1  x.shape
```

```
(150, 4)
```

```
[4]  1  y.shape
```

```
(150,)
```

```
[5]  1  from sklearn.model_selection import train_test_split
```

```
[6]  1  x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8)
```

```
[7]  1  x_train.shape
```

```
(120, 4)
```

```
[8]  1  x_test.shape
```

```
(30, 4)
```

```
[9]  1  y_train.shape
```

```
(120,)
```

```
[10]  1  y_test.shape
```

```
(30,)
```

# fit 関数の使い方

- x_train と y_train 引数に指定

```
nn.fit(x_train,y_train)
```

```
[11]  1  from sklearn.neighbors import KNeighborsClassifier
      2  nn = KNeighborsClassifier(n_neighbors=5)

[12]  1  nn.fit(x_train, y_train)

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

# predict 関数の使い方

- x_test を引数に指定（複数の同時推定が可能）

```
nn.predict(x_test)
```

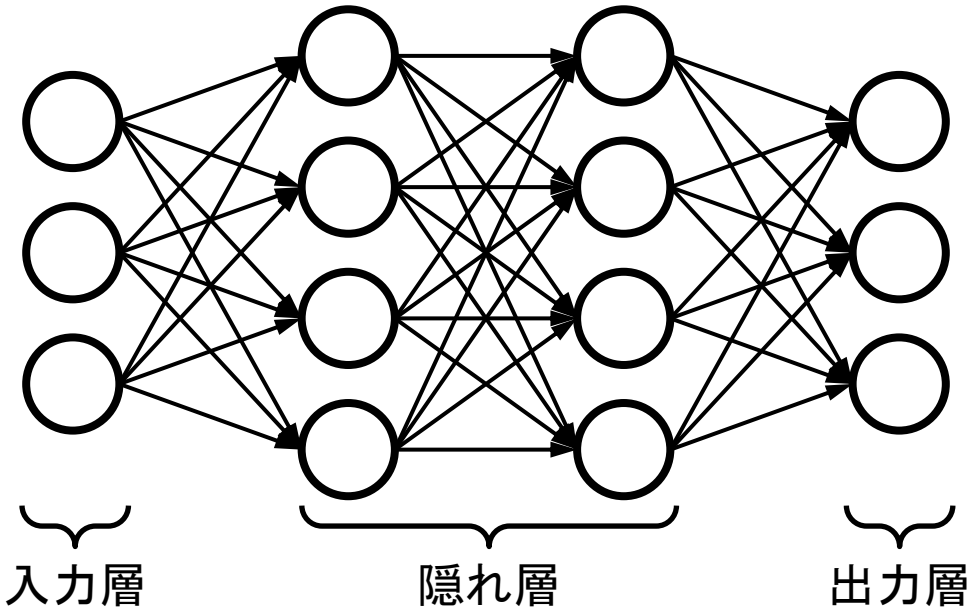
```
[14]  1  nn.predict(x_test)

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

[15]  1  y_test

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

# ニューラルネットワーク

scikit-learnをマスターしよう（その２）

# ニューラルネットワークとは？

- 脳の神経回路網に着想を得た機械学習技術
  - ディープラーニングや深層学習として近年注目の技術
    - 第3次AIブームの根幹をなす技術
    - ニューロンの出力を重み付け和し，活性化関数を通して次の層へと情報を伝達



# scikit-learnのモジュールをインポート

- ニューラルネットワークモジュールをインポート
  - sklearn モジュールの neural_network から MLPClassifier をインポート

```
from sklearn.neural_network import MLPClassifier
```

# ニューラルネットワークを使うための準備

- MLPClassifier を初期化

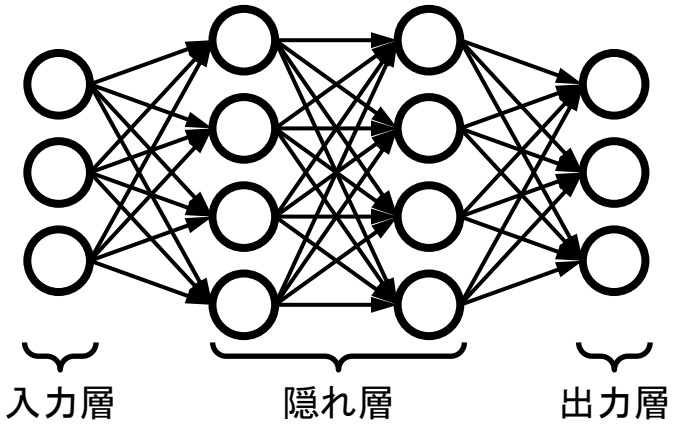
```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(パラメータを指定)
```

以降の処理はこの変数に対して行う

パラメータについては次ページ

# MLPClassifier のパラメータ

- hidden_layer_sizes
  - 隠れ層のユニット数を層の数だけタプルで並べる
    - （ユニット数1，ユニット数2，ユニット数3）※隠れ層3つの場合
- Activation:
  - 活性化関数（default: 'relu'）
- batch_size
  - 1回の反復で使う学習データ数
- learning_rate_init
  - 学習率の初期値（default: 0.001）
- max_iter
  - 学習の最大反復回数（default: 200）



# fit 関数の使い方（K最近傍法と同じ）

- x_train と y_train 引数に指定

```
mlp.fit(x_train,y_train)
```

```
[26]  1  from sklearn.neural_network import MLPClassifier

[27]  1  mlp=MLPClassifier(hidden_layer_sizes=(100,100,100))

[28]  1  mlp.fit(x_train, y_train)

MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100, 100, 100), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)

[29]  1  mlp.n_layers_

5
```

# predict 関数の使い方（K最近傍法と同じ）

- x_test を引数に指定（複数の同時推定が可能）

```
mlp.predict(x_test)
```

```
[30]  1  mlp.predict(x_test)

array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-virginica',
       'Iris-virginica'], dtype='<U15')

[31]  1  y_test

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

# モデルの評価

scikit-learnをマスターしよう（その3）

# 正解率（Accuracy）

- 予測結果がどの程度正しかったかを図る指標

$$Accuracy = \frac{\text{正しく予測できたデータ数}}{\text{評価データ数}}$$

  - sklearn モジュールの metrics から accuracy_score をインポート

```
[14]  1  from sklearn.metrics import accuracy_score

[15]  1  y_pred = mlp.predict(x_test)

[16]  1  accuracy_score(y_test, y_pred)

     0.9666666666666667
```

# 混同行列（Confusion Matrix）

- 予測結果の正解・不正解をクラス毎にまとめた表
  - sklearn モジュールの metrics から confusion_matrix と ConfusionMatrixDisplay をインポート

- confusion_matrix
  - 正解ラベルと予測結果から混同行列を計算

```
cm = confusion_matrix(正解ラベル, 予測結果)
```

- ConfusionMatrixDisplay
  - 混同行列をラベル名とともに色付けして表示

```
disp = ConfusionMatrixDisplay(cm, display_labels=ラベル名)
disp.plot()
```

混同行列を表示

nn や mlp 変数の classes_ を指定

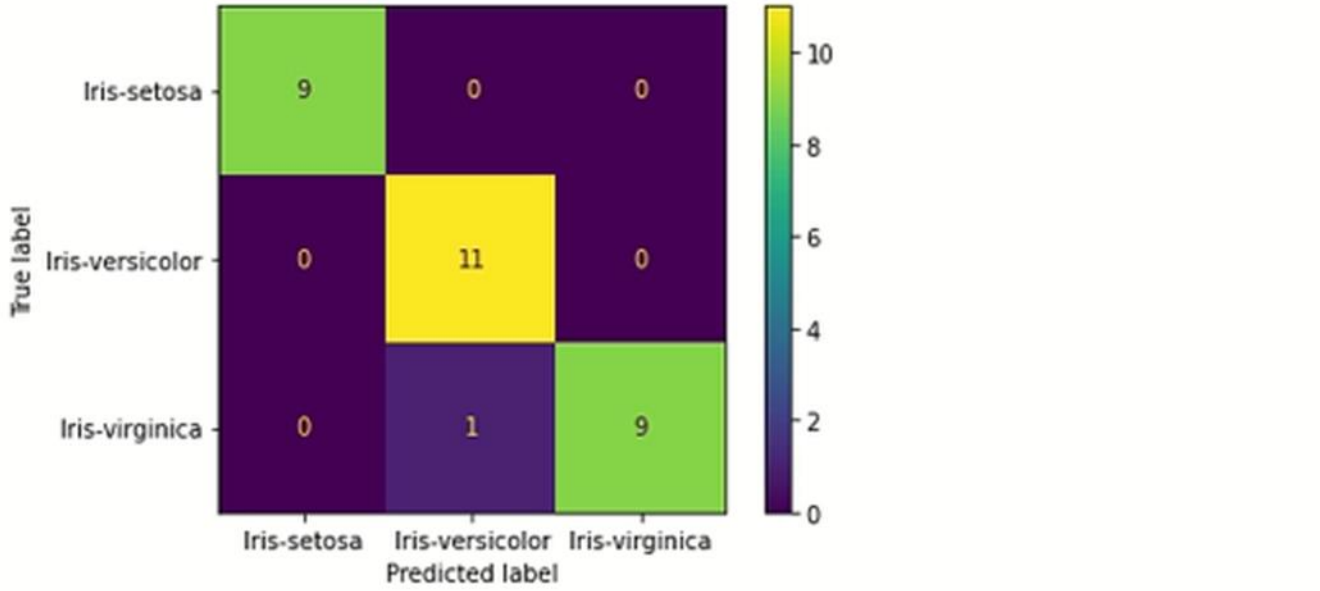
# 混同行列（Confusion Matrix）

```
[17] 1  from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```
[18] 1  cm = confusion_matrix(y_test, y_pred)
```

```
[19] 1  disp = ConfusionMatrixDisplay(cm, display_labels=mlp.classes_)
     2  disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7ff1c3dc7b38>
```



# 課題

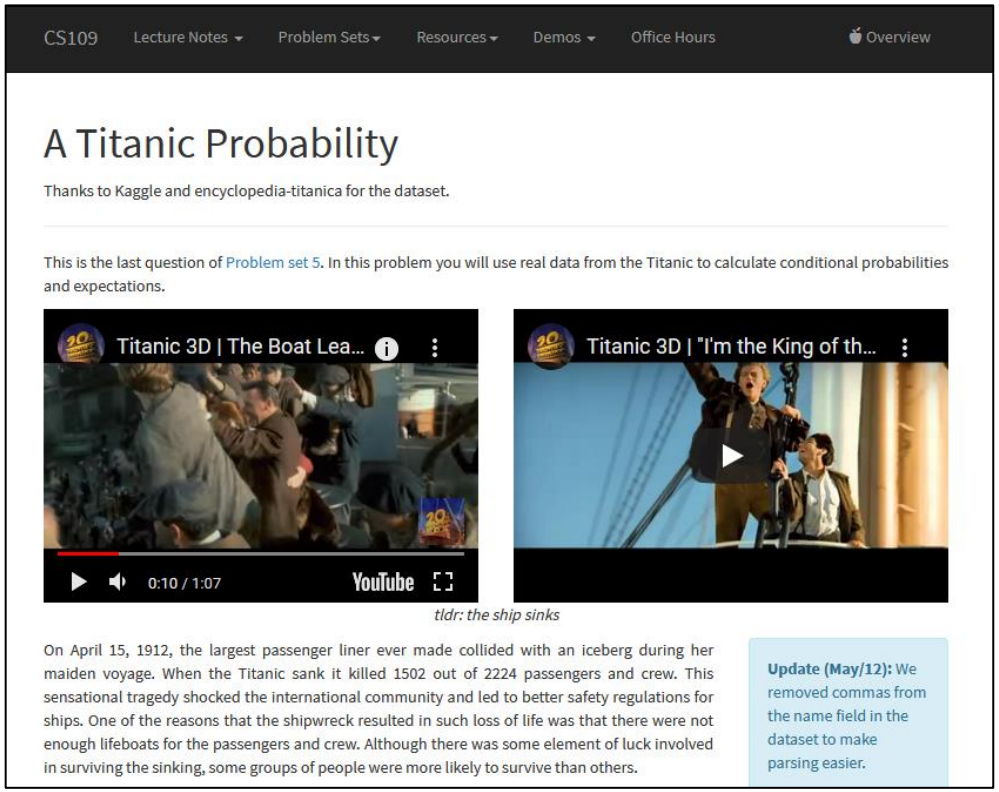
- タイタニック号の生存者を予測するモデルの構築にチャレンジしてみよう

- 下記URLから演習課題のノートブックを自身のGoogle Driveにコピーし，Google Collaboratoryを起動して各設問に回答すること
  - https://bit.ly/36CsUPu

- 演習課題を提出する際は，ノートブックのURLで共有し，そのURLを提出すること

# タイタニック号の生存者予測データセット

- Kaggleのチャレンジ課題「Titanic - Machine Learning from Disaster」で使われているデータセット
  - スタンフォード大学の講義でも使われている
    - http://stanford.io/3hWsoR7




**タイタニック号**

1912年4月に氷山に衝突して沈没したイギリス発でアメリカ行きの豪華客船

1997年に公開されたジェームズ・キャメロン監督の映画「タイタニック」のモデルとなった

# データセットの入手方法

- データセットのCSVファイルの入手先

https://bit.ly/3q3kCYF

- CSVファイルの中身

| Survived | Name | Sex | Age | Pclass | Siblings/Spouses Aboard | Parents/Children Aboard | Fare |
|---|---|---|---|---|---|---|---|
| 0 | Mr. Owen Harris Braund | 0 | 22 | 3 | 1 | 0 | 7.25 |
| 1 | Mrs. John Bradley (Florence Briggs Thayer) Cumings | 1 | 38 | 1 | 1 | 0 | 71.2833 |
| 1 | Miss. Laina Heikkinen | 1 | 26 | 3 | 0 | 0 | 7.925 |
| 1 | Mrs. Jacques Heath (Lily May Peel) Futrelle | 1 | 35 | 1 | 1 | 0 | 53.1 |
| 0 | Mr. William Henry Allen | 0 | 35 | 3 | 0 | 0 | 8.05 |

# CSVファイルの各列の中身

- Survived　# 生存者は 1，死亡は 0
- Name　# 搭乗者名
- Sex　# 男性は 0，女性は 1
- Age　# 年齢
- Pclass　# 搭乗券のクラス（1～3）
- Siblings/Spouses Aboard　# 同乗した兄弟・配偶者の数
- Parents/Children Aboard　# 同乗した親・子供の数
- Fare　# 運賃（ポンド）

| Survived | Name | Sex | Age | Pclass | Siblings/Spouses Aboard | Parents/Children Aboard | Fare |
|---|---|---|---|---|---|---|---|
| 0 | Mr. Owen Harris Braund | 0 | 22 | 3 | 1 | 0 | 7.25 |
| 1 | Mrs. John Bradley (Florence Briggs Thayer) Cumings | 1 | 38 | 1 | 1 | 0 | 71.2833 |

# 【補足】PandasからNumpyへのデータ受け渡し

- read_csvの戻り値に対して「.values」を追加することでNumpy配列に変換

```
c = pd.read_csv(…)
a = c.values #aはnumpy配列
```

数値，文字列，等が混ざった配列

```
[1]  1  import numpy as np
     2  import pandas as pd
     3  c  = pd.read_csv('https://bit.ly/3rZLTgw')
     4  a  = c.values
```

```
1  a
```

```
array([[5.1, 3.5, 1.4, 0.2, 'Iris-setosa'],
       [4.9, 3.0, 1.4, 0.2, 'Iris-setosa'],
       [4.7, 3.2, 1.3, 0.2, 'Iris-setosa'],
       [4.6, 3.1, 1.5, 0.2, 'Iris-setosa'],
       [5.0, 3.6, 1.4, 0.2, 'Iris-setosa'],
       [5.4, 3.9, 1.7, 0.4, 'Iris-setosa'],
```

# 【補足】PandasからNumpyへのデータ受け渡し

- PandasからNumpy配列に変換した際は，配列のデータの型は object 型になる
  - 最近傍法など正解ラベルにobject型は利用できない
- Numpy配列のデータ型を変換する方法
  - 方法1
    - dtypeを指定して新しい配列を作成する
  - 方法2
    - astype関数を利用して配列のデータ型を変換する

# Pythonにおけるデータ型の一例

- int
  - 整数を表すデータ型(符号有り)
- float
  - 浮動小数を表すデータ型
- bool
  - True/Falseのいずれかの真偽値を取るデータ型
- str
  - 文字列を表すデータ型

# Numpy配列のデータ型の変換(方法1)

- dtypeを指定して新しい配列を作成
  - 整数のデータ型(int)に変換する場合

```
np.array(Numpy配列,dtype=int)
```

```
[2]  1  a

array([[5, 3, 1],
       [4, 3, 1],
       [4, 3, 1]], dtype=object)

[3]  1  np.array(a, dtype=int)

array([[5, 3, 1],
       [4, 3, 1],
       [4, 3, 1]])
```

# Numpy配列のデータ型の変換(方法2)

- astype関数を利用して配列のデータ型を変換
  - 整数のデータ型(int)に変換する場合

```
Numpy配列.astype(dtype=int)
```

```
[2]  1  a

array([[5, 3, 1],
       [4, 3, 1],
       [4, 3, 1]], dtype=object)

[3]  1  a.astype(dtype=int)

array([[5, 3, 1],
       [4, 3, 1],
       [4, 3, 1]])
```

# 課題7での注意点

- 課題①-2で下記 [ エ ] を埋める際はデータ型をint型に変換すること

```
# pandas.DataFrameからnumpy.ndarrayに変換
a = [     イ     ]


# 特徴量Xおよび正解ラベルyを獲得
## 注意：yはデータ型をintに変換すること
X = [     ウ     ]
y = [     エ     ]
```

方法1もしくは方法2を使ってデータ型を変換した結果を変数 y に代入すること

# 課題提出時の注意点

- 課題提出の際には提出前に必ず以下2点を確認すること
  - Google Collaboratoryの共有設定の際には『リンクを知っている全員』にチェックを入れる
  - 課題提出時に貼り付けたURLの末尾が「?usp=sharing」となっている（共有設定を開き、「リンクのコピー」ボタンを押して共有用のURLをコピー＆ペーストする）