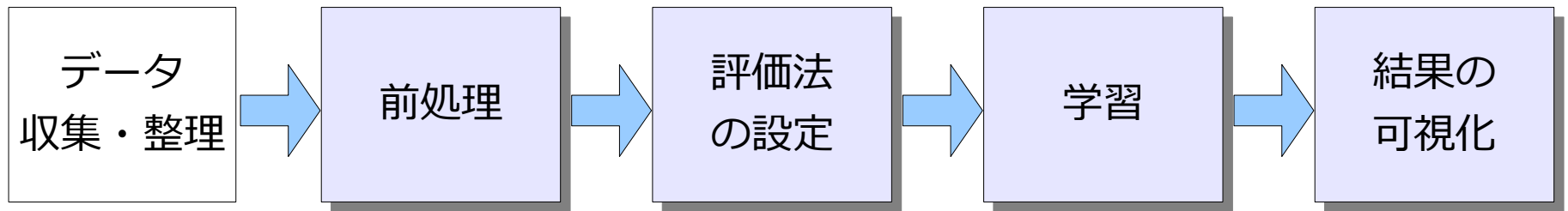



## 2. 機械学習の基本的な手順



 : ツールによる支援が可能

## 2.1 Weka を用いた機械学習

- Weka とは
  - Waikato Environment for Knowledge Analysis
  - 機械学習のアルゴリズムを実装した Java ライブラリ
  - データファイルを直接操作できる GUI を持つ
  - 説明は開発者版 3.9.3 に基づく
  - ライセンスは GNU GPL
    - プログラムの実行・改変・再配布が自由
    - ただし二次的著作物に対しても GNU GPL が適用される

# 勉強のためのデータセット

表 2.2 Weka 付属のデータ (一部)

| データ名            | 内容          | 特徴   | 正解情報      |
|-----------------|-------------|------|-----------|
| breast-cancer   | 乳癌の再発       | カテゴリ | クラス (2 値) |
| contact-lenses  | コンタクトレンズの推薦 | カテゴリ | クラス (3 値) |
| cpu             | CPU の性能評価   | 数値   | 数値        |
| credit-g        | 融資の審査       | 混合   | クラス (2 値) |
| diabetes        | 糖尿病の検査      | 数値   | クラス (2 値) |
| iris            | アヤメの分類      | 数値   | クラス (3 値) |
| ReutersCorn     | 記事分類        | 文字列  | クラス (2 値) |
| supermarket     | スーパーの購買記録   | カテゴリ | なし        |
| weather.nominal | ゴルフをする条件    | カテゴリ | クラス (2 値) |
| weather.numeric | ゴルフをする条件    | 混合   | クラス (2 値) |

# 起動

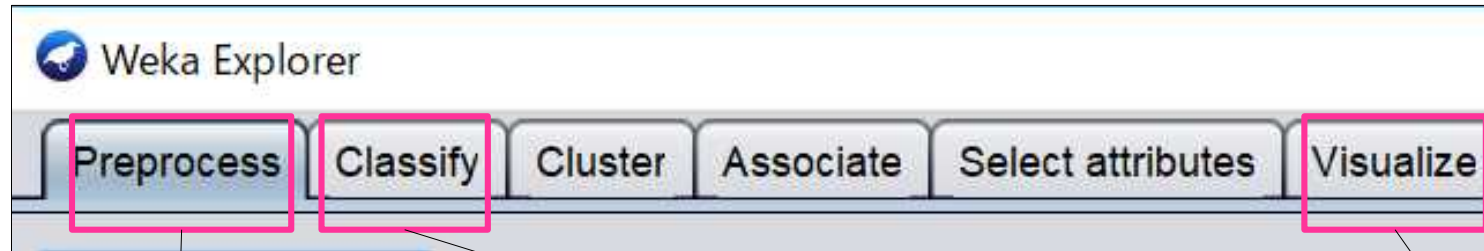
- アプリケーションの選択



- **Explorer** : データの読み込みから、特徴選択・学習・評価を試行錯誤的に行うのに適した操作を提供
- **Workbench** : すべてのアプリケーションをまとめた GUI (カスタマイズ可能)

- **Experimenter** : ハイパーパラメータ等を変えて性能を比較実験
- **KnowledgeFlow** : 実験プロセスを GUI で組み立て
- **SimpleCLI** : コマンドラインインタフェース

# Explorer での操作



- 前処理

- データの読み込み
- 標準化
- 特徴選択
- 特徴の分析

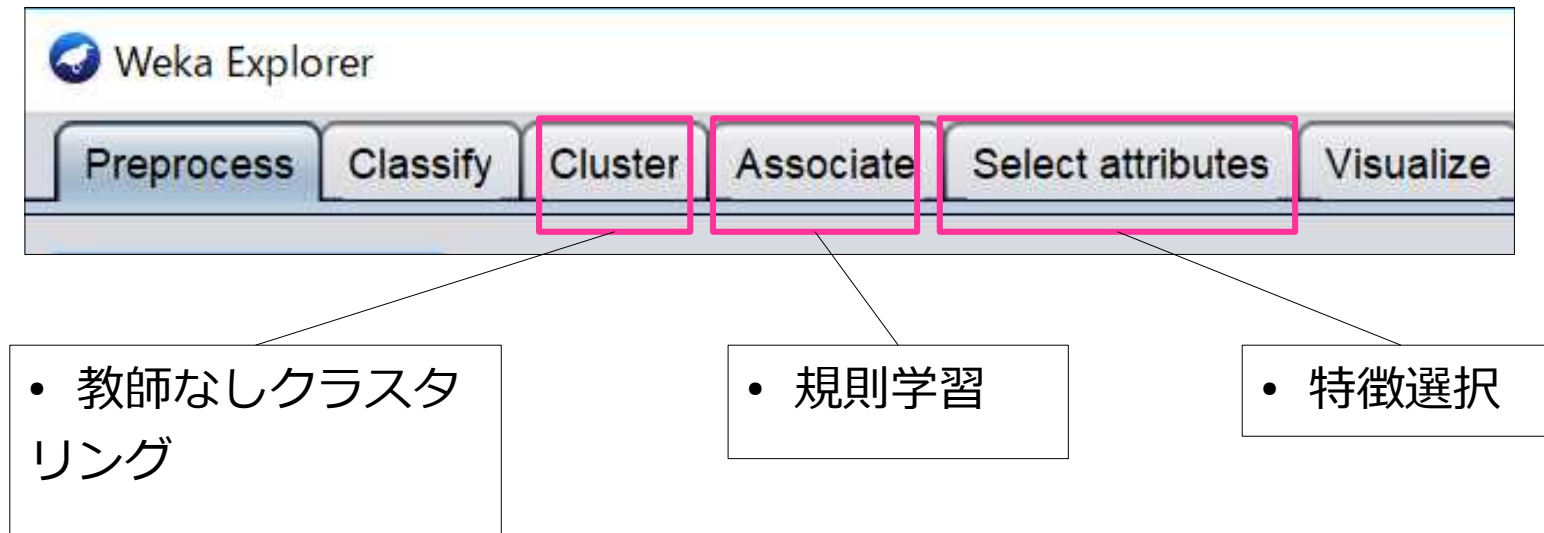
- 識別

- 100 以上の識別アルゴリズムの実装
- 学習の設定
- ハイパーパラメータの設定
- 学習結果の評価

- 可視化

- データの 2 次元プロット

# Explorer での操作



# 前処理 (Preprocess)

- 読み込み可能なデータ形式
  - ARFF (Attribute Relationship File Format) 形式
  - ヘッダ部とデータ部で構成
    - ヘッダ部
      - @relation : データ集合の名前 (ファイル名と同じでよい)
      - @attribute : 特徴の各次元の名前とデータの型を宣言
    - データ部
      - @data 以降に 1 行 1 件のデータを CSV 形式で記述
      - 各特徴・クラスラベルはカンマ区切り

# 前処理 (Preprocess)

- アヤメの分類データ (iris)



setosa



versicolor



virginica

```
% 1. Title: Iris Plants Database
@RELATION iris

@ATTRIBUTE sepallength    REAL
@ATTRIBUTE sepalwidth     REAL
@ATTRIBUTE petallength    REAL
@ATTRIBUTE petalwidth     REAL
@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA
5.1, 3.5, 1.4, 0.2, Iris-setosa
4.9, 3.0, 1.4, 0.2, Iris-setosa
...
7.0, 3.2, 4.7, 1.4, Iris-versicolor
6.4, 3.2, 4.5, 1.5, Iris-versicolor
...
6.3, 3.3, 6.0, 2.5, Iris-virginica
5.8, 2.7, 5.1, 1.9, Iris-virginica
...
```

データセット名

特徴名と型

萼・花びらの  
長さ・幅

アヤメの  
種類

これ以降、1行に1事例  
(ExcelのCSV形式と同じ)



# 前処理 (Preprocess)

- 特徴抽出後のデータを読み込む
- いくつかの特徴の操作（フィルタの適用）が可能

The screenshot shows the Weka Explorer application window. The interface is divided into several sections: a top menu bar with tabs like Preprocess, Classify, Cluster, etc.; a toolbar with buttons like Open file..., Open URL..., Open DB..., Generate..., Undo, Edit..., and Save...; a Filter section with a 'Choose' button; a Current relation section showing 'Relation: iris' and 'Instances: 150'; an Attributes section with a list of attributes (sepalength, sepalwidth, petallength, petalwidth, class); a Selected attribute section showing statistics for 'sepalength'; a Class section with a dropdown menu; and a Status section at the bottom.

Annotations with arrows pointing to specific parts of the interface:

- 読み込み** (Loading): Points to the **Open file...** button.
- フィルタ** (Filter): Points to the **Choose** button in the Filter section.
- データの全体像** (Overall view of the data): Points to the **Current relation** section, specifically the 'Relation: iris' and 'Instances: 150' text.
- 分析対象の特徴（属性）の選択** (Selection of features (attributes) for analysis): Points to the list of attributes in the **Attributes** section.
- データの表示** (Data display): Points to the **Edit...** button.
- 選択された特徴の分析** (Analysis of the selected feature): Points to the **Selected attribute** section, specifically the statistics table for 'sepalength'.

**Selected attribute statistics (sepalength):**

| Statistic | Value |
|-----------|-------|
| Minimum   | 4.3   |
| Maximum   | 7.9   |
| Mean      | 5.843 |
| StdDev    | 0.828 |

**Class: class (Nom)**

**Visualize All**

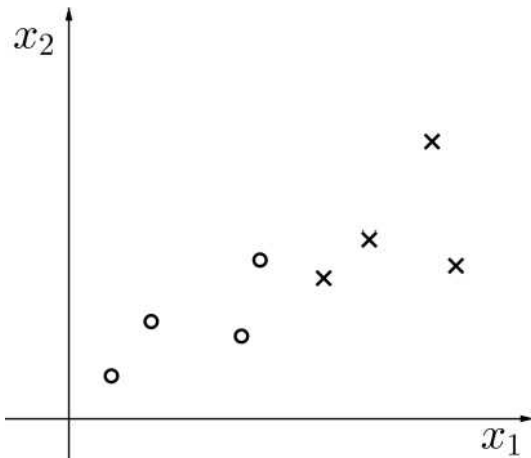
**Status:** OK

## 2.1.2 前処理

- 分析
  - 主成分分析（次元削減）
    - データの散らばりをできるだけ保存する低次元空間へ写像
    - データの可視化に有効
- データの標準化
  - すべての次元を平均 0、分散 1 にそろえる
  - 各次元に対して平均値を引き、標準偏差で割る

$$x'_i = \frac{x_i - m_i}{\sigma_i} \quad m_i, \sigma_i : \text{軸 } i \text{ の平均、標準偏差}$$

# 主成分分析の考え方



共分散行列 $\Sigma$ の計算

$\bar{x}_1, \bar{x}_2$  : 平均値、 $N$  : データ数

対角成分は分散、  
非対角成分は相関を表す

$$\Sigma = \frac{1}{N} \begin{pmatrix} \sum (x_1 - \bar{x}_1)^2 & \sum (x_1 - \bar{x}_1)(x_2 - \bar{x}_2) \\ \sum (x_1 - \bar{x}_1)(x_2 - \bar{x}_2) & \sum (x_2 - \bar{x}_2)^2 \end{pmatrix}$$

$\Sigma$ は

半正定値(→固有値がすべて0以上の実数)

対称行列(→固有ベクトルが実数かつ直交)

であるので、以下のように分解できる

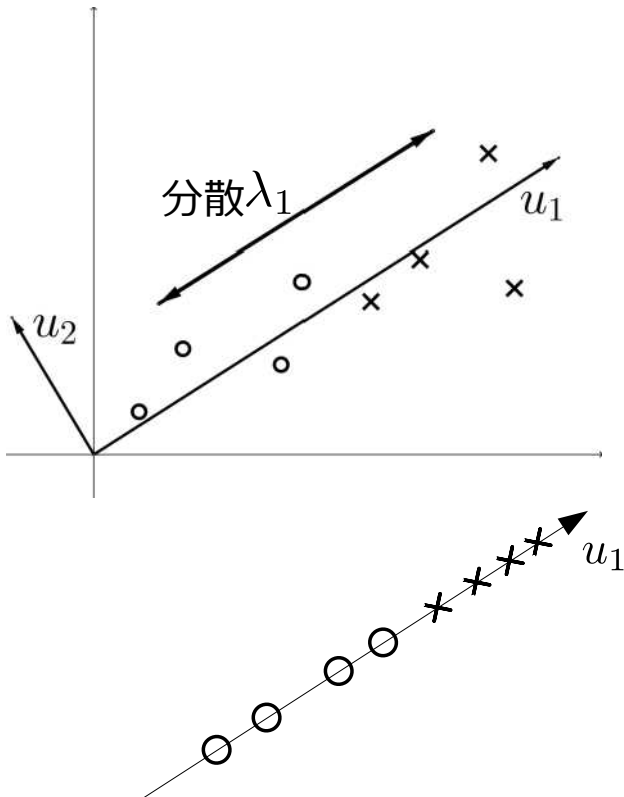
$$\Sigma' = U^T \Sigma U = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

$\lambda$ は固有値の大きい順、 $U$ は対応する  
固有ベクトル $U_1, U_2$ を並べたもの

$\lambda_1$ に対応する固有ベクトル $U_1$ で  
2次元データを1次元に射影

$$u_1 = U_1^T x$$

$$\text{寄与率} = \frac{\lambda_1}{\lambda_1 + \lambda_2}$$



# 前処理 (Preprocess)

- 標準化

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize | Collective | Forecast

Open file... | Open URL... | Open DB... | Generate... | Undo | Edit... | Save...

適用

選択

Filter

Choose Standardize Apply Stop

Current relation

Relation: iris-weka.filters.unsupervised.attribu... Attributes: 5  
Instances: 150 Sum of weights: 150

Attributes

All | None | Invert | Pattern

| No. | Name   |
|-----|--|
| 1   | <input checked="" type="checkbox"/> sepalength |
| 2   | <input type="checkbox"/> sepalwidth            |
| 3   | <input type="checkbox"/> petallength           |
| 4   | <input type="checkbox"/> petalwidth            |
| 5   | <input type="checkbox"/> class                 |

Remove

Status

OK

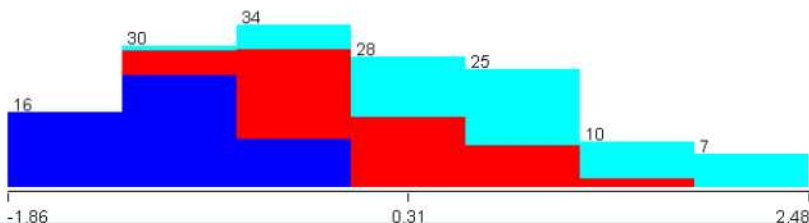
Selected attribute

Name: sepalength Type: Numeric  
Missing: 0 (0%) Distinct: 35 Unique: 9 (6%)

| Statistic | Value  |
|-----------|--------|
| Minimum   | -1.864 |
| Maximum   | 2.484  |
| Mean      | -0     |
| StdDev    | 1      |

平均 0  
標準偏差 1

Class: class (Nom) Visualize All



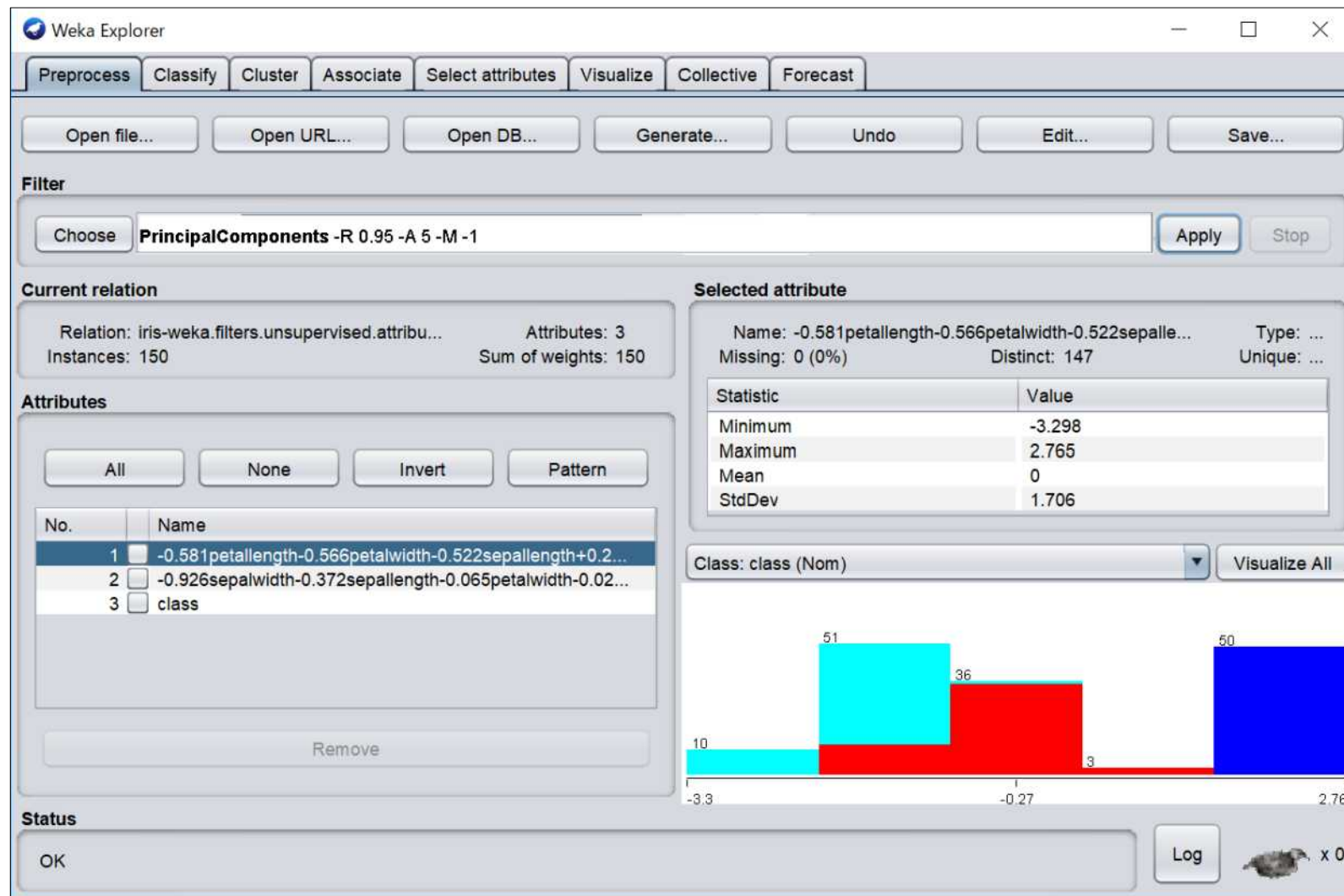
The histogram shows the distribution of sepalength values. The x-axis ranges from -1.86 to 2.48. The y-axis represents frequency. The distribution is bimodal, with peaks around -1.86 and 0.31. The bars are colored blue and red.

| Bin Range      | Frequency |
|----------------|-----------|
| -1.86 to -1.41 | 16        |
| -1.41 to -0.96 | 30        |
| -0.96 to -0.51 | 34        |
| -0.51 to -0.06 | 28        |
| -0.06 to 0.39  | 25        |
| 0.39 to 0.84   | 10        |
| 0.84 to 1.29   | 7         |

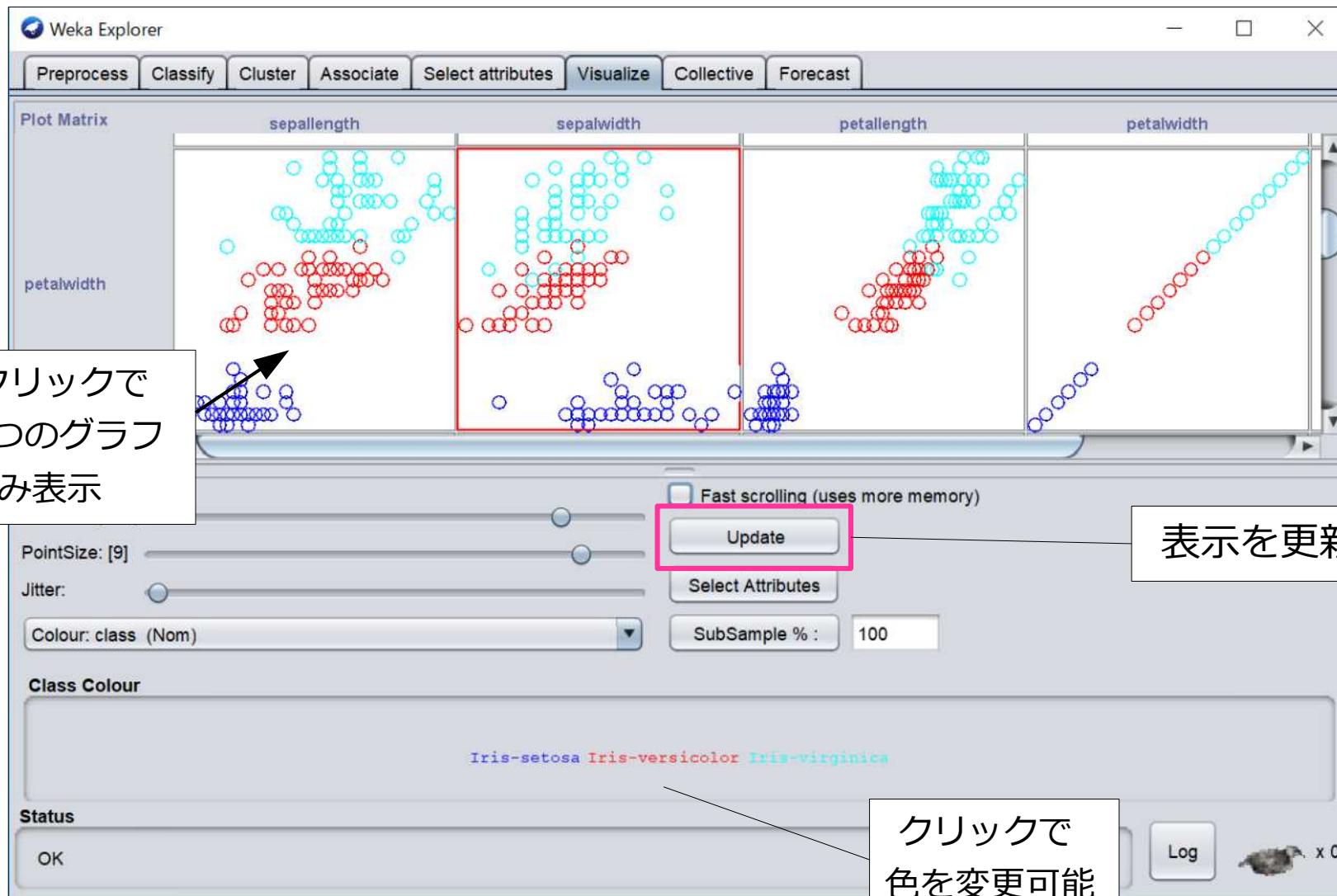
Log x 0

# 前処理 (Preprocess)

- 主成分分析
  - iris データ (4 次元特徴) を 2 次元に



# データのプロット (Visualize)



# データのプロット (Visualize)

- 1つのグラフのみ表示

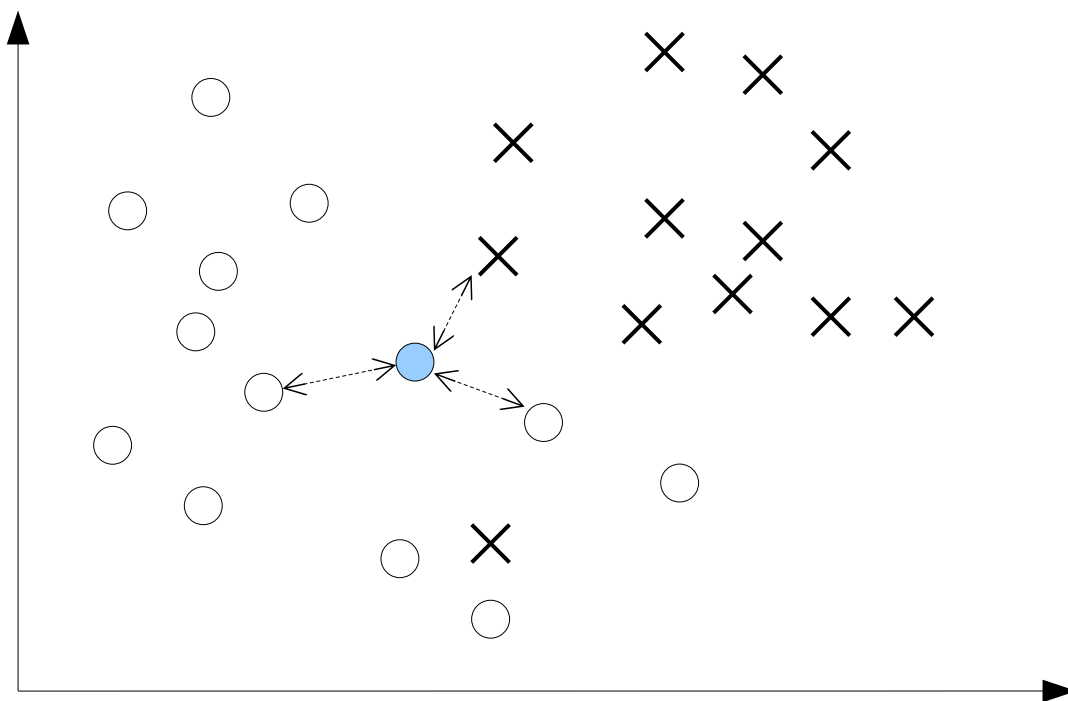


x 軸、y 軸、色の  
基準が選べる

クリックで  
色を変更可能

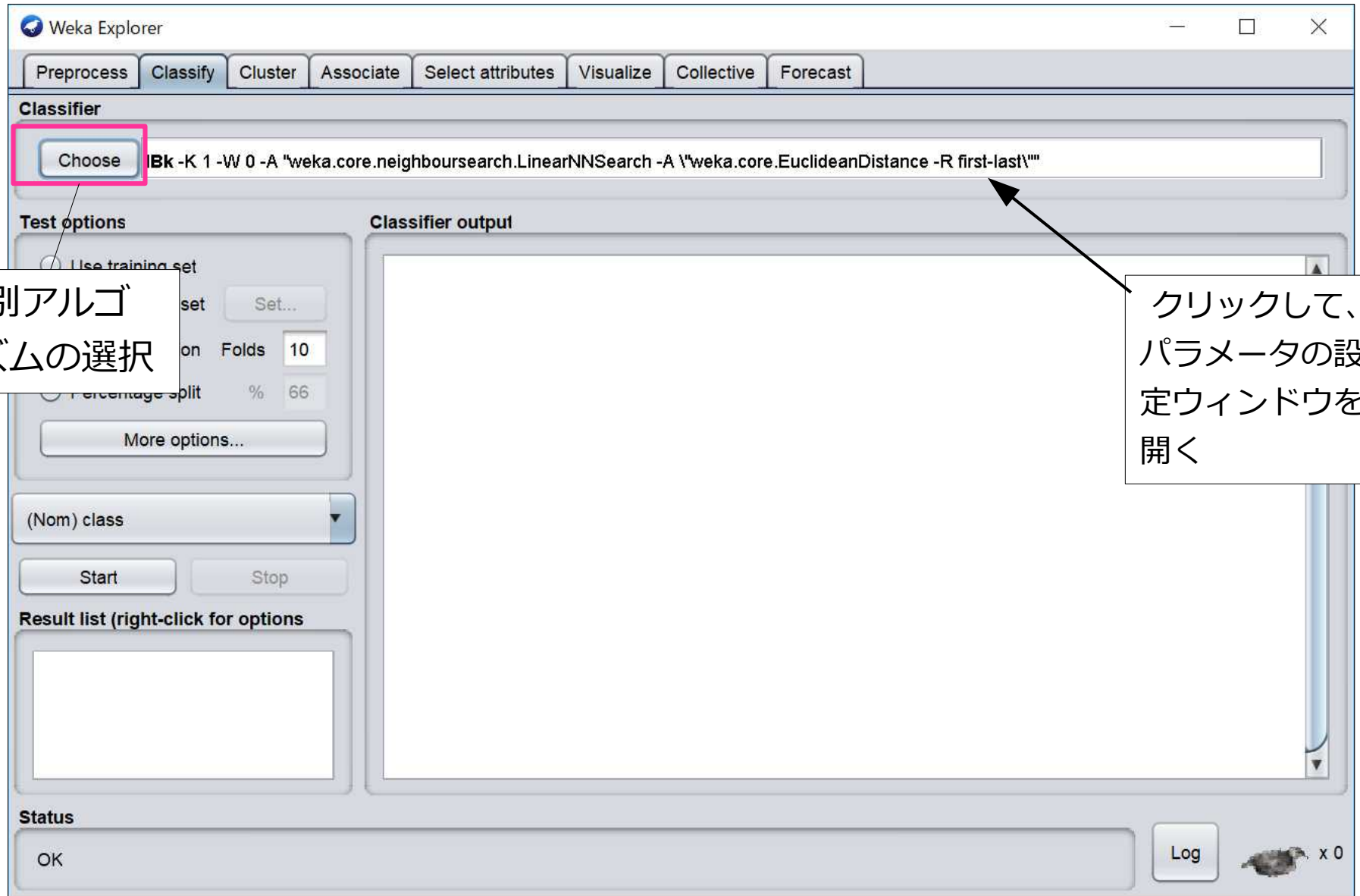
## 2.1.4 学習 k-NN 法

- NN (Nearest Neighbor: 最近傍) 法
  - 識別したいデータと最も近い事例を求め、その事例の属するクラスを識別結果とする (1-NN 法)
  - k 番目までの近い事例を求め、多数決を採るのが k-NN 法





# 識別器の学習 (Classify)

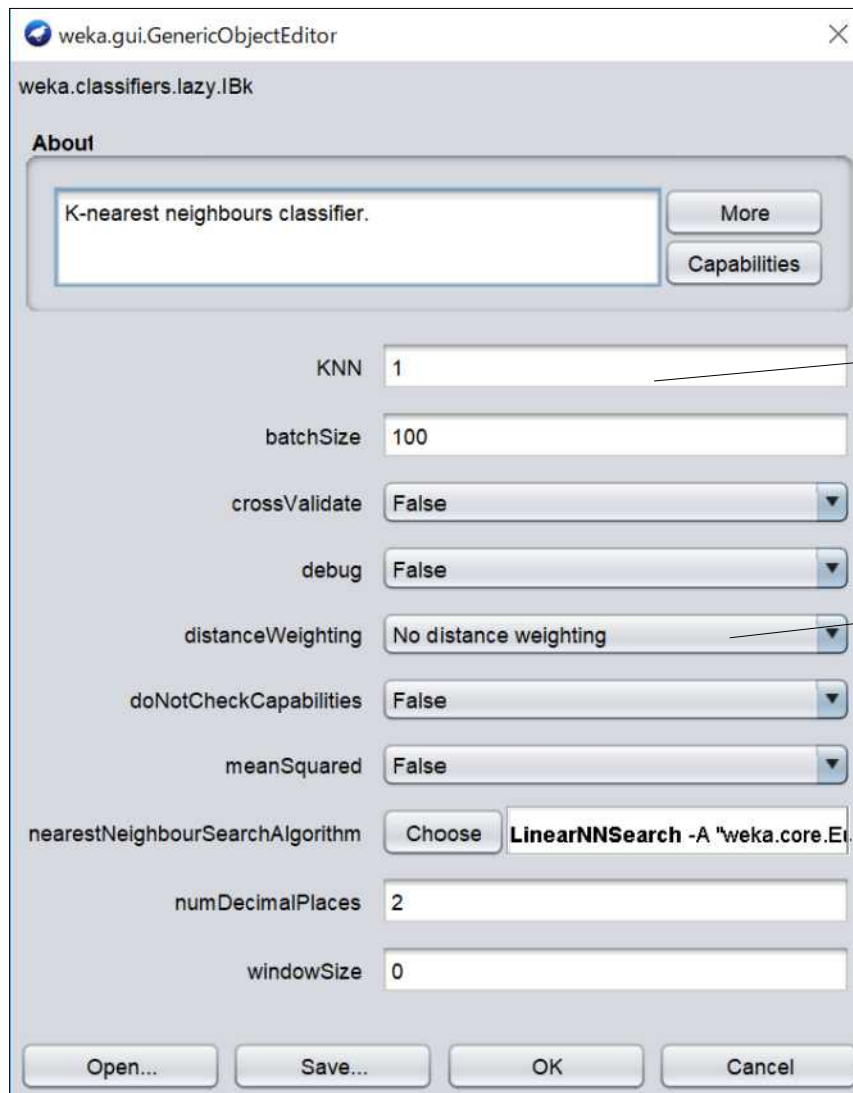


識別アルゴ  
リズムの選択

クリックして、  
パラメータの設  
定ウィンドウを  
開く

# 識別器の学習 (Classify)

- IBk (k-NN 法) のパラメータ



The screenshot shows the 'weka.gui.GenericObjectEditor' window for the 'weka.classifiers.lazy.IBk' classifier. The 'About' section describes it as a 'K-nearest neighbours classifier.' with 'More' and 'Capabilities' buttons. The main parameter list includes:

- KNN: 1
- batchSize: 100
- crossValidate: False
- debug: False
- distanceWeighting: No distance weighting
- doNotCheckCapabilities: False
- meanSquared: False
- nearestNeighbourSearchAlgorithm: Choose LinearNNSearch -A "weka.core.Et...
- numDecimalPlaces: 2
- windowSize: 0

Buttons at the bottom: Open..., Save..., OK, Cancel.

k

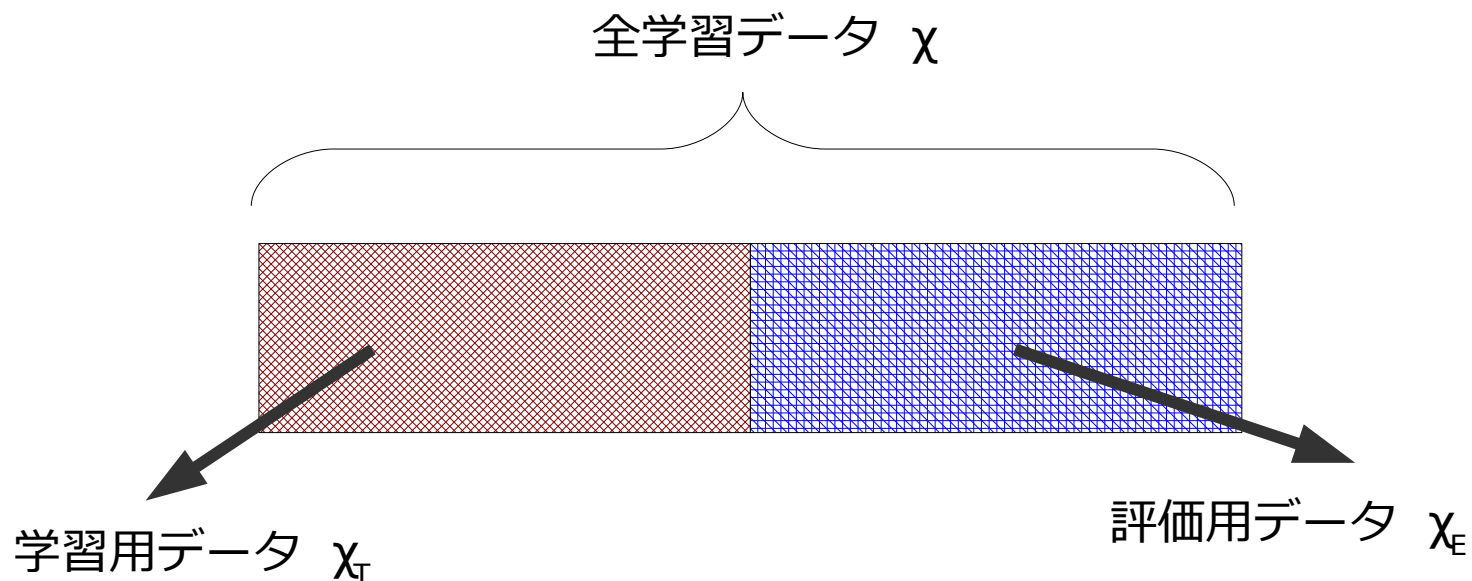
距離による重み付けの有無

## 2.1.3 評価基準の設定

- 分割学習法
  - データの半分を学習用、残りの半分を評価用とする
  - ハイパーパラメータを調整する場合は、学習用・検証用・評価用に分ける
- 交差確認法
  - データを  $m$  個の集合に分割し、 $m-1$  個の集合で学習、残りの  $1$  個の集合で評価を行う
  - 評価する集合を入れ替え、合計  $m$  回評価を行う
  - 分割数をデータ数とする場合を一つ抜き法とよぶ

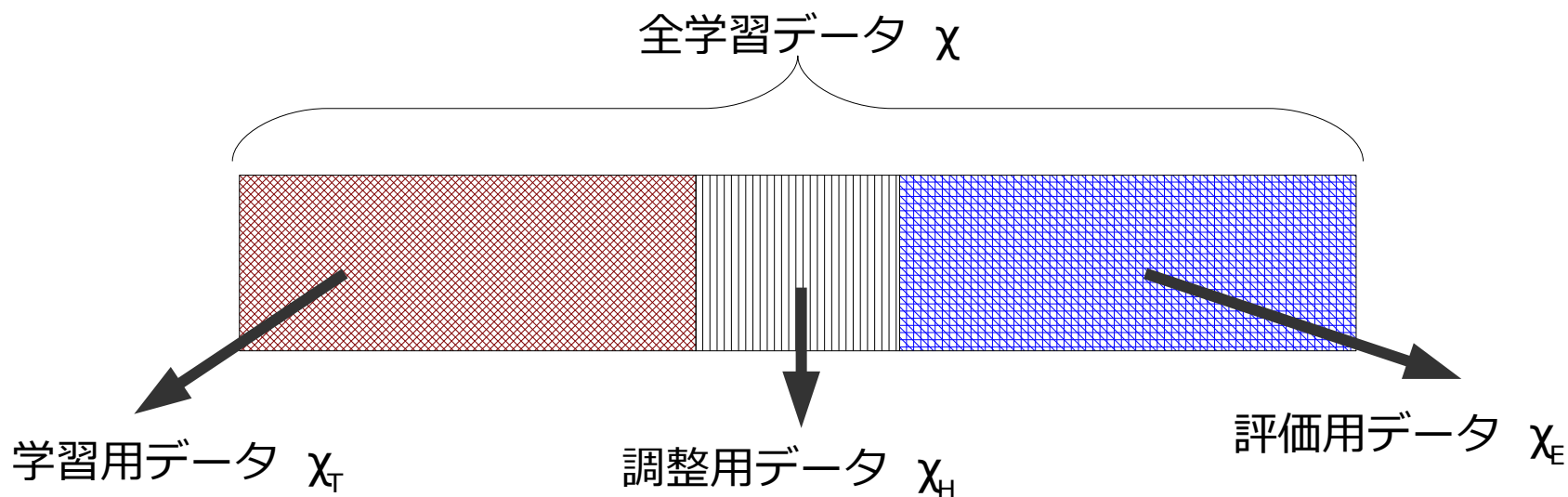
## 2.1.3 評価基準の設定

- 分割学習法（１）
  - 全学習データ  $\chi$  を学習用データ集合  $\chi_T$  と評価用データ集合  $\chi_E$  に分割する
  - $\chi_T$  を用いて識別機を設計し、 $\chi_E$  を用いて誤識別率を推定する



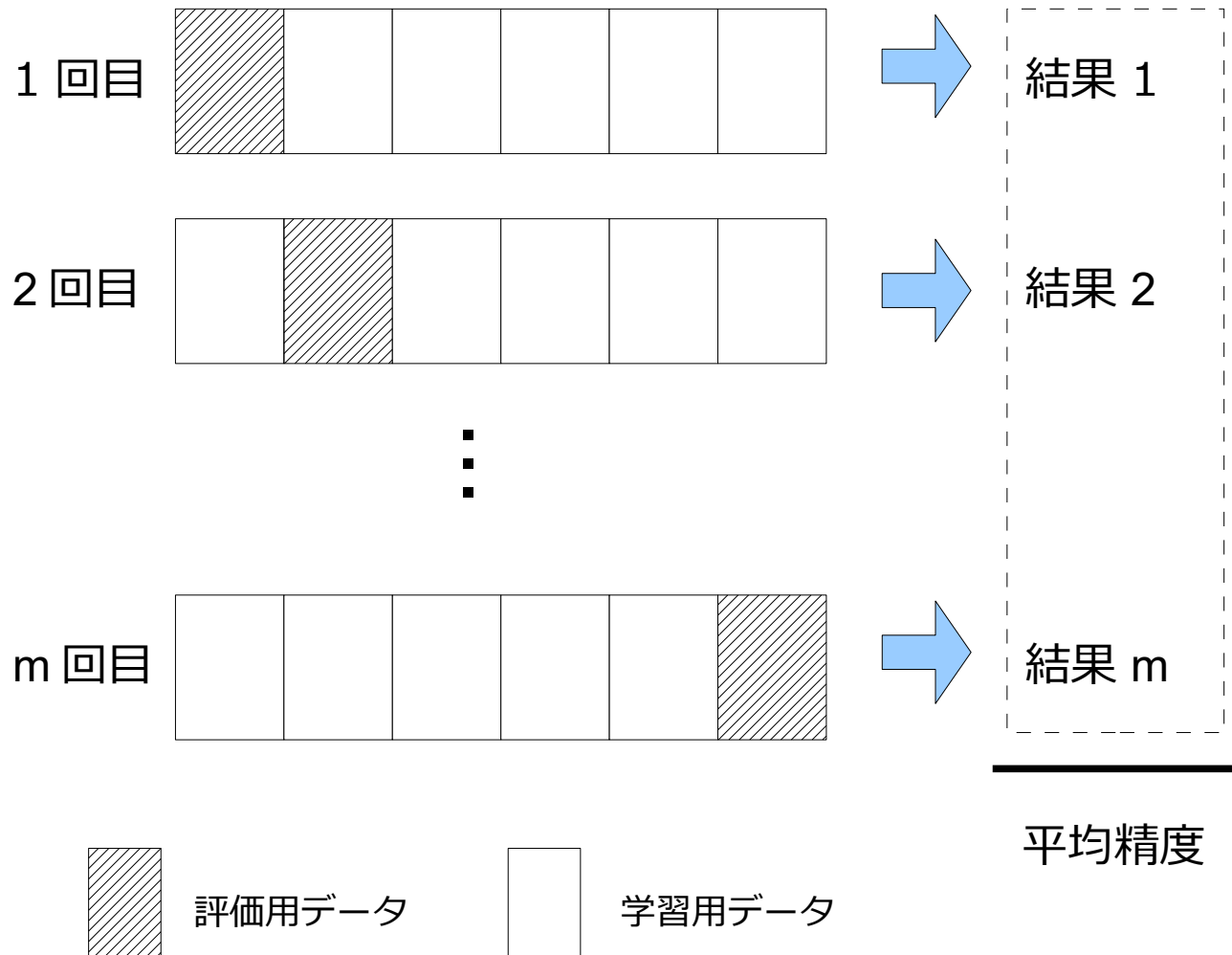
## 2.1.3 評価基準の設定

- 分割学習法（2）
  - 全学習データ  $\chi$  を学習用データ集合  $\chi_T$ 、調整用データ集合  $\chi_H$ 、評価用データ集合  $\chi_E$  に分割する
  - $\chi_T$  を用いて識別機を設計、 $\chi_H$  を用いてハイパーパラメータを調整、 $\chi_E$  を用いて誤識別率を推定する



## 2.1.3 評価基準の設定

- 交差確認法



# 識別器の学習 (Classify)

- 評価法の設定

学習データを使  
って評価

分割学習法

交差確認法

Test options

☐ Use training set

☐ Supplied test set

☒ Cross-validation Folds

☐ Percentage split %

データ分割数

# 識別器の学習 (Classify)

- 学習結果の見方

=== Summary ===

|                                  |           |
|----------------------------------|-----------|
| Correctly Classified Instances   | 14        |
| Incorrectly Classified Instances | 1         |
| Kappa statistic                  | 0.9167    |
| Mean absolute error              | 0.1051    |
| Root mean squared error          | 0.1645    |
| Relative absolute error          | 31.4161 % |
| Root relative squared error      | 39.3051 % |
| Total Number of Instances        | 15        |

...

=== Confusion Matrix ===

| a | b | c | d | e | <-- classified as |
|---|---|---|---|---|-------------------|
| 3 | 0 | 0 | 0 | 0 | a = a             |
| 0 | 3 | 0 | 0 | 0 | b = i             |
| 0 | 0 | 3 | 0 | 0 | c = u             |
| 0 | 0 | 0 | 3 | 0 | d = e             |
| 1 | 0 | 0 | 0 | 2 | e = o             |

正解率

93.3333 %  
6.6667 %

縦方向が正解、横方向が予測  
対角成分が正解数



# 結果の可視化

- 混同行列

|     | 予測+                 | 予測-                 |
|-----|---------------------|---------------------|
| 正解+ | true positive (TP)  | false negative (FN) |
| 正解- | false positive (FP) | true negative (TN)  |

- 正解率  $Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$

- 精度  $Precision = \frac{TP}{TP + FP}$

- 再現率  $Recall = \frac{TP}{TP + FN}$

- F 値  $F\text{-measure} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

正解の割合  
クラスの出現率に  
偏りがある場合は不適

正例の判定が  
正しい割合

正しく判定された  
正例の割合

↑  
トレードオフ  
↓

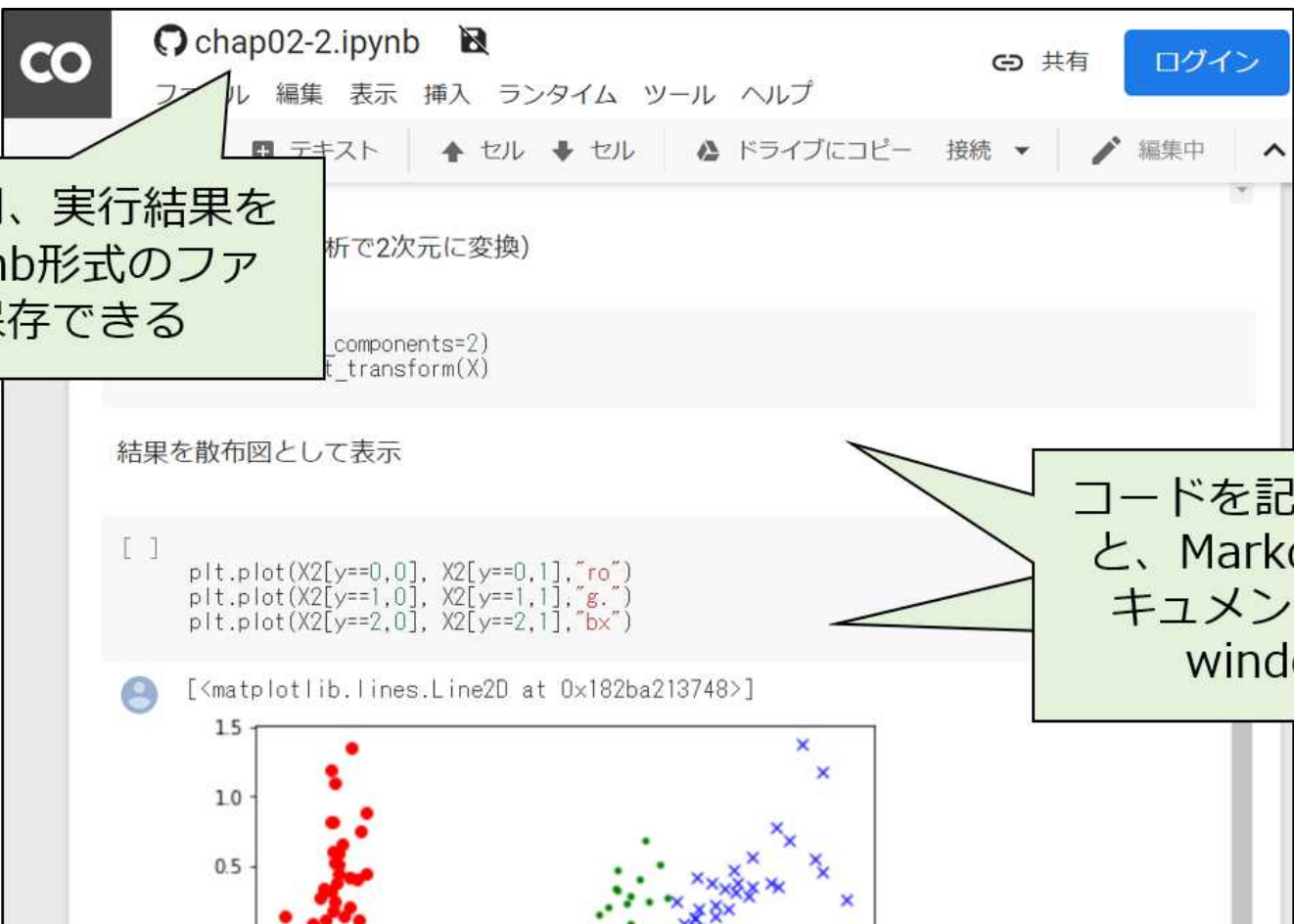
精度と再現率の  
調和平均

## 2.2 Python による機械学習

- プログラミング言語 Python
  - オブジェクト指向スクリプト言語
  - 特徴
    - 動的型付け・インデントによるブロック化・デフォルト引数を用いた関数呼び出し
- 機械学習に関連する多くのライブラリが充実
  - matplotlib : グラフ描画
  - pandas : データの読み込み・解析を支援
  - scikit-learn : 多くの機械学習アルゴリズム
  - tensorflow : 深層学習

## 2.2 Python による機械学習

- Jupyter notebook
  - ブラウザで実行できる Python 開発環境



The screenshot shows a Jupyter Notebook interface with the title 'chap02-2.ipynb'. The top bar includes a '共有' (Share) button and a 'ログイン' (Login) button. The main area displays a code cell with the following Python code:

```
[ ]  
plt.plot(X2[y==0,0], X2[y==0,1], "ro")  
plt.plot(X2[y==1,0], X2[y==1,1], "g.")  
plt.plot(X2[y==2,0], X2[y==2,1], "bx")
```

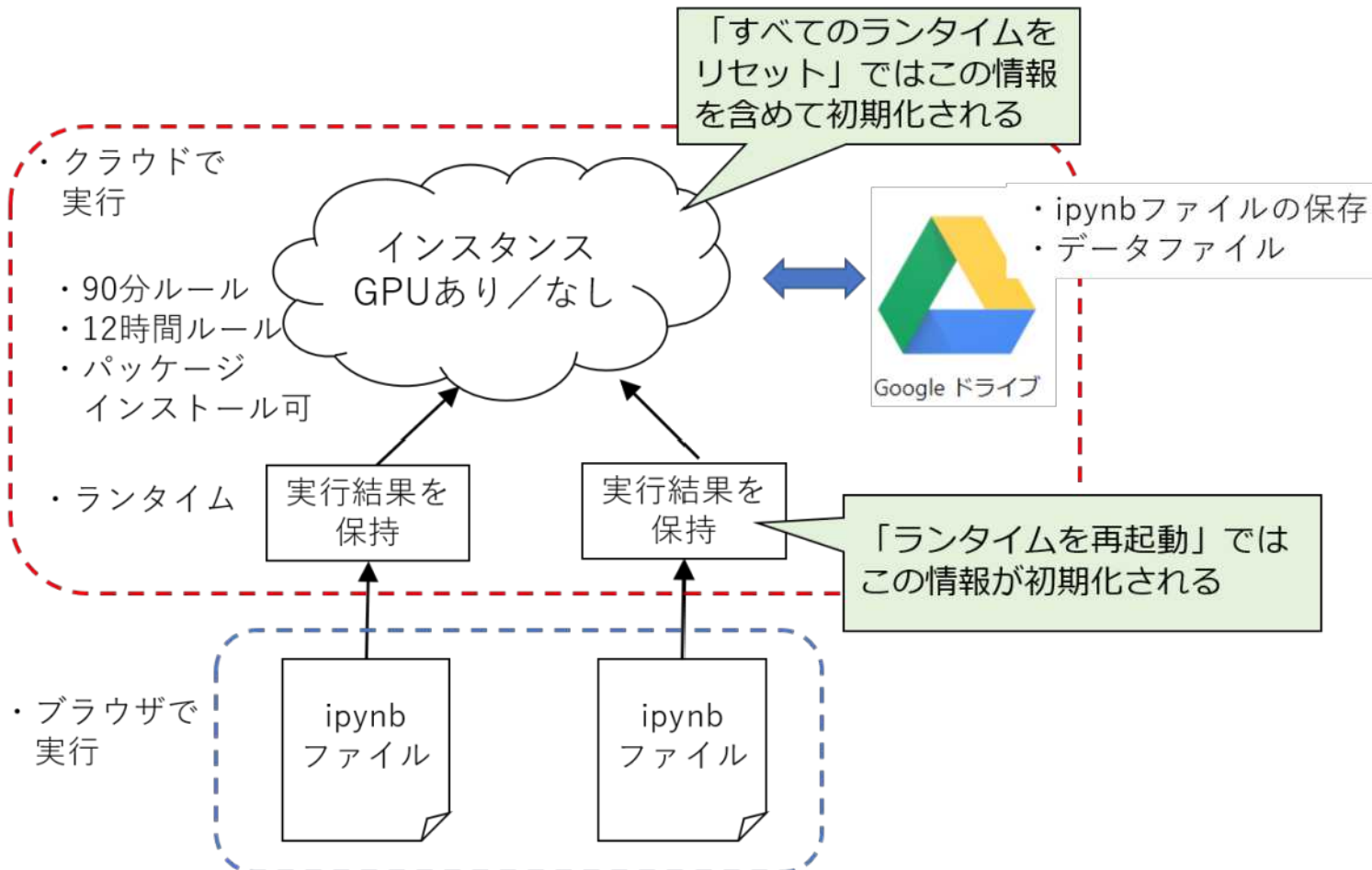
Below the code cell, the output shows a scatter plot with three distinct clusters of data points: red circles ('ro'), green dots ('g.'), and blue crosses ('bx'). The plot is titled '結果を散布図として表示' (Display results as a scatter plot).

Two callout boxes provide additional information:

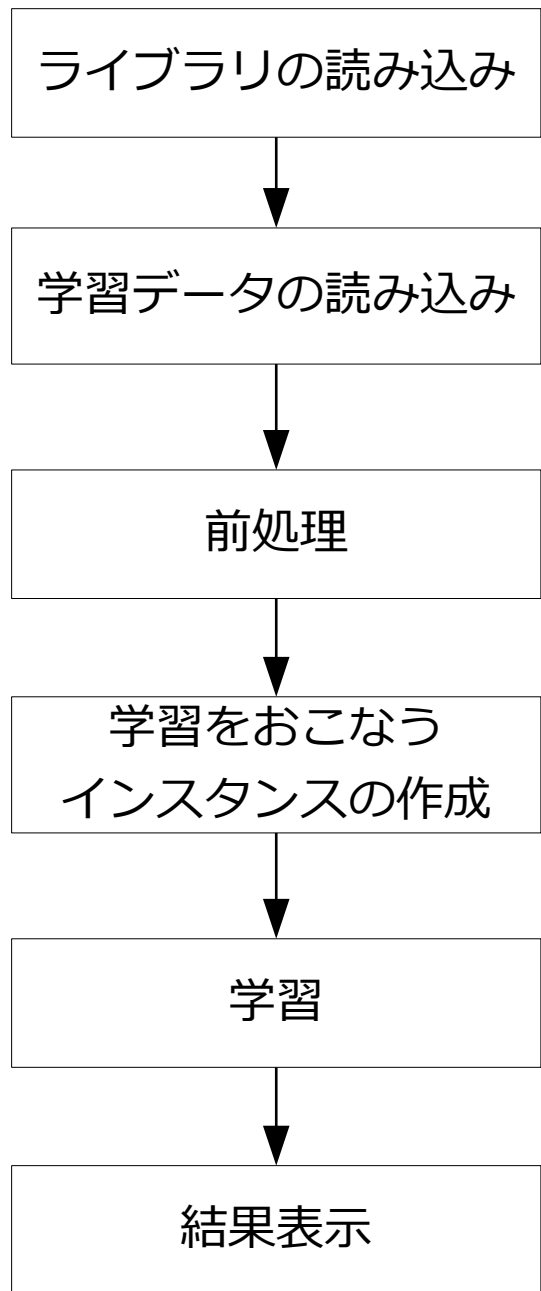
- Left Callout:** コード、説明、実行結果をまとめてipynb形式のファイルとして保存できる (Code, explanation, and execution results can be saved together as an ipynb format file).
- Right Callout:** コードを記述するwindowと、Markdown形式でドキュメントを記述するwindowがある (There are windows for describing code and windows for describing documents in Markdown format).

## 2.2 Python による機械学習

- Google Colaboratory
  - クラウドで実行される Jupyter notebook 環境
  - 機械学習関係のライブラリはインストール済み



## 2.2 Python による機械学習



- 組み込みデータは datasets パッケージを利用
- 外部データは pandas の read\_csv 等を利用

- 標準化 : scale
- 主成分分析 : PCA

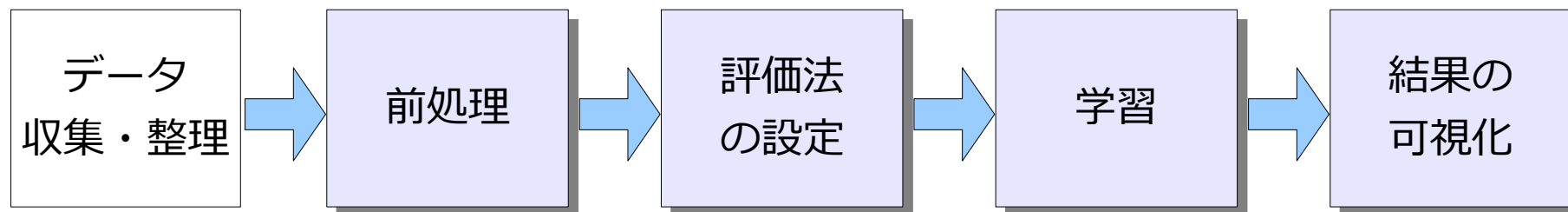
- 学習パラメータを与えてインスタンスを作成

- fit に学習データを与えて学習
- 分割学習法では predict で予測を得る
- 交差確認法では cross\_val\_score を実行

- 分割学習法では confusion\_matrix で混同行列を求める
- 交差確認法では、結果から平均・標準偏差などを求める

# まとめ

- 機械学習の基本的なプロセス



- データの前処理で有効な技法
  - 標準化、主成分分析
- 代表的なベースライン手法
  - k-NN 法（1-NN は最も近い事例を答とする）
- 結果の評価
  - 分割学習法、交差確認法
  - 正解率、精度、再現率、F 値