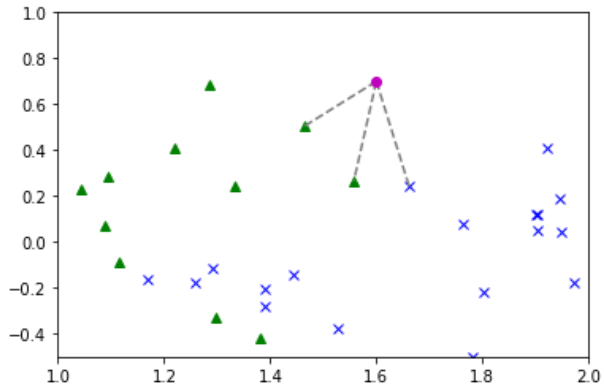
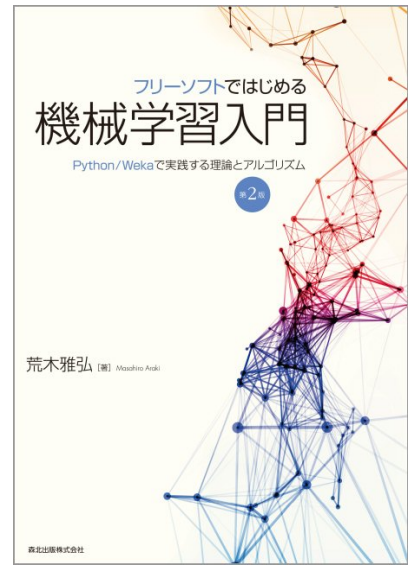
2. 機械学習の基本的な手順(Python 編)



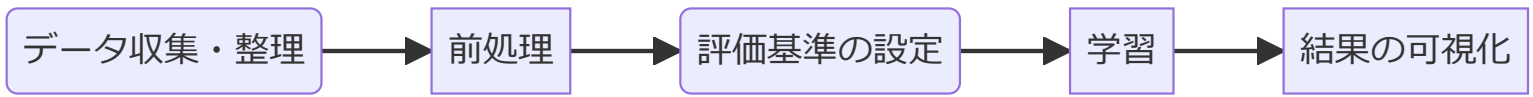
- 2.2.1 scikit-learn を用いた機械学習の手順
- 2.2.2 データの読み込み
- 2.2.3 前処理
- 2.2.4 評価基準の設定と学習
- 2.2.5 結果の表示



- 荒木雅弘:『フリーソフトではじめる機械学習入門(第2版)』(森北出版, 2018年)
- スライドとJupyter notebook
- サポートページ

2. 機械学習の基本的な手順

- 一般的な機械学習の手順
  - それぞれの手順に適したライブラリによる支援が可能

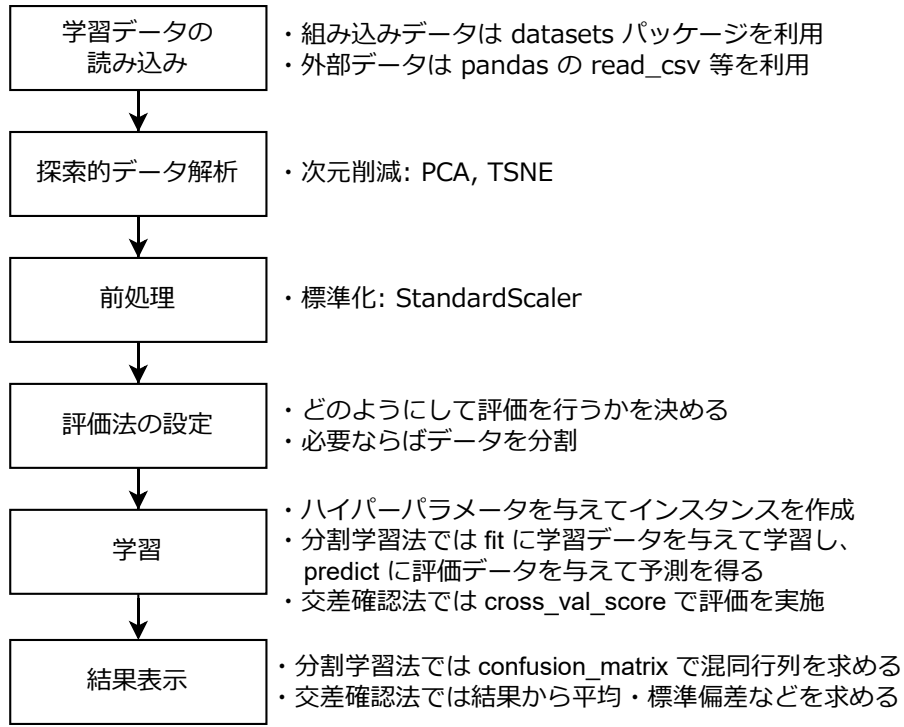


2.2.1 scikit-learn を用いた機械学習の手順

- 機械学習システムの開発に Python を使うメリット
  - データ処理や機械学習のパッケージが充実
    - numpy：多次元配列を効率よく扱う
    - scipy：高度な数値計算
    - 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, pytorch：深層学習
  - グラフ表示などの可視化が容易
    - matplotlib：グラフ描画
    - seaborn：統計データの可視化
  - Jupyter Notebook で実行手順を記録しながらコーディングが可能

2.2.1 scikit-learn を用いた機械学習の手順

- 機械学習の手順と使用するライブラリ・クラス・メソッド



2.2.1 scikit-learn を用いた機械学習の手順

- パッケージの読み込み
  - データの格納や基本的な操作に numpy は必須
  - 入力したデータの分析や前処理を行うには pandas を使う
  - データや結果の可視化を行うには matplotlib.pyplot を使う
  - scikit-learn はパッケージ全体ではなくクラスや関数を個別に指定

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report
```

2.2.2 データの読み込み

- サンプルデータ: iris
  - 3種類のアヤメ(setosa, versicolor, virginica)を萼(がく; sepal) の長さ・幅、花びら(petal)の長さ・幅の計4つの特徴から分類する
  - 各クラス50事例ずつで計150事例のデータ数
    - 冒頭の5事例

index	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

2.2.2 データの読み込み

- scikit-learn でのデータの持ち方
  - パターン行列：X
    - 全データの特徴ベクトルを列方向に並べたもの
    - iris データの場合は150事例、4特徴の150行4列の行列
  - 正解：y
    - 正解ラベルを整数値にしてデータの数だけ並べたもの
    - iris データの場合は150個の数字(0,1,2のいずれか)が並ぶベクトル

2.2.2 データの読み込み

- numpy の ndarray（n次元テンソル）として読み込む方法
  - load_iris 関数の戻り値は Bunch オブジェクト
    - 特徴ベクトル, 正解データ, 特徴名, データの説明などのさまざまな情報を属性として持つ

```
iris = load_iris()
X = iris.data
y = iris.target
```

    - X や y は ndarray なので、scikit-learn の学習データとして用いることができる
- pandas の DataFrame および Series として読み込む方法
  - 実データでは、異常値・欠損値・記述ミス・不要な特徴の混入などへの対処が必要

    → このような用途では numpy では不十分なので、pandas を使う
  - データロード関数の引数：as_frame=True

```
iris = load_iris(as_frame=True)
```

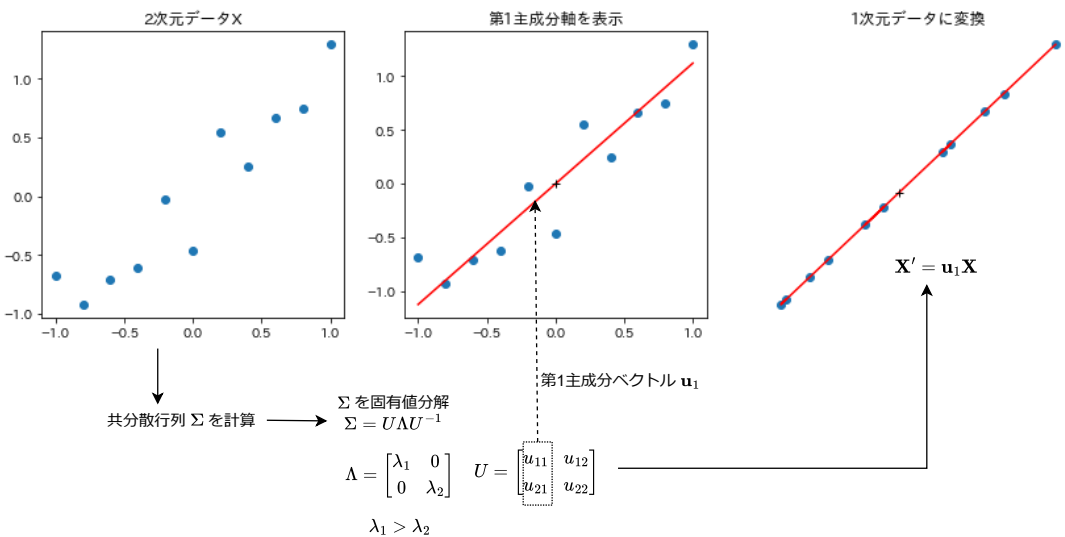
探索的データ解析

- 探索的データ解析（EDA; Explanatory Data Analysis）とは
  - 設定した問題に対して、対象としているデータが解決に適したものであるか、また機械学習を適用する前にどの程度の整理が必要かを調べること
  - 手法
    - データの統計的性質を分析
    - データを可視化
- pandas: データ分析・操作
  - 統計的分析: describe, hist, …
  - 異常値・欠損値(NA)処理: query, dropna, fillna, …
- matplotlib: グラフ表示

探索的データ解析

- 主成分分析(PCA)
  - 高次元空間上のデータの散らばり方をできるだけ保存する低次元空間への写像を求める
  - データの次元削減に有効

```
pca = PCA(n_components=1)  ## n_components: 削減後の次元数
X2 = pca.fit_transform(X)
```



探索的データ解析

- t-SNE
  - 高次元空間でのデータの類似度を反映した低次元空間への写像を求める
  - データの可視化に有効

```
tsne = TSNE(perplexity=5)  ## perplexity: 考慮する近傍のデータ数（5〜50程度の値で全データ数が多いほど大きく）
X3 = tsne.fit_transform(X)
```

t-SNEの考え方

- 元の高次元空間
  - どの範囲のデータを類似度計算の対象とみなすかをパラメータ perplexity で与える
  - データ $\boldsymbol{x}_i$ と $\boldsymbol{x}_j$ の類似度を、$\boldsymbol{x}_i$ の近傍として $\boldsymbol{x}_j$ を選択する条件付き確率 p_{ij} とする
    - p_{ij}: 平均を$\boldsymbol{x}_i$、分散をperplexityに基づいて求めたσ^2とする正規分布に基づいて計算
- 削減後の低次元空間
  - データ $\boldsymbol{y}_i$ と $\boldsymbol{y}_j$ の類似度 q_{ij} を、自由度1のt分布に基づいて計算
    - t分布は正規分布よりも値の大きい範囲が広い
- 最適化
  - p_{ij}, q_{ij} 両分布間の距離(KL-divergence)を最小化するように $Y = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$ の位置を逐次更新

$$KL(P, Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

2.2.3 前処理

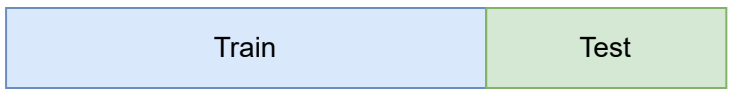
- 特徴のスケーリング
  - 特徴の各次元のスケールが著しく異なると、特徴の扱いが不公平になる
  - 標準化：すべての次元を平均0、分散1に揃える
    - 各次元（軸）に対して平均値を引き、標準偏差で割る

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

```
X_scaled = StandardScaler().fit_transform(X)
```

2.2.4 評価基準の設定と学習

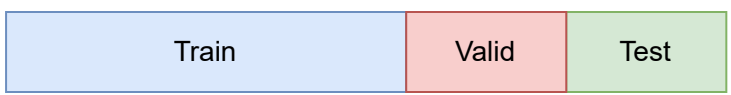
- 分割学習法(データ数が多いとき)
  - データを学習用と評価用に適切な割合で分ける



  - 実験の再現性を確保するためにはrandom_stateを固定しておく

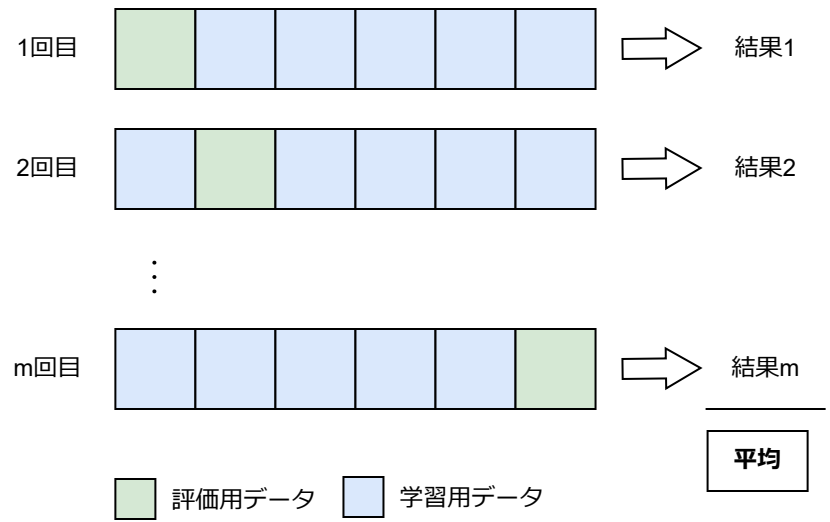
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.33, random_state=7)
```

  - 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



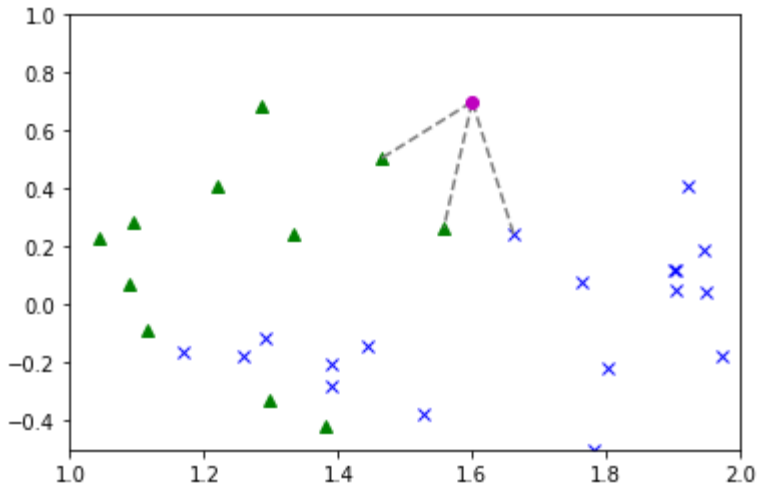
2.2.4 評価基準の設定と学習

- 交差確認法(データ数が少ないとき)
  - 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
  - 学習用データで交差確認法によりハイパーパラメータ調整を行い、評価用データで評価してもよい



2.2.4 評価基準の設定と学習

- k-NN法
  - 識別したいデータの近傍のk個の学習データを探し、属するクラスの多数決で識別



2.2.4 評価基準の設定と学習

- k-NN法のパラメータ
  - 近傍として探索するデータ数: k
    - k が1の場合にもっとも複雑な境界となり、汎化性能は低くなる傾向がある
    - k が増えるに従って境界は滑らかになるが、大き過ぎると識別性能が低下する
  - 距離尺度
    - 通常はユークリッド距離
    - 値を持つ次元が少ない場合はマンハッタン距離
  - 探索方法
    - 入力と全データとの距離を計算してソート
    - データが多い場合は事前にデータを木構造化

2.2.4 評価基準の設定と学習

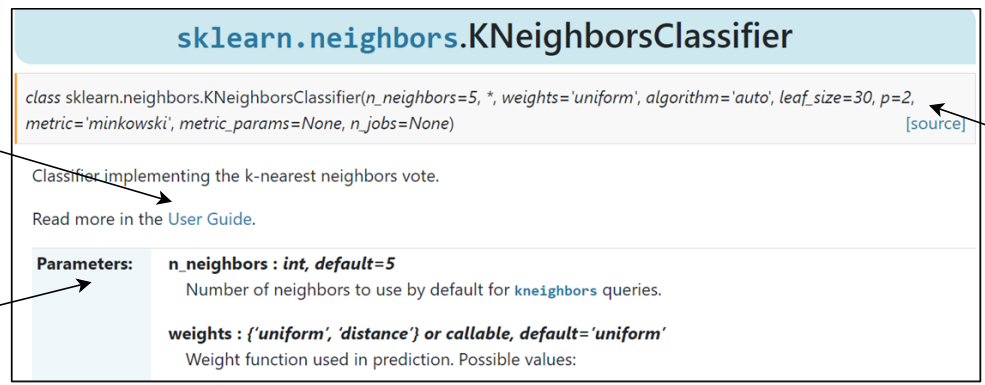
- 学習を行うインスタンスの生成
  - モデルの構成に関するパラメータ(ハイパーパラメータ)は、インスタンス生成時に与える
    - 詳しくはAPIドキュメントを参照

```
clf = KNeighborsClassifier(n_neighbors=3)
```

アルゴリズムの詳細な説明や、事例が記載されているページへのリンク

インスタンス生成時に指定するパラメータの説明

デフォルト引数の値が示されている。*以降は必ずキーワード引数で指定する。



sklearn.neighbors.KNeighborsClassifier

class sklearn.neighbors.KNeighborsClassifier(*n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None*) [source]

Classifier implementing the k-nearest neighbors vote.

Read more in the User Guide.

Parameters:

n_neighbors : *int, default=5*
Number of neighbors to use by default for `kneighbors` queries.

weights : *{'uniform', 'distance'} or callable, default='uniform'*
Weight function used in prediction. Possible values:

2.2.5 結果の表示

- 学習したモデル
  - 式、木構造、ネットワークの重み、etc.
- 性能
  - 正解率、適合率、再現率、F値
  - グラフ
    - パラメータを変えたときの性能の変化
    - 異なるモデルの性能比較

2.2.5 結果の表示

- 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
  - 正解の割合。不均衡データには不適

    $$Accuracy = \frac{TP+TN}{TP+FN+FP+TN}$$
- 適合率
  - 正例の予測が正しい割合

    $$Precision = \frac{TP}{TP+FP}$$
- 再現率
  - 正しく予測された正例の割合

    $$Recall = \frac{TP}{TP+FN}$$
- F値
  - 適合率と再現率の調和平均

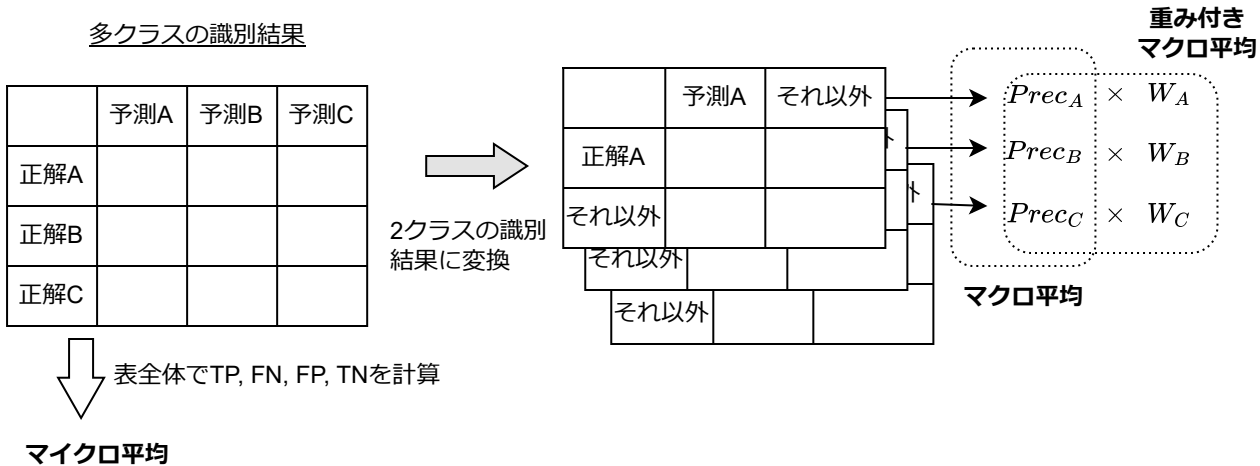
    $$F\text{-}measure = 2 \times \frac{Precision \times Recall}{Precision+Recall}$$

2.2.5 結果の表示

- 多クラス識別の評価法
  - マイクロ平均
    - クラスごとにTP, FN, FP, TNを求め、それらを足し合わせて評価する
    - マルチラベルの設定以外では適合率・再現率・F値がすべて正解率に一致する
  - マクロ平均
    - ひとつのクラスを正、残りのクラスを負とした混同行列を作成し、クラスごとの適合率や再現率を求め, 平均を計算する
    - すべてのクラスを平等に評価している
  - 重み付きマクロ平均
    - クラス毎の正解事例数を評価に反映させる

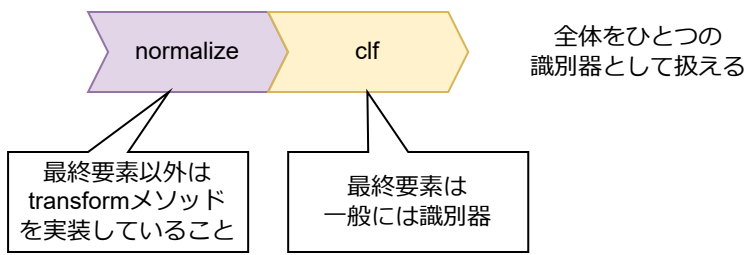
2.2.5 結果の表示

- 多クラス識別の評価法



パイプライン

- パイプラインとは
  - 複数の前処理と学習モジュールなど、連続した処理をパイプラインとして結合して、ひとつの識別器のインスタンスとみなせる
- パイプラインのメリット
  - 処理をカプセル化して実行を簡単にできる
  - ハイパーパラメータ調整を一度に行える
  - テストデータが混入していないことを保証できる



2.3 まとめ

- 機械学習の基本的な手順
  - 探索的データ解析
    - 統計的分析、可視化
  - 前処理
    - 標準化、次元削減
  - 評価基準の設定
    - 分割法、交差確認法
  - 学習
    - ハイパーパラメータ調整
  - 結果の可視化

実開発ではこれ以前のデータ収集・フォーマットの統一等の段階がもっとも時間がかかることも多い