ae-3.機械学習の基礎①

ー　教師あり学習編　ー

（AI演習）（全１５回）

https://www.kkaneko.jp/ai/ae/index.html

金子邦彦

人工知能（AI）の学習のための指針

1. **実践重視**： AIツールを実際に使用し、機能に慣れる

2. **エラーを恐れない**： 実行においては、エラーの発生の可能性がある．エラーを恐れず，むしろ学習の一部として捉えるポジティブさが大切．

3. **段階的学習**：基礎から応用へと段階的に学習を進め，AIの可能性を前向きに捉える

アウトライン

1. イントロダクション（機械学習）
2. 機械学習における回帰
3. 機械学習における分類
4. 決定木を用いた分類
5. 機械学習での学習と汎化

Python

- **Python** は多くの人々に利用されている**プログラミング言語**の１つ
- **読みやすさ，書きやすさ，幅広い応用範囲**が特徴

```
from keras.models import Sequentia
model = Sequential()
from keras.layers import Dense, Ac

model.add(Dense(units=64, input_di
model.add(Activation('relu'))
model.add(Dense(units=max(set(y_tr
model.add(Activation('softmax'))
model.compile(loss='sparse_categor
              optimizer='sgd',
              metrics=['accuracy']
model.fit(x_train, y_train, epochs
score=model.evaluate(x_test, y_tes
print(score)
model.predict(x_test)
model.summary()
```

```
poch 1/200
'3 [==============================] - 0s
 3200
 och 2/200
 3 [==============================] - 0s
 200
  ch 3/200
   [==============================] - 0s
  0
```

Python 言語が広く使用されている理由

文法のシンプルさ

- **Python** は，**直感的で読みやすい文法**
- 例えば、**print** で簡単に**出力**できる、**if** や **else** で**条件分岐**、**for** や **while** で**繰り返し（ループ）**

拡張性

- **多岐にわたる分野で利用が可能**
- 例えば、**関数やクラスを定義**するための **def** や **class**、**継承**や**オブジェクトの属性名と値**を操作するための **super** や **vars** などがある。

柔軟性

- **シンプルなスクリプトも、高度なプログラムも作成可能**
- **オブジェクト指向**の機能を持ち、**__init__** や **self** のようなキーワードを使用して**クラス**を利用できる。

trinket

- **Trinket** は**オンライン**の **Python、HTML** 等の**学習サイト**
- ブラウザで動作
- 有料の機能と無料の機能がある
- **自分が作成した Python プログラムを公開し、他の人に実行してもらうことが可能**（そのとき、書き替えて実行も可能）

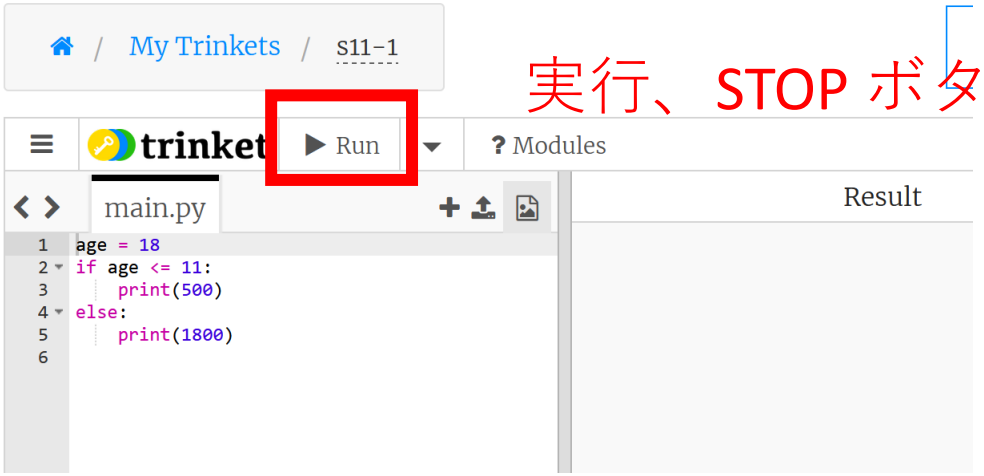
- **Python の標準機能**を登載、その他、次の外部ライブラリがインストール済み

matplotlib.pyplot, numpy, processing, pygal

trinket でのプログラム実行

- **trinket は Python, HTML などのプログラムを書き実行できる**サイト
- https://trinket.io/python/0fd59392c8

のように、違うプログラムには違う URL が割り当てられる



- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

3-1. イントロダクション

機械学習の基本

機械学習は，**コンピュータ**が**データ**を使用して**学習**することにより**知的能力を向上**させる技術

- **情報の抽出**：データからパターンや関係性を自動で見つけ出す能力を持つ
- **知的なタスクの実行**：予測，分類などの知的なタスクを実行

機械学習の３つの特徴

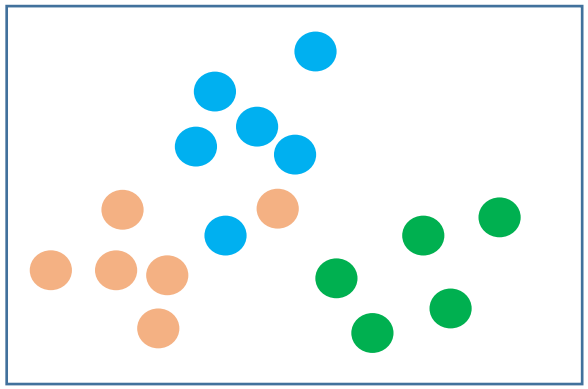
1. **情報の抽出**：データからパターンや関係性を自動で見つけ出す能力を持つ
2. **簡潔さ**：人間が設定していたルール等を，自動生成できる
3. **限界の超越**：従来の方法では困難だった課題も解決できる可能性がある

機械学習における訓練データの役割

- **訓練データ**は，**機械学習の学習に使用されるデータ**である．
- **データを用いた学習**により，コンピュータは分類や予測などの**知的な能力を獲得**する．

  例：画像分類では分類済みの大量データを使用する．

訓練データ



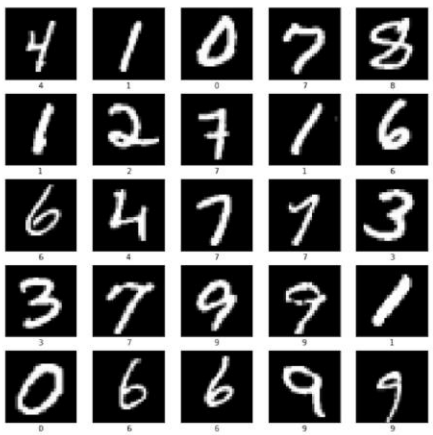
3種類に分類済み

学習

学習者

機械学習の学習プロセス

①データの準備：
目的に応じた訓練データを準備



画像 60000枚
（うち一部）

4 1 0 7 8
1 2 7 1 6
6 4 7 7 3
3 7 9 9 1
0 6 6 9 9

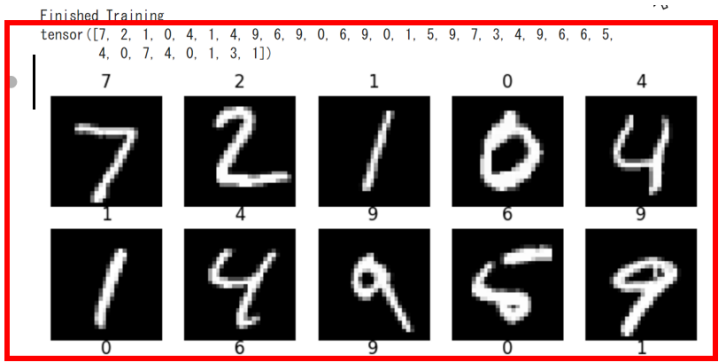
正解 60000個
（うち一部）

②学習の実行：
データを用いてパターンを学習.
モデルを構築

プログラム

データを用いて学習を行う．学習の結果、文字認識の能力を獲得.

③タスクの実行：
新しいデータを処理



機械学習まとめ

機械学習の特徴

- データを用いて知的能力を向上
- 自動でデータのパターンを抽出
- さまざまなタスクを自動実行

応用事例

画像理解、自然言語処理、予測など多数

機械学習の定義：

- **訓練データ**を用いて**学習**し、その結果として知的能力が向上
- **訓練データの追加**により、さらに**知的能力が向上**する可能性

機械学習の特徴と応用

- **能力の多様性**:

分類、検出、識別、認識（文字認識、画像の認識、音声認識）、予測、合成、翻訳、特徴抽出、ランク付け、情報推薦など、**さまざまな能力を持つ**

- **応用範囲の広さ**:

経済、金融、ロボット、医療、医学研究（遺伝子解析）など、様々な分野で問題解決に活用

機械学習は多様な能力を持ち、経済や金融から医療やロボットまで広範な応用が可能

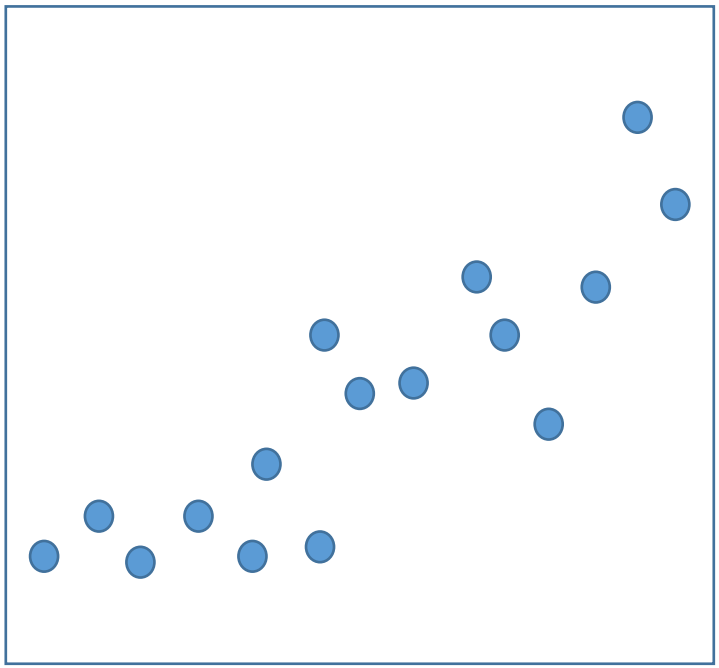
3-2. 回帰

回帰分析の基本

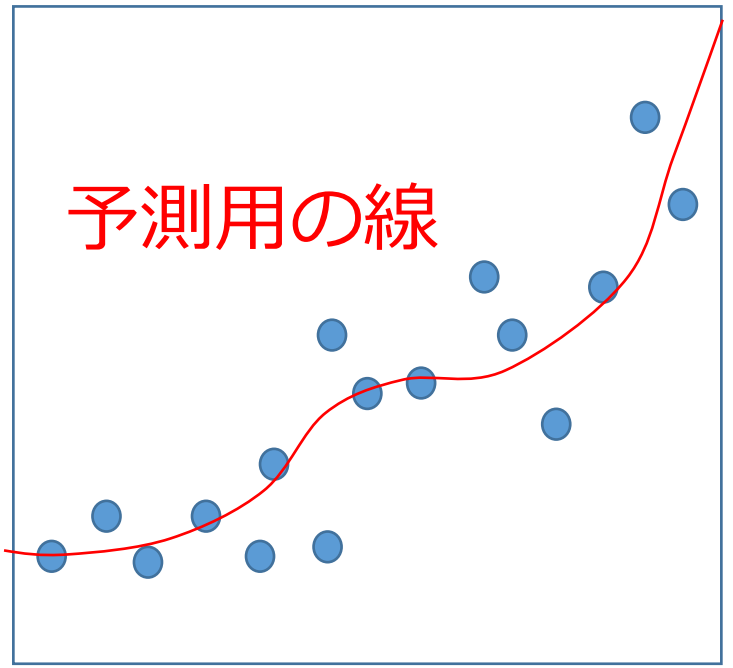
回帰：**ある量から別の量を予測**する手法
訓練データから構築されたモデルを用いて予測を行う。
例：身長と体重の関係，年齢と収入の関係

主要プロセス①　訓練データからのモデル構築

訓練データ



構築されたモデル



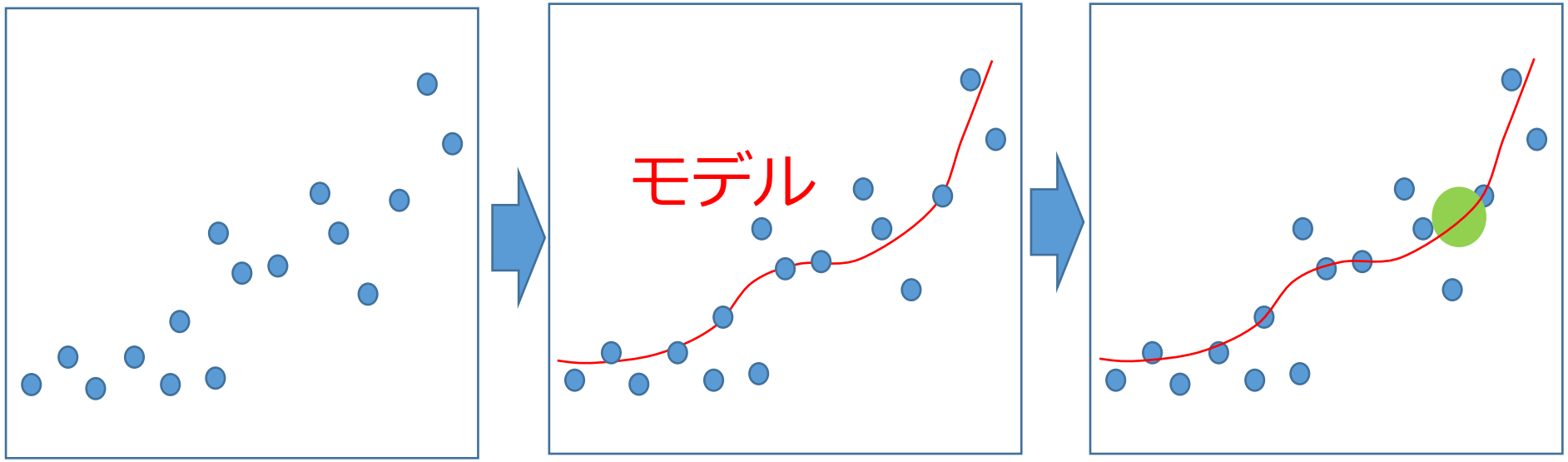
機械学習による回帰の仕組み

① **データの収集**：予測に必要な訓練データを準備.

② **モデルの構築**：データから予測用のモデルを作成.

③ **予測の実行**：新しいデータに対して予測値を算出.

① データの収集　② モデルの構築　③ 予測の実行



モデルを用いて，ＡＩが予測
（例）長さが５センチのときは，
幅は１．８センチ

回帰のバリエーション

- 線形回帰

  直線的な関係　　例：$y = mx + b$

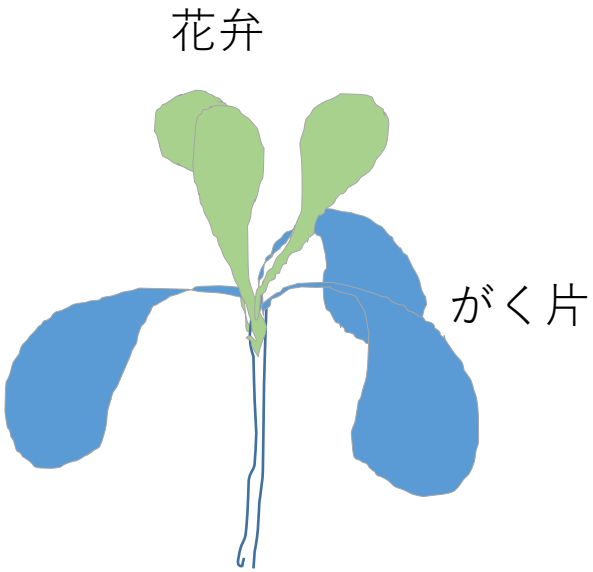
- 多項式回帰

  直線よりも複雑な関係　例：$y = ax^2 + bx + c$

- ディープラーニングによる高度なモデル

その他

アヤメ属 (Iris)



- 多年草
- 世界に 150種. 日本に 9種.
- **がく片**　Sepal

  3個（大型で下に垂れる）
- **花弁**　Petal

  3個（直立する）

Iris データセット

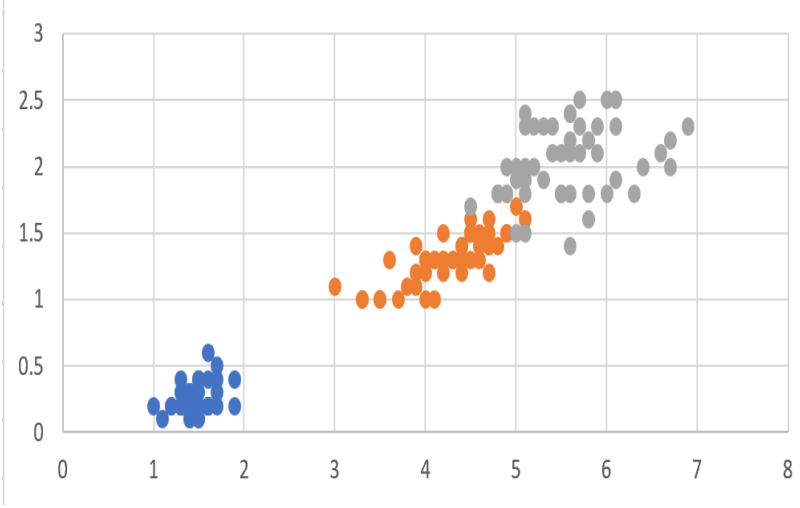
【概要】

- 有名なデータセット.
- 3種類のアヤメ（Iris）の花の特徴を記録したもの.

【データの内容】

- 150個のサンプル（各種類50個ずつ）.
- 4つの特徴（**がく片**の長さと幅、**花弁**の長さと幅）.
- 3つの種類（品種：setosa、versicolor、virginica）.

縦軸：花弁の**幅**



横軸：花弁の**長さ**

次の**3種類**で色を付けた散布図

setosa
versicolor
virginica

Iris データセット

Iris データセット
１５０サンプルのうち先頭１０

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa

がく片(Sepal)
の長さと幅

花弁(Petal)
の長さと幅

種類

演習１．多項式回帰

ページ２１～２５

- Irisデータセットを散布図にする．**データから，多項式回帰を行い，予測用の線を得る**

    １つめ：**がく片の長さと幅**

    ２つめ：**花弁の長さと幅**

- Matplotlib ライブラリを使用
- 予測や，データの関係性の分析に役立つ回帰曲線が得られる

（例）「花弁の長さが5.0cmの花に対して，その幅を1.7cmと予測」といった予測も可能になる

プログラム

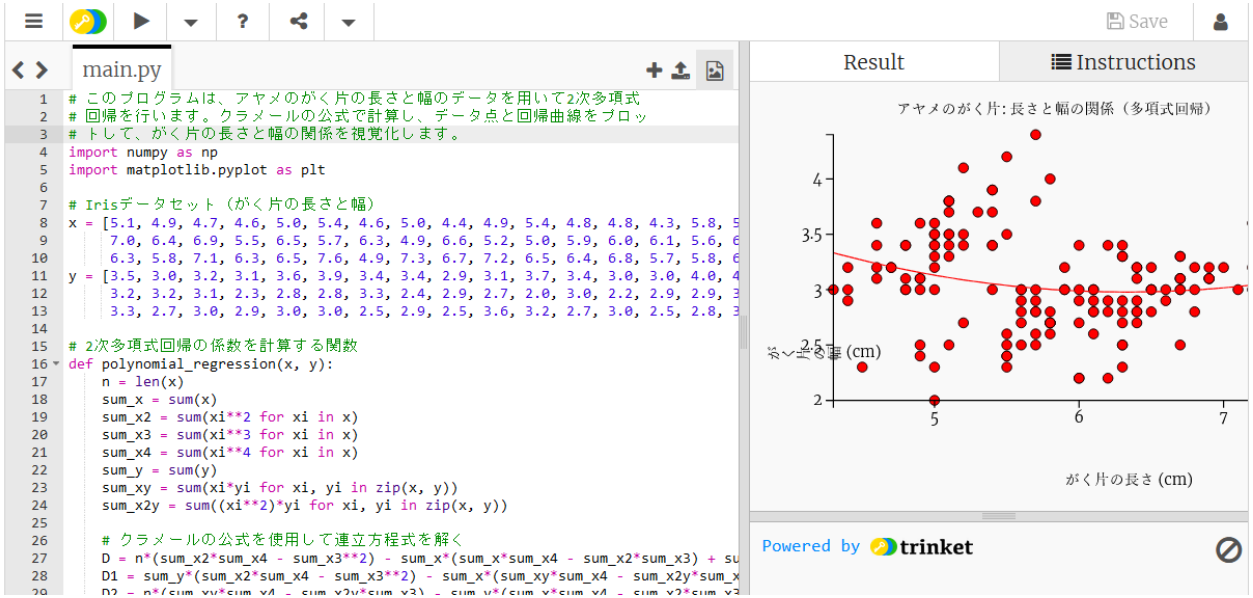

```
# このプログラムは、アヤメのがく片の長さと幅のデータを用いて2次多項式回帰を行います。
# クラメールの公式で計算し、データ点と回帰曲線をプロットして、がく片の長さと幅の関係
# を視覚化します。
import numpy as np
import matplotlib.pyplot as plt

# Irisデータセット（がく片の長さと幅）
x = [5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.8, 4.8, 4.3, 5.8, 5.7, 5.4, 5.1, 5.7, 5.1, 5.4, 5.1, 4.6, 5.1, 4.8, 5.0, 5.0, 5.2, 5.2, 4.7, 4.8, 5.4, 5.2, 5.5, 4.9, 5.0, 5.5, 4.9, 4.4, 5.1, 5.0, 4.5, 4.4, 5.0, 5.1, 4.8, 5.1, 4.6, 5.3, 5.0,
     7.0, 6.4, 6.9, 5.5, 6.5, 5.7, 6.3, 4.9, 6.6, 5.2, 5.0, 5.9, 6.0, 6.1, 5.6, 6.7, 5.6, 5.8, 6.2, 5.6, 5.9, 6.1, 6.3, 6.1, 6.4, 6.6, 6.8, 6.7, 6.0, 5.7, 5.5, 5.5, 5.8, 6.0, 5.4, 6.0, 6.7, 6.3, 5.6, 5.5, 5.5, 6.1, 5.8, 5.0, 5.6, 5.7, 5.7, 6.2, 5.1, 5.7,
     6.3, 5.8, 7.1, 6.3, 6.5, 7.6, 4.9, 7.3, 6.7, 7.2, 6.5, 6.4, 6.8, 5.7, 5.8, 6.4, 6.5, 7.7, 7.7, 6.0, 6.9, 5.6, 7.7, 6.3, 6.7, 7.2, 6.2, 6.1, 6.4, 7.2, 7.4, 7.9, 6.4, 6.3, 6.1, 7.7, 6.3, 6.4, 6.0, 6.9, 6.7, 6.9, 5.8, 6.8, 6.7, 6.7, 6.3, 6.5, 6.2, 5.9]
y = [3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.4, 3.0, 3.0, 4.0, 4.4, 3.9, 3.5, 3.8, 3.8, 3.4, 3.7, 3.6, 3.3, 3.4, 3.0, 3.4, 3.5, 3.4, 3.2, 3.1, 3.4, 4.1, 4.2, 3.1, 3.2, 3.5, 3.6, 3.0, 3.4, 3.5, 2.3, 3.2, 3.5, 3.8, 3.0, 3.8, 3.2, 3.7, 3.3,
     3.2, 3.2, 3.1, 2.3, 2.8, 2.8, 3.3, 2.4, 2.9, 2.7, 2.0, 3.0, 2.2, 2.9, 2.9, 3.1, 3.0, 2.7, 2.2, 2.5, 3.2, 2.8, 2.5, 2.8, 2.9, 3.0, 2.8, 3.0, 2.9, 2.6, 2.4, 2.4, 2.7, 2.7, 3.0, 3.4, 3.1, 2.3, 3.0, 2.5, 2.6, 3.0, 2.6, 2.3, 2.7, 3.0, 2.9, 2.9, 2.5, 2.8,
     3.3, 2.7, 3.0, 2.9, 3.0, 3.0, 2.5, 2.9, 2.5, 3.6, 3.2, 2.7, 3.0, 2.5, 2.8, 3.2, 3.0, 3.8, 2.6, 2.2, 3.2, 2.8, 2.8, 2.7, 3.3, 3.2, 2.8, 3.0, 2.8, 3.0, 2.8, 3.8, 2.8, 2.8, 2.6, 3.0, 3.4, 3.1, 3.0, 3.1, 3.1, 3.1, 2.7, 3.2, 3.3, 3.0, 2.5, 3.0, 3.4, 3.0]

# 2次多項式回帰の係数を計算する関数
def polynomial_regression(x, y):
    n = len(x)
    sum_x = sum(x)
    sum_x2 = sum(xi**2 for xi in x)
    sum_x3 = sum(xi**3 for xi in x)
    sum_x4 = sum(xi**4 for xi in x)
    sum_y = sum(y)
    sum_xy = sum(xi*yi for xi, yi in zip(x, y))
    sum_x2y = sum((xi**2)*yi for xi, yi in zip(x, y))

    # クラメールの公式を使用して連立方程式を解く
    D = n*(sum_x2*sum_x4 - sum_x3**2) - sum_x*(sum_x*sum_x4 - sum_x2*sum_x3) + sum_x2*(sum_x*sum_x3 - sum_x2**2)
    D1 = sum_y*(sum_x2*sum_x4 - sum_x3**2) - sum_x*(sum_xy*sum_x4 - sum_x2y*sum_x3) + sum_x2*(sum_xy*sum_x3 - sum_x2y*sum_x2)
    D2 = n*(sum_xy*sum_x4 - sum_x2y*sum_x3) - sum_y*(sum_x*sum_x4 - sum_x2*sum_x3) + sum_x2*(sum_x*sum_x2y - sum_xy*sum_x2)
    D3 = n*(sum_x2*sum_x2y - sum_xy*sum_x3) - sum_x*(sum_x*sum_x2y - sum_xy*sum_x2) + sum_y*(sum_x*sum_x3 - sum_x2**2)

    a0 = D1 / D
    a1 = D2 / D
    a2 = D3 / D

    return a2, a1, a0

# 多項式回帰の係数を計算
a2, a1, a0 = polynomial_regression(x, y)

# データポイントのプロット
plt.plot(x, y, 'bo', markersize=4, alpha=0.7)

# 回帰曲線の描画
x_plot = np.linspace(min(x), max(x), 100)
y_plot = [a2*xi**2 + a1*xi + a0 for xi in x_plot]
plt.plot(x_plot, y_plot, color='red')

# グラフの設定
plt.xlabel('がく片の長さ (cm)')
plt.ylabel('がく片の幅 (cm)')
plt.title('アヤメのがく片: 長さと幅の関係（多項式回帰）')

# プロットの範囲を設定
plt.axis([min(x), max(x), min(y), max(y)])

# グラフの表示
plt.show()
```

がく片の長さと幅

① trinket の次のページを開く

https://trinket.io/python/f430c55eba59

② 実行結果が，次のように表示されることを確認



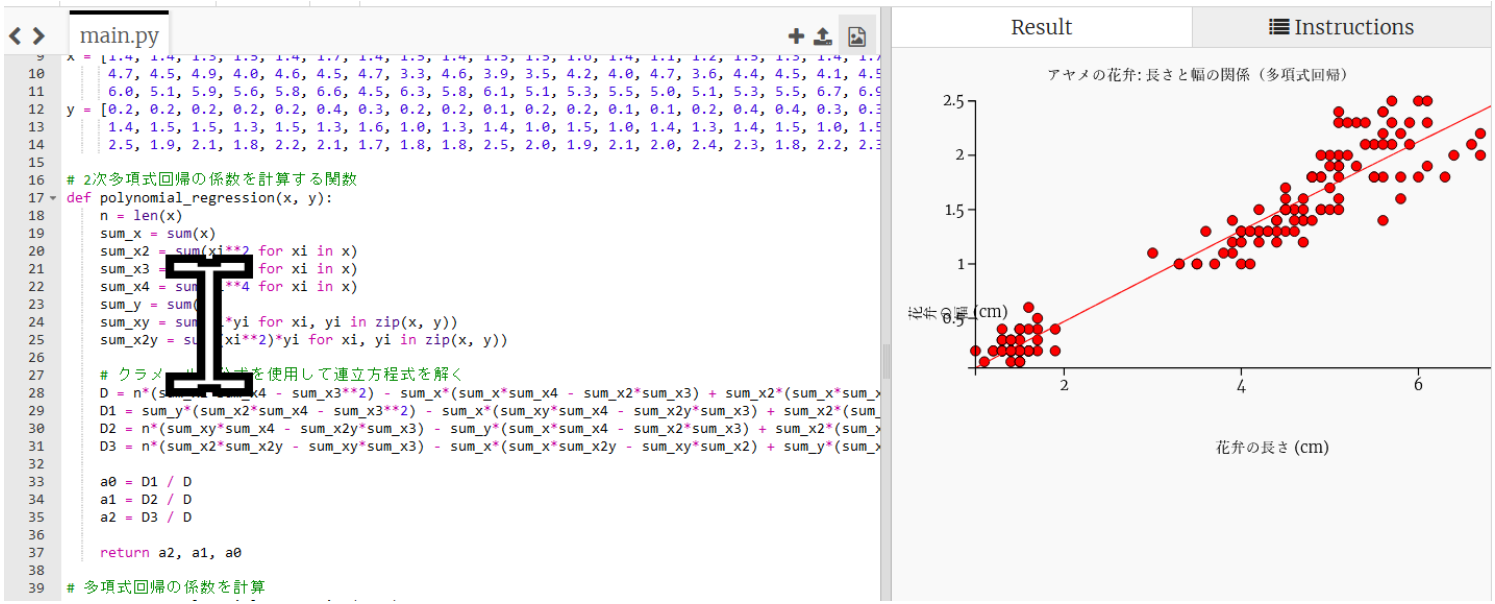
- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

花弁の長さと幅

③ trinket の次のページを開く

https://trinket.io/python/8067a00f4938

④ 実行結果が，次のように表示されることを確認



- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

演習２．多項式回帰の観察

ページ２６～２９

- 自分で５つのデータを与えて，多項式回帰を実施

  （例）３２１３５

- 与えるデータを変えて，プログラムを繰り返し実行してみる

多項式回帰の観察

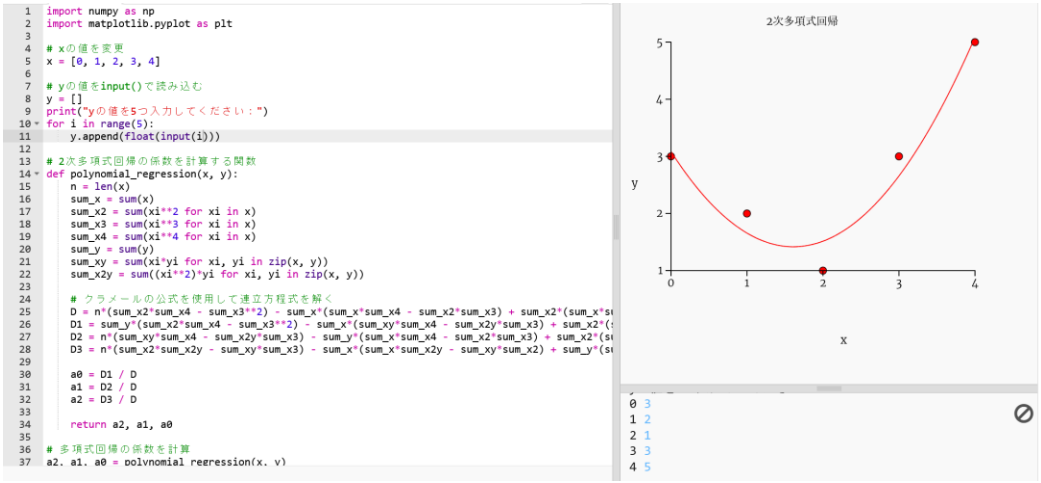
① trinket の次のページを開く

https://trinket.io/python/f24bf28125a9

② **右下の画面で，5つのデータ**を与える

操作例 **3 Enter 2 Enter 1 Enter 3 Enter 5 Enter**

③ 実行結果が，次のように表示されることを確認



- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

機械学習の概要と応用

① データの収集：

例： Irisデータセットの花弁の長さと幅のデータ

② モデルの構築：

- データを使って，コンピュータに学習させる.

③ 予測の実行：

- 学習の結果，新しいデータに対して予測ができるようになる.
- 例：花弁の長さが5.0cmの花に対して，その幅を1.7cmと予測

機械学習の特徴

- 従来のプログラミングは「花弁の長さがXcmなら幅はYcmである」といったルールを人間が記述する必要があった.
- 機械学習ではこれらの関係性を自動的に学習する.

データから価値ある情報を抽出し，新たな知見を得ることが可能になる.

ここまでのまとめ

- **回帰**は，**ある量から別の量を予測**するためのもの
- **線形回帰**は**直線的な関係**を，**多項式回帰**は**より複雑な関係**を表現する.
- **Irisデータセット**は3種類のアヤメの花の特徴を記録した有名なデータセットである.
- 演習では，**データ**から，**多項式回帰**を行い，**回帰曲線**を得た．**予測や，データの関係性の分析に役立つ回帰曲線**が得られた.

3-3. 色付きの散布図

Trinket を用いた色付き散布図

現在のツール：Trinket

- 練習用に適している
- 外部ライブラリは限定的
- 機能（色付き散布図など）に制限あり

将来のツール：Google Colaboratory

- 多数の外部ライブラリが利用可能
- 高度なデータ分析・可視化・AI実践が可能

【学習の流れ】

- **現在：Trinketで基礎を学ぶ**
- 将来：より高機能なGoogle Colaboratoryへ移行予定

いまは，機能が少なく，その分使いやすい Trinket で頑張る

演習３．Trinket による色付き散布図

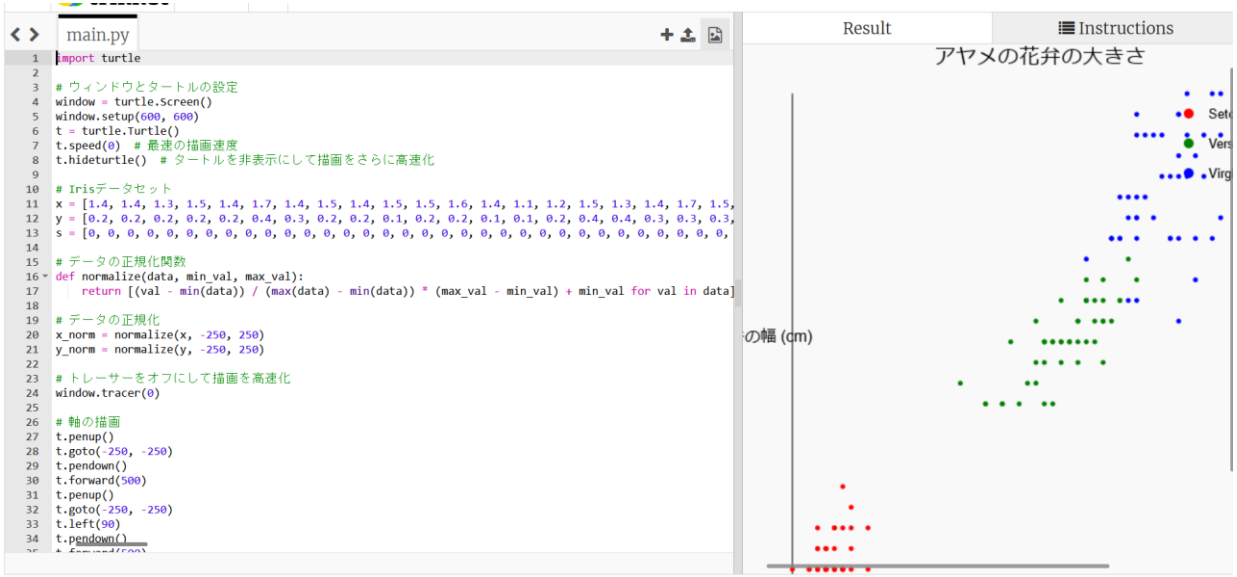
ページ３３～３４

Trinket で色付きの散布図

① trinket の次のページを開く

https://trinket.io/python/b0a410c9d8c2

② 実行結果が，次のように表示されることを確認



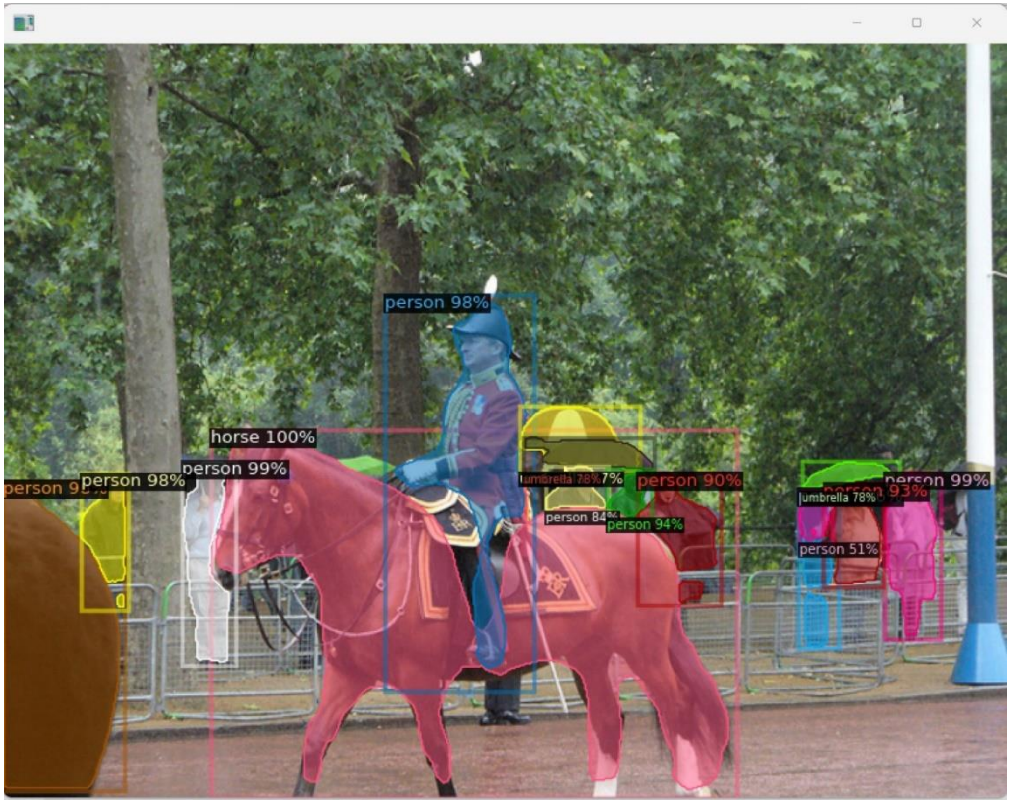

```python
import turtle

# ウィンドウとタートルの設定
window = turtle.Screen()
window.setup(600, 600)
t = turtle.Turtle()
t.speed(0)  # 最速の描画速度
t.hideturtle()  # タートルを非表示にして描画をさらに高速化

# Irisデータセット
x = [1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.6, 1.4, 1.1, 1.2, 1.5, 1.3, 1.4, 1.7, 1.5,
y = [0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.2, 0.1, 0.1, 0.2, 0.4, 0.4, 0.3, 0.3, 0.3,
s = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

# データの正規化関数
def normalize(data, min_val, max_val):
    return [(val - min(data)) / (max(data) - min(data)) * (max_val - min_val) + min_val for val in data]

# データの正規化
x_norm = normalize(x, -250, 250)
y_norm = normalize(y, -250, 250)

# トレーサーをオフにして描画を高速化
window.tracer(0)

# 軸の描画
t.penup()
t.goto(-250, -250)
t.pendown()
t.forward(500)
t.penup()
t.goto(-250, -250)
t.left(90)
t.pendown()
```

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

3-4. 分類

分類の基本

- **分類**は，**データを予め定義された種類**（例：自動車，人間，自転車）**に分類**する手法である.
- 機械学習では，**訓練データから分類の規則を学習**し，**新しいデータを自動的に分類**する（例：画像認識での物体分類）.



AIの実行結果
画像全体の画素を，
種類に応じて分類する
画像セグメンテーション

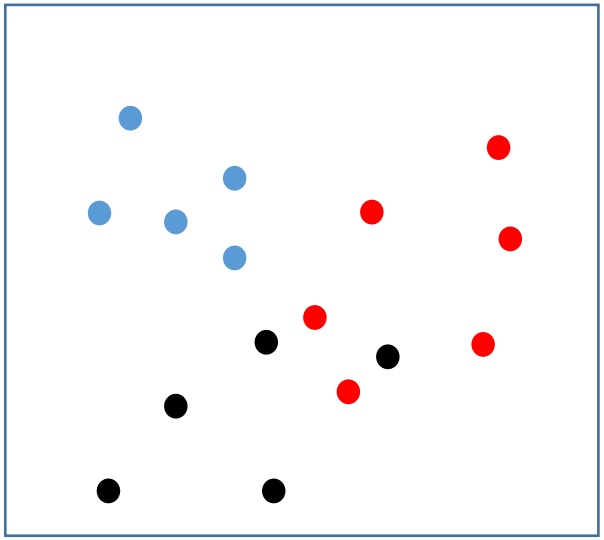
機械学習における分類の仕組み

① **データの収集**：予測に必要な訓練データを準備.

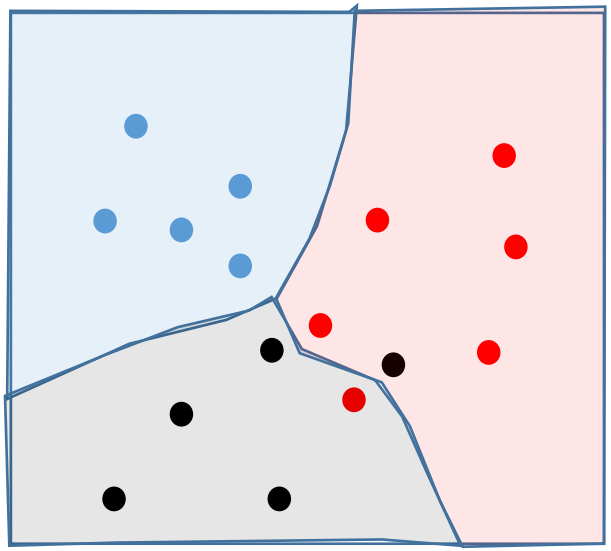
② **モデルの構築**：データから分類用のモデルを作成.

③ **予測の実行**：新しいデータを適切なクラスに分類する.

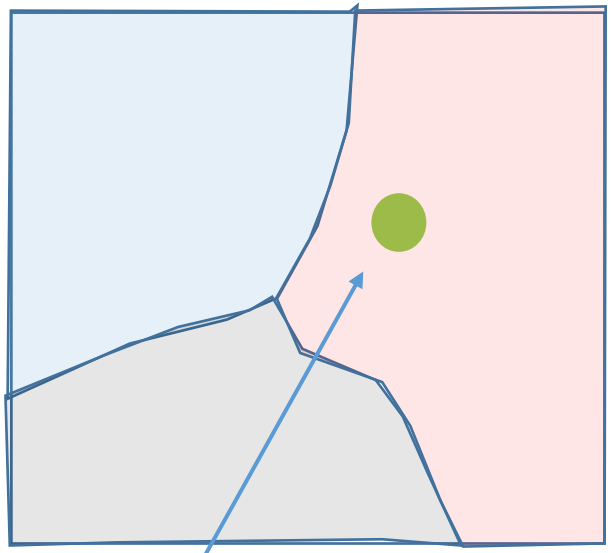
① データの収集



② モデルの構築



③ 分類の実行



モデルを用いて，
ＡＩが種類を予測
（例）種類は赤

演習４．分類のモデル

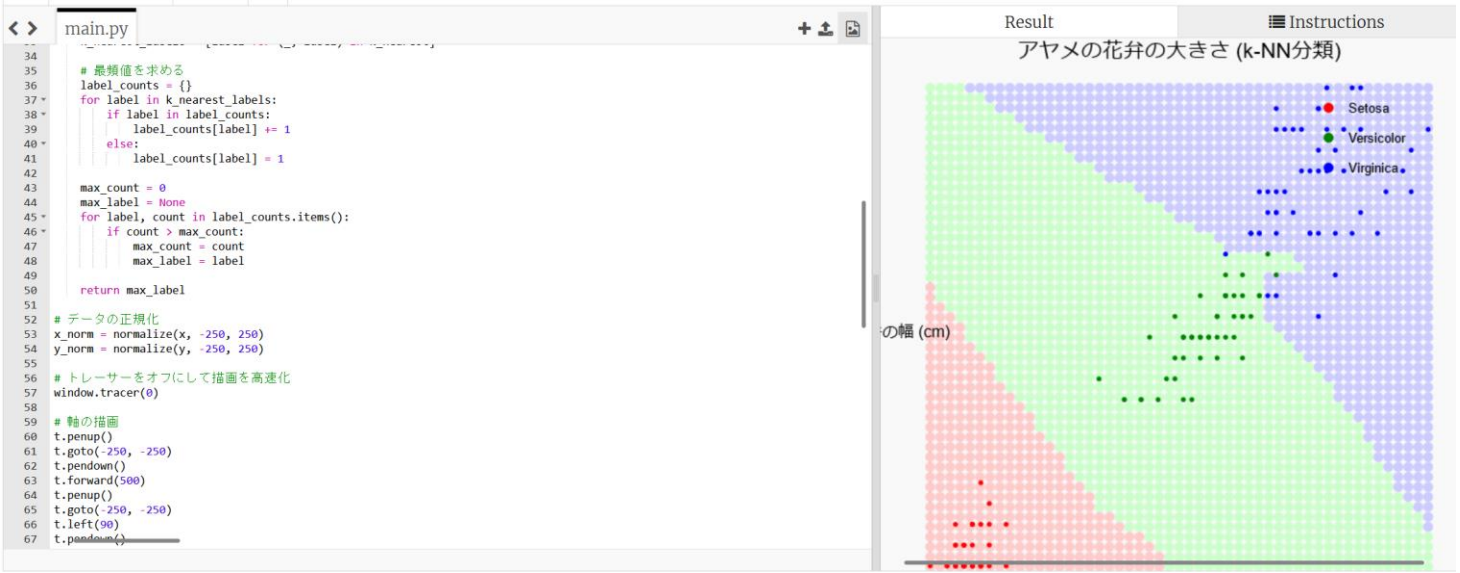
ページ４０～４２

分類のモデル

① trinket の次のページを開く

https://trinket.io/python/918ff17a9ad9

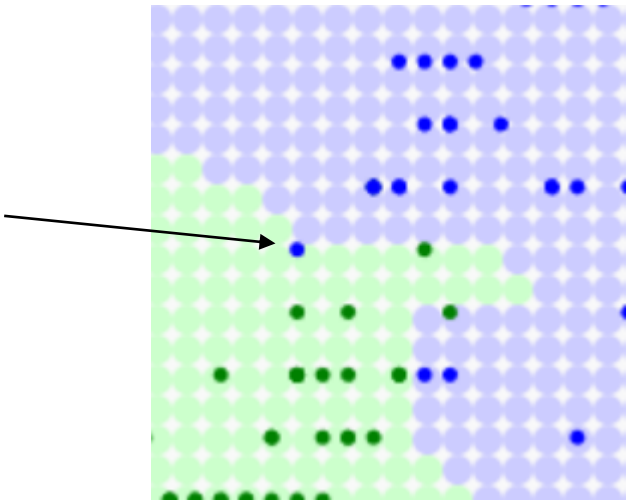
② 実行結果が，次のように表示されることを確認



- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

演習４の学習ポイント

- プログラムは，**アヤメの花の分類モデル**を作成
- **データの分布：3種類のアヤメがどのように分布しているかを直感的に理解**できる
- **予測への利用**：分類モデルは，**アヤメの花弁の長さと幅のデータを使用**し、**3つの種類**（セトサ、バーシカラー、バージニカ）に**分類**することが可能
- クラス間の重複: 一部のクラスが重なっている領域があることを観察でき、**分類の難しさ**を理解できる



演習５．分類モデルの観察

ページ４３～４６

- 二クラス分類の分類モデル（クラス１，クラス２に分類）

- 自分でデータを与える

（例）

クラス１

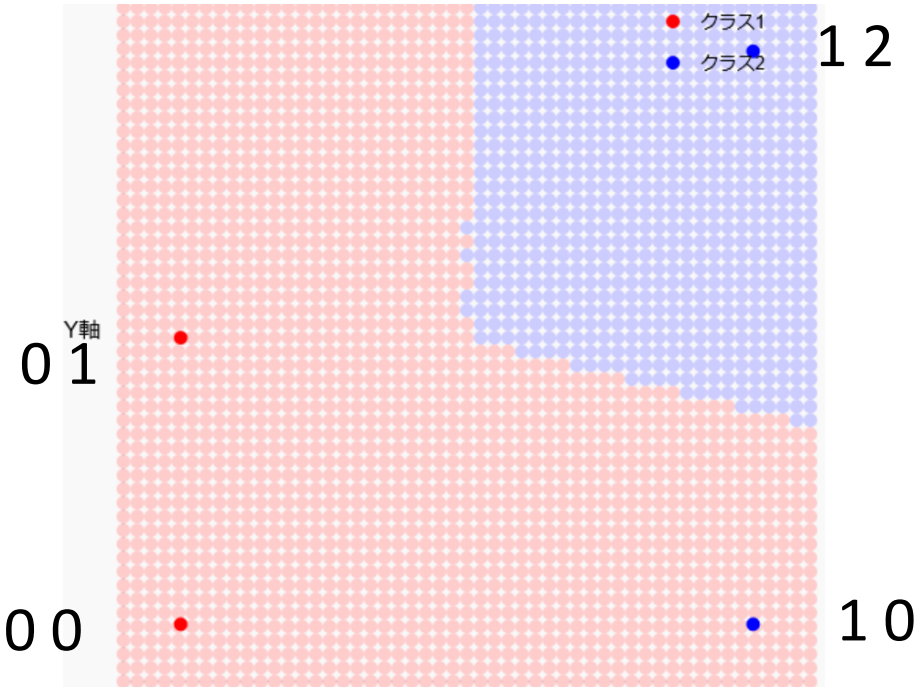
０ ０

０ １

クラス２

１ ０

１ ２



生成される分類モデル

分類モデルの観察

① trinket の次のページを開く

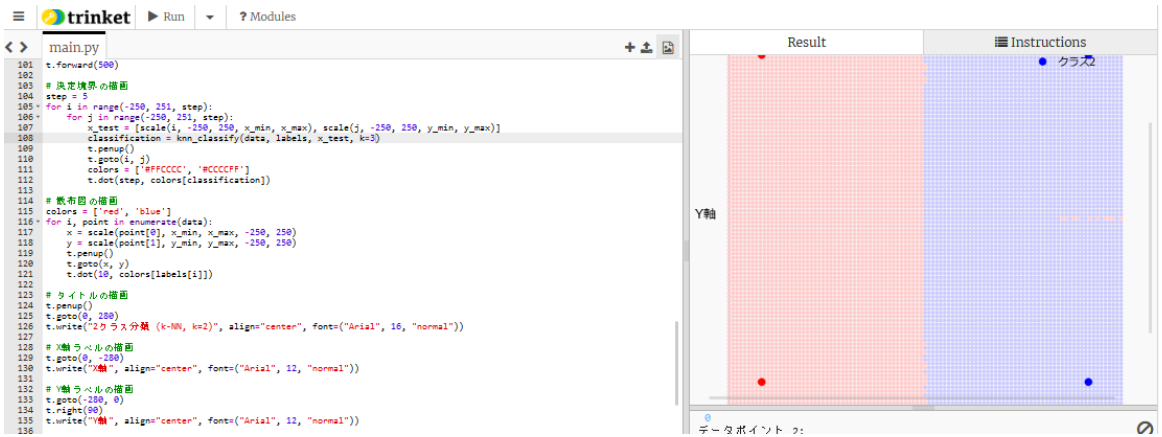
https://trinket.io/python/cd9fe4c24453

② **右下の画面で，8つのデータ**を与える

操作例 **0 Enter 0 Enter 0 Enter 1 Enter**

1 Enter 0 Enter 1 Enter 1 Enter

③ 実行結果が，次のように表示されることを確認



右側の画面は，
スクロール，
大きさ変更などで
調整

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

データ入力の手順

- クラス1のデータポイント1のx座標を入力し、Enterキー
- クラス1のデータポイント1のy座標を入力し、Enterキー
- クラス1のデータポイント2のx座標を入力し、Enterキー
- クラス1のデータポイント2のy座標を入力し、Enterキー
- クラス2のデータポイント1のx座標を入力し、Enterキー
- クラス2のデータポイント1のy座標を入力し、Enterキー
- クラス2のデータポイント2のx座標を入力し、Enterキー
- クラス2のデータポイント2のy座標を入力し、Enterキー

すべて半角の数値で入れてください

値を変えて，実行を繰り返してみましょう

ここまでのまとめ

- データを、**あらかじめ分かっている種類**（例：自動車，人間，自転車）**に分類**すること
- 分類の主要プロセス

①訓練データからのモデル構築

②モデルを用いた予測

- 分布の理解（例：3種類のアヤメがどのように分布しているか），自動分類への応用
- クラス間の重複

一部のクラスが重なっている領域があることを観察、分類の難しさを理解

※　演習４，５では「k-NN」という分類法を使っています

3-5. 決定木を用いた分類

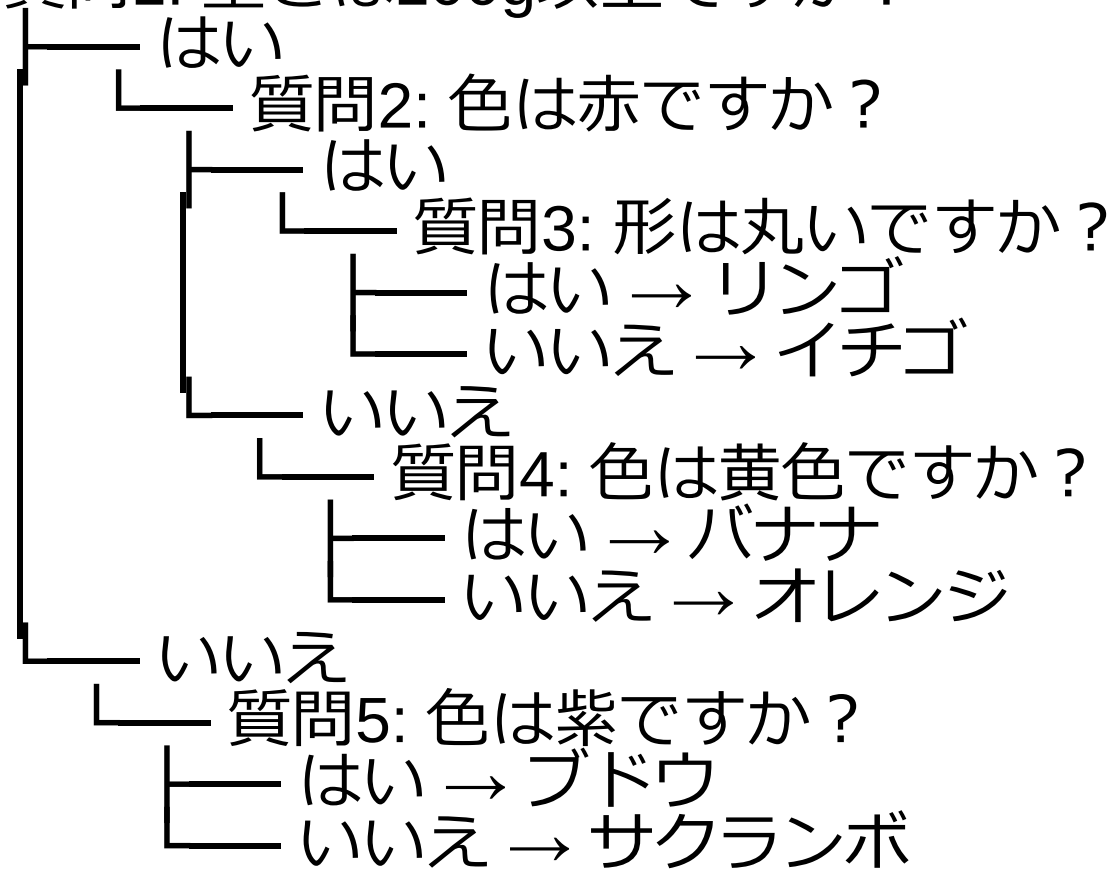
機械学習での決定木

- **決定木**：データの特徴に基づいて段階的に分類を行う手法
- **質問と回答の木構造を用いて判断**を行う（例：果物の特徴から種類を判断する）.

機械学習での決定木

訓練データからのモデル構築

データを使って、質問と回答の木を作る

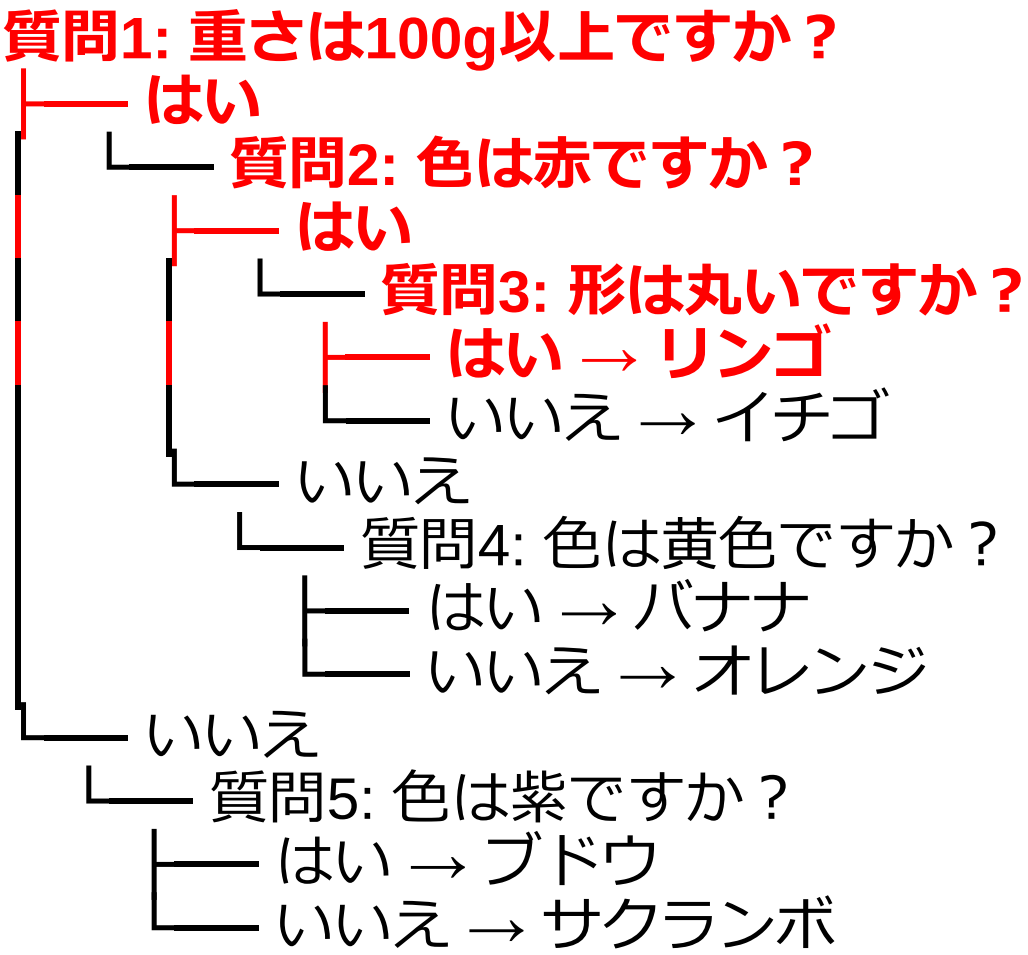


機械学習での決定木

モデルを用いて、AIが分類や予測を行う

（例）新しい果物のデータ：重さ: 150g, 色: 赤, 形: 丸い

AIの予測: この果物はリンゴです



演習５．決定木

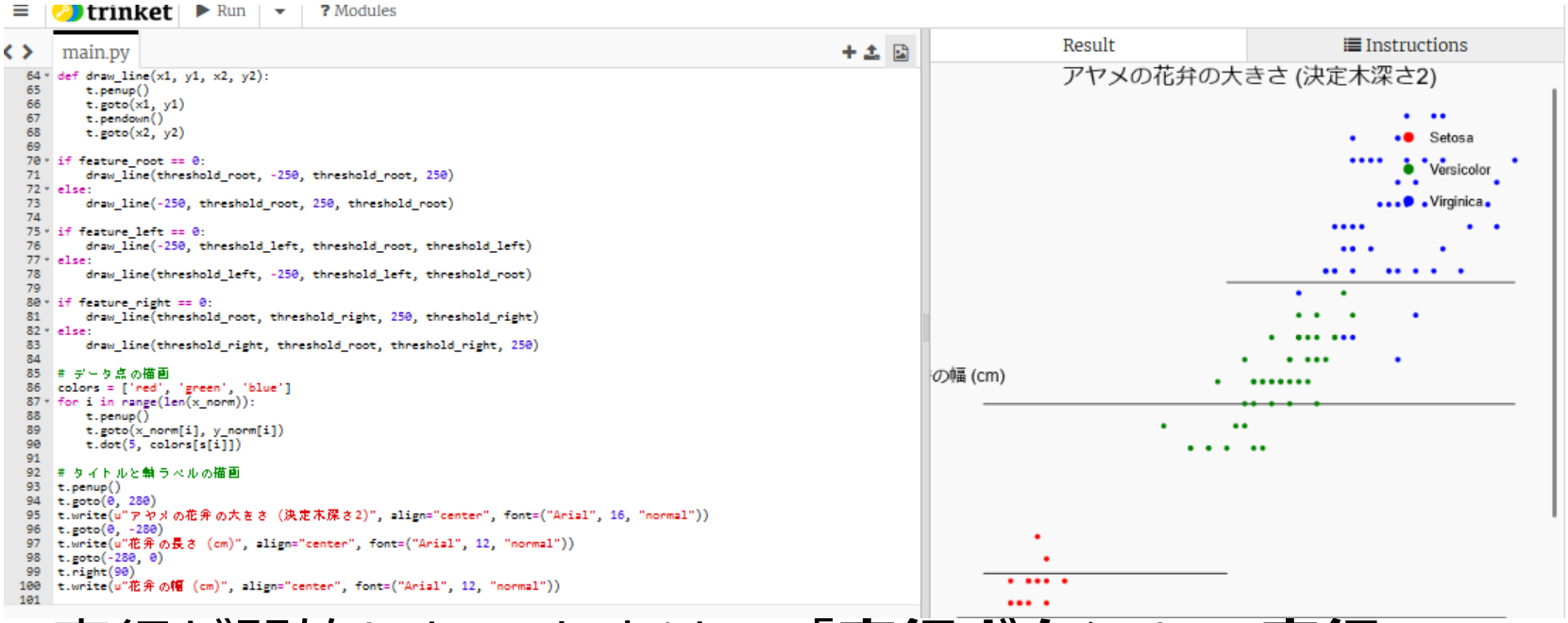
ページ５２～５４

分類のモデル

① trinket の次のページを開く

https://trinket.io/python/37d6744c8013

② 実行結果が，次のように表示されることを確認



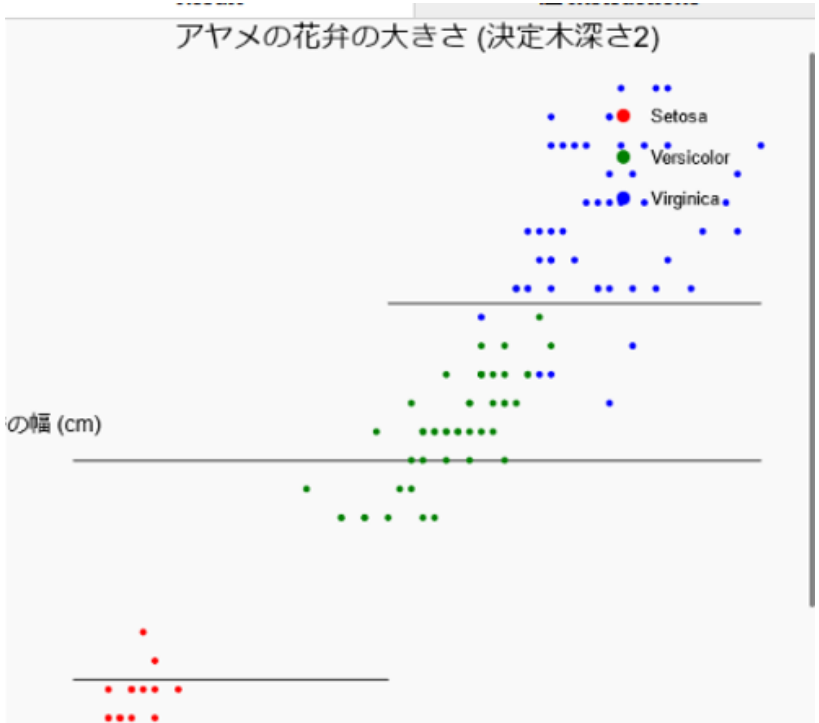

```
def draw_line(x1, y1, x2, y2):
    t.penup()
    t.goto(x1, y1)
    t.pendown()
    t.goto(x2, y2)

if feature_root == 0:
    draw_line(threshold_root, -250, threshold_root, 250)
else:
    draw_line(-250, threshold_root, 250, threshold_root)

if feature_left == 0:
    draw_line(-250, threshold_left, threshold_root, threshold_left)
else:
    draw_line(threshold_left, -250, threshold_left, threshold_root)

if feature_right == 0:
    draw_line(threshold_root, threshold_right, 250, threshold_right)
else:
    draw_line(threshold_right, threshold_root, threshold_right, 250)

# データ点の描画
colors = ['red', 'green', 'blue']
for i in range(len(x_norm)):
    t.penup()
    t.goto(x_norm[i], y_norm[i])
    t.dot(5, colors[s[i]])

# タイトルと軸ラベルの描画
t.penup()
t.goto(0, 280)
t.write(u"アヤメの花弁の大きさ (決定木深さ2)", align="center", font=("Arial", 16, "normal"))
t.goto(0, -280)
t.write(u"花弁の長さ (cm)", align="center", font=("Arial", 12, "normal"))
t.goto(-280, 0)
t.right(90)
t.write(u"花弁の幅 (cm)", align="center", font=("Arial", 12, "normal"))
```

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

演習６の学習ポイント

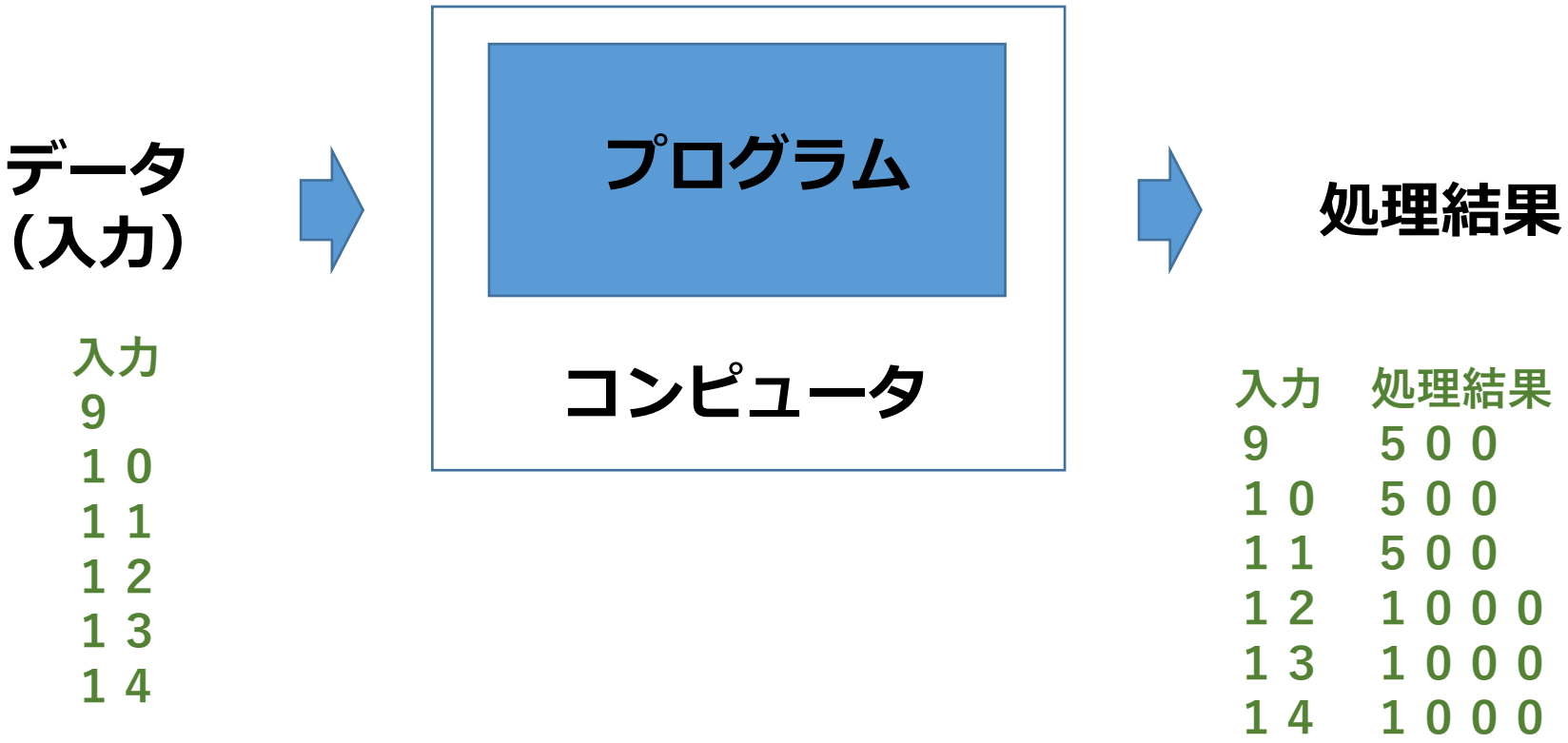
- 決定木のビジュアル表示：　決定木で，データの分類がどのように行われているかを直感的に理解できる

- 決定境界の形状：　決定木では，分類境界が直線的（軸に平行）でシンプル．これが決定木の特徴．



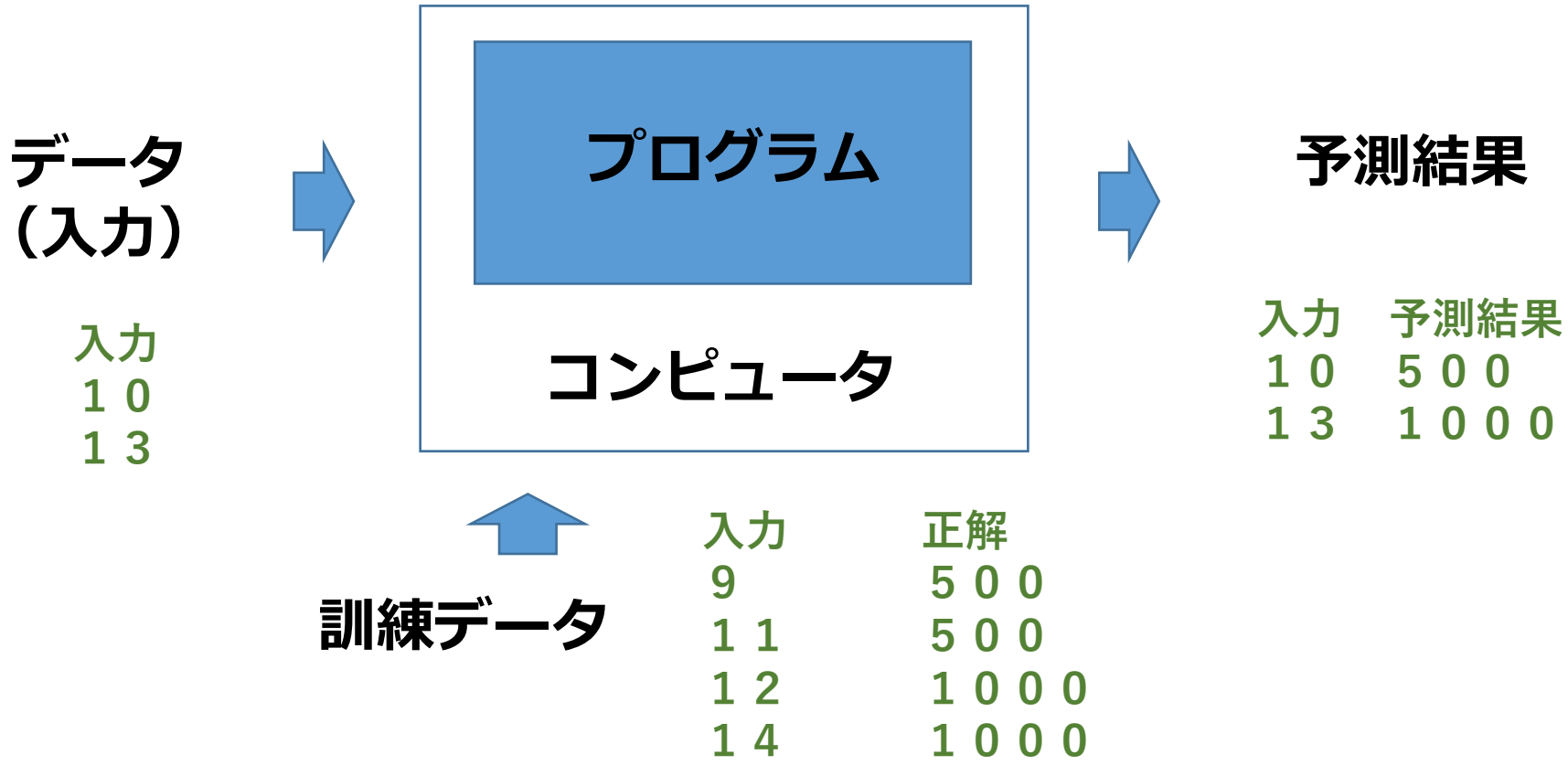
3-6. 機械学習での学習と汎化

従来のプログラミング手法



従来のプログラミングでは，プログラム作成者が全ての処理規則を記述する必要がある．

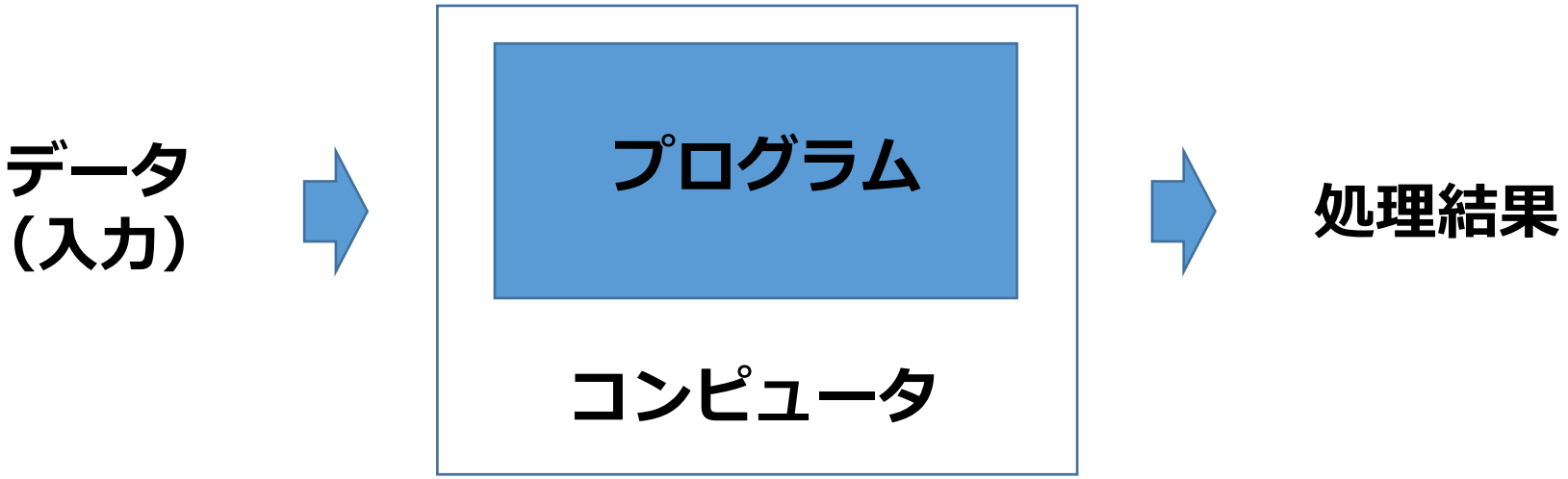
機械学習によるプロセス



- 機械学習では，訓練データから自動的にパターンや関連性を学習する．
- 学習結果のモデルを利用して、新しいデータ（未知のデータ）に対して，タスクが実行される．人手による規則の記述が不要である．

① 一般のプログラミング

・プログラムは人間が作成しテストし，調整する．



② 機械学習

・データを利用して知的能力を向上させる

機械学習の４つの特徴

1. **データ駆動型学習**：訓練データを使用して知的能力を向上させる.
2. **パターン認識**：データから自動的にパターンや関連性を抽出する.
3. **自動化と効率化**：学習した能力を用いて処理を自動実行する.
4. **多様な能力**：分類，予測，合成など様々なタスクに対応する.

汎化の概念と特徴

- **汎化**は，**訓練データを基に学習**し，**未知のデータも適切に処理**する能力である.
- 機械学習の重要な特徴.
- 汎化の結果は必ずしも正解とは限らない.

汎化

訓練データ

入力	正解
9	500
11	500
12	1000
14	1000

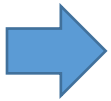
訓練データと未知のデータ

入力	予測結果
7	**500**
8	**500**
9	500
10	**500**
11	500
12	1000
13	**1000**
14	1000
15	**1000**
16	**1000**

汎化は，**訓練データを基に学習**し，
未知のデータも適切に処理する能力

汎化の結果は「必ず正解」ではない

汎化により「**はい**」，「**いいえ**」の**予測**



		予測	
		はい	いいえ
正解	はい	**予測成功**	予測失敗
	いいえ	予測失敗	**予測成功**

複雑な問題になると、**予測失敗**の可能性が出てくる

まとめ

- **機械学習の基本**：データを用いた学習による知的能力の向上
- **訓練データの重要性**：学習は訓練データに依存する
- **予測と分類**：機械学習の主要なタスク
- **汎化**：**未知データへの対応**
- 実践的応用：様々な分野での活用可能性