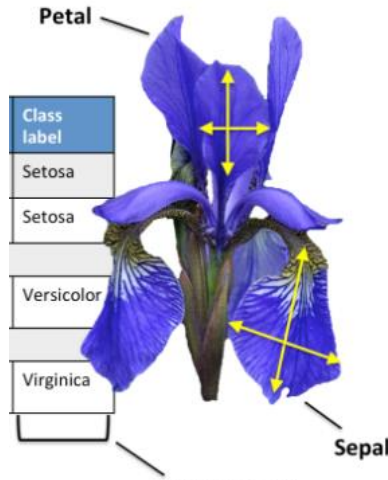
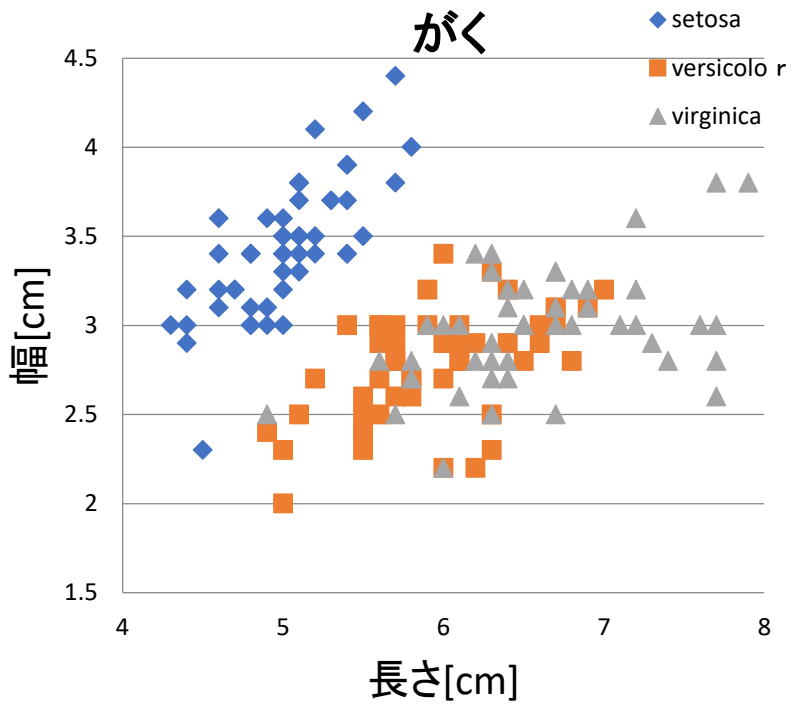
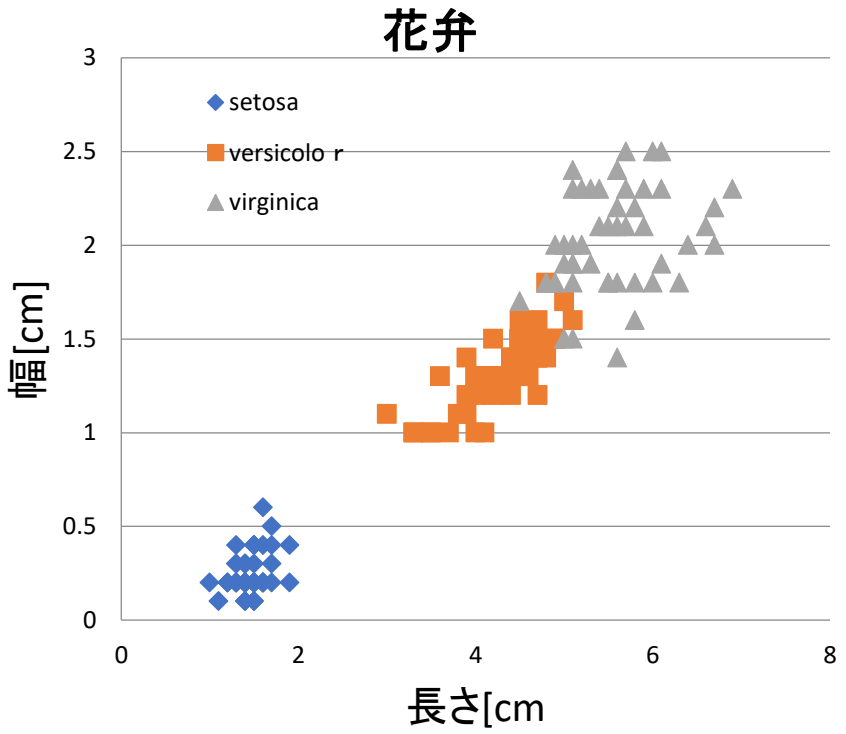
アイリスのデータを機械学習して、品種を予測する

花弁



がく

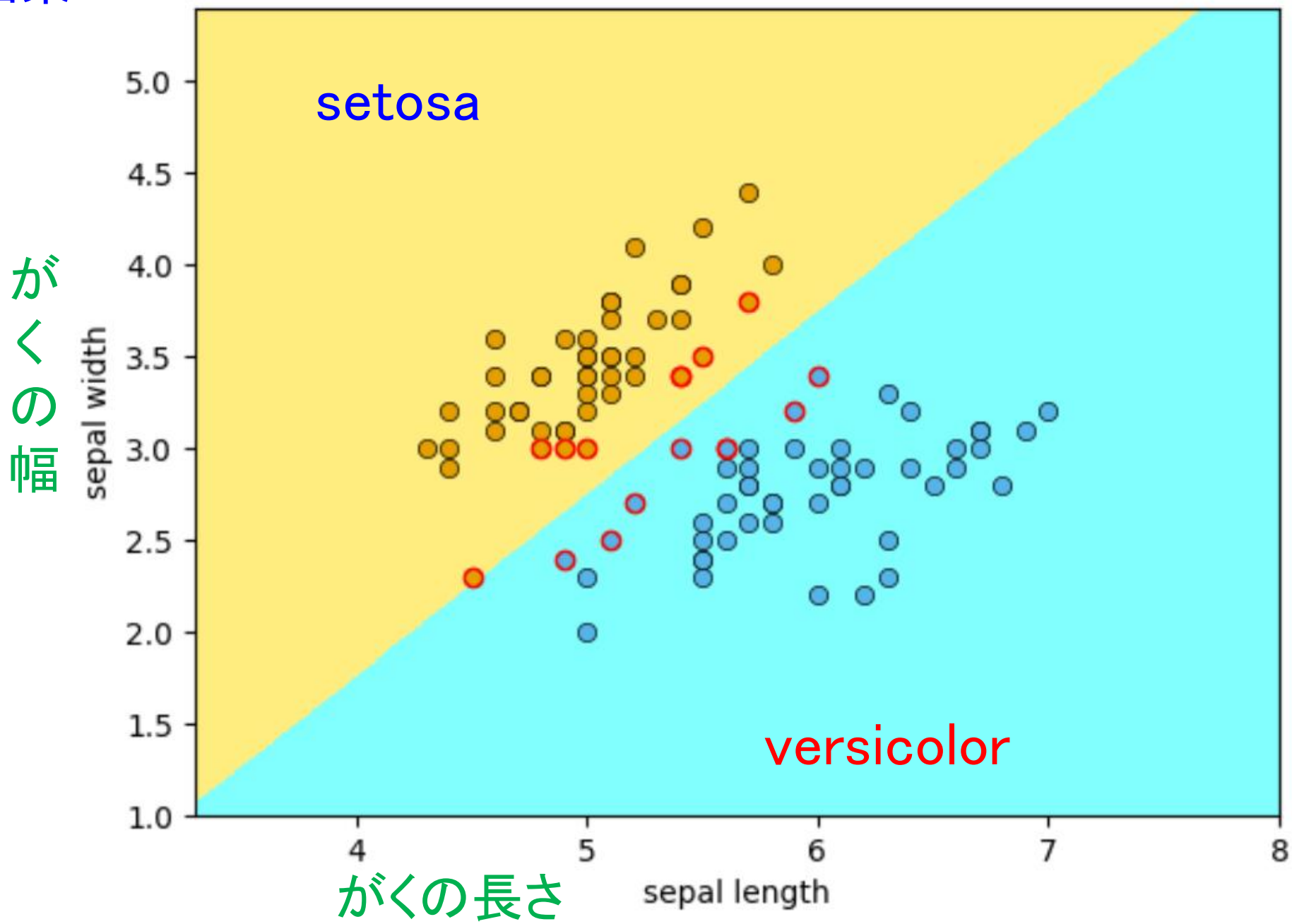




課題	特徴量	特徴量の次元	分類先	分類先のクラス数
アヤメの分類1	がくの長さ	1	iris setosa iris versicolor	2
アヤメの分類2	がくの長さ がくの幅	2	iris setosa iris versicolor	2
アヤメの分類3	がくの長さ がくの幅	2	iris setosa iris versicolor iris virginica	3
アヤメの分類4	がくの長さ がくの幅 花弁の長さ 花弁の幅	4	iris setosa iris versicolor iris virginica	3

実行結果

setosaとversicolor



がくの幅

がくの長さ

Scikit-learnの機能

パッケージ	モジュール	機能の例	意味
sklearn	datasets	Load_iris	アヤメデータの読み込み
		Load_digits	手書き画像データの読み込み
	svm	SVC	分類用のサポートベクターマシン
	Neural_network	MLPClassifier	分類用の多層ニューラルネットワーク
	decomposition	PCA	主成分分析
	Linear_model	Perceptron	単純パーセプトロン

変数名	キーワード	内容	
iris	data	アヤメの特徴量を含むデータ	
	target	ターゲット　0　1　2　とおく	
	Target_names	ターゲットの名前(setosa/versicolor/virginica)	⇦3つアヤメの種類
	Feature_names	特徴量の名前(sepal length/sepal width/petal length/petal width)	⇦4つの特徴量
	DESCR	アヤメのデータに関する詳細な解説	

コメント文　実行に関係しない

プログラム中の日本語がUTF-8という文字コードであることを示す

```
# -*- coding: utf-8 -*-
from sklearn import datasets
import numpy as np

# アヤメのデータをロードし、変数irisに格納
iris = datasets.load_iris()

# 特徴量のセットを変数Xに、ターゲットを変数yに格納
X = iris.data
y = iris.target

# Xとyをそのまま表示
print(X)
print(y)

# Xとyの次元を表示。それぞれ(150, 4)と(150,)となる。
# サンプル数150, 特徴量の次元4, という意味
print(X.shape)
print(y.shape)

# サンプル数と特徴量の次元の取り出し方法
(n_samples, n_features) = X.shape
print('サンプル数: {0}'.format(n_samples))
print('特徴量の次元: {0}'.format(n_features))

# クラス数の取り出し方法
n_classes = len(np.unique(y))
print('クラス数: {0}'.format(n_classes))
```

Scikit-learnの中にあるサンプルデータを取り出す

変数＝データを格納する容器

(150,4)を出力

(150,)を出力

{0}の部分が変数n_samples(150)の中身で置き換えられる

ターゲットに「0、1、2」の3つの数字しか現れないのでクラスは3となる

len関数

```
print(X)
print(y)
```

実行結果

4つの特徴量

sepal length	/sepal width	/petal length	/petal width
ガクの長さ	ガクの幅	花弁の長さ	花弁の幅

X

150個のデータ
（1クラスあたり50個
3クラス分）

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 ...
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]
```

x_0 x_1 . . x_{149}

⇦4つデータで一つのかたまり

アヤメの種類（クラス）
setosa → 0
versicolor → 1
virginica → 2

ターゲットy

```
[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 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

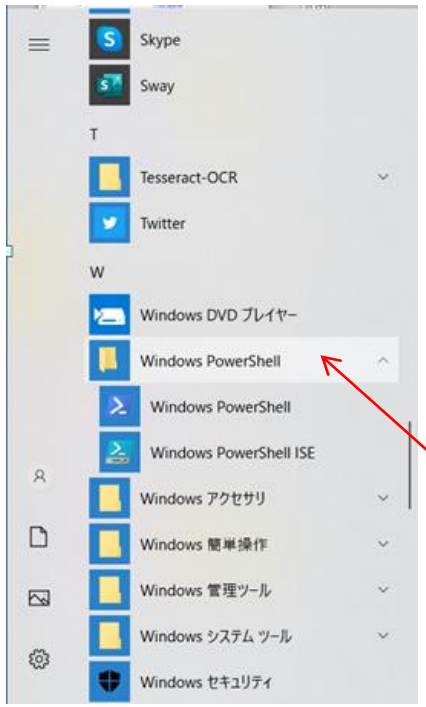
y_0 y_1 .. y_{149}

```
print(X.shape)
print(y.shape)
```

```
(150, 4)
(150,)
```

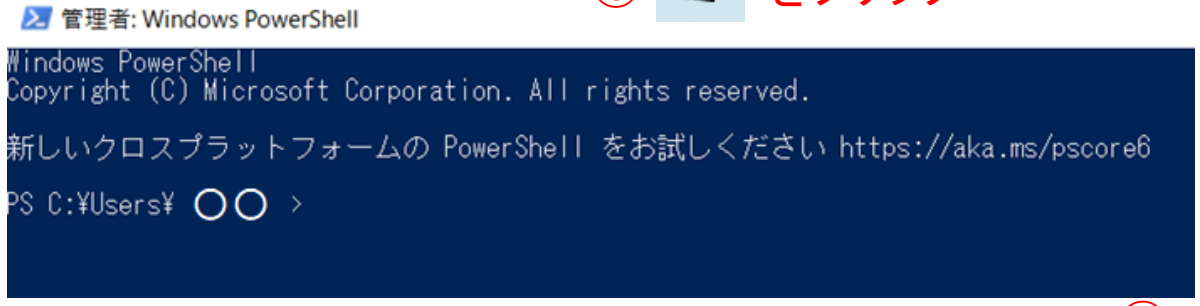
```
print('サンプル数: {0}'.format(n_samples))
print('特徴量の次元: {0}'.format(n_features))
print('クラス数: {0}'.format(n_classes))
```

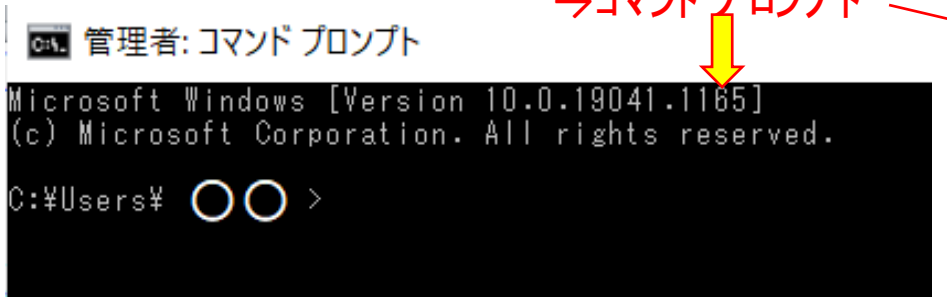
```
サンプル数: 150
特徴量の次元: 4
クラス数: 3
```



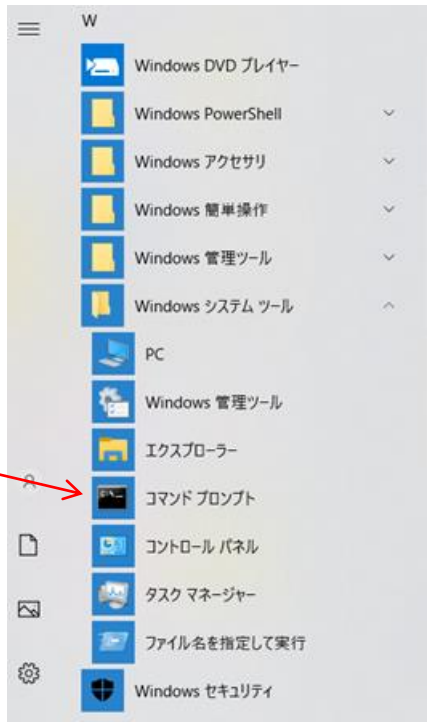
②Windows PowerShell

①　をクリック





②'Windows システムツール
→ｺﾏﾝﾄﾞﾌﾟﾛﾝﾌﾟﾄ



入力するコマンド　ﾗｲﾌﾞﾗﾘｰ名	用途
C:¥Users¥○○> py −m pip install −U pip	ライブラリーをインストールする
C:¥Users¥○○> py −m pip install numpy	多次元配列の数値計算ライブラリ
C:¥Users¥○○> py −m pip install matplotlib	グラフ描画ライブラリ
C:¥Users¥○○> py −m pip install scikit−learn	機械学習ライブラリ
C:¥Users¥○○> py −m pip install pandas	データ解析用ライブラリ
C:¥Users¥○○> py -m pip install opencv	画像や動画を処理ライブラリ
C:¥Users¥○○> py -m pip install opencv-contrib-python	Opencvの拡張モジュール群

```python
# -*- coding: utf-8 -*-
from sklearn import datasets, svm
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

# アヤメのデータをロードし、変数irisに格納
iris = datasets.load_iris()

# 特徴量のセットを変数Xに、ターゲットを変数yに格納
X = iris.data
y = iris.target

# 特徴量を外花被片の長さ(sepal length)と幅(sepal width)の
# 2つのみに制限(2次元で考えるため)
X = X[:,:2]

# ターゲットは2 (iris virginica) でないもの,
# つまり iris setosa (0) と iris versicolor (1) のみを対象とする
# (領域の2分割)
X = X[y!=2]
y = y[y!=2]

# 分類用にサポートベクトルマシン (Support Vector Classifier) を用意
clf = svm.SVC(C=1.0, kernel='linear')
# データに最適化
clf.fit(X, y)

##### 分類結果を背景の色分けにより表示

# 外花被片の長さ(sepal length)と幅(sepal width)の
# 最小値と最大値からそれぞれ1ずつ広げた領域を
# グラフ表示エリアとする
x_min = min(X[:,0]) - 1
x_max = max(X[:,0]) + 1
y_min = min(X[:,1]) - 1
y_max = max(X[:,1]) + 1

```

scikit-learnからdatasetsとsvmモジュールを利用可能に

特徴量4→2　⇒　X[: , :2]　　Xに上書き

クラス3→2　⇒　X[y!=2]　y[y!=2]

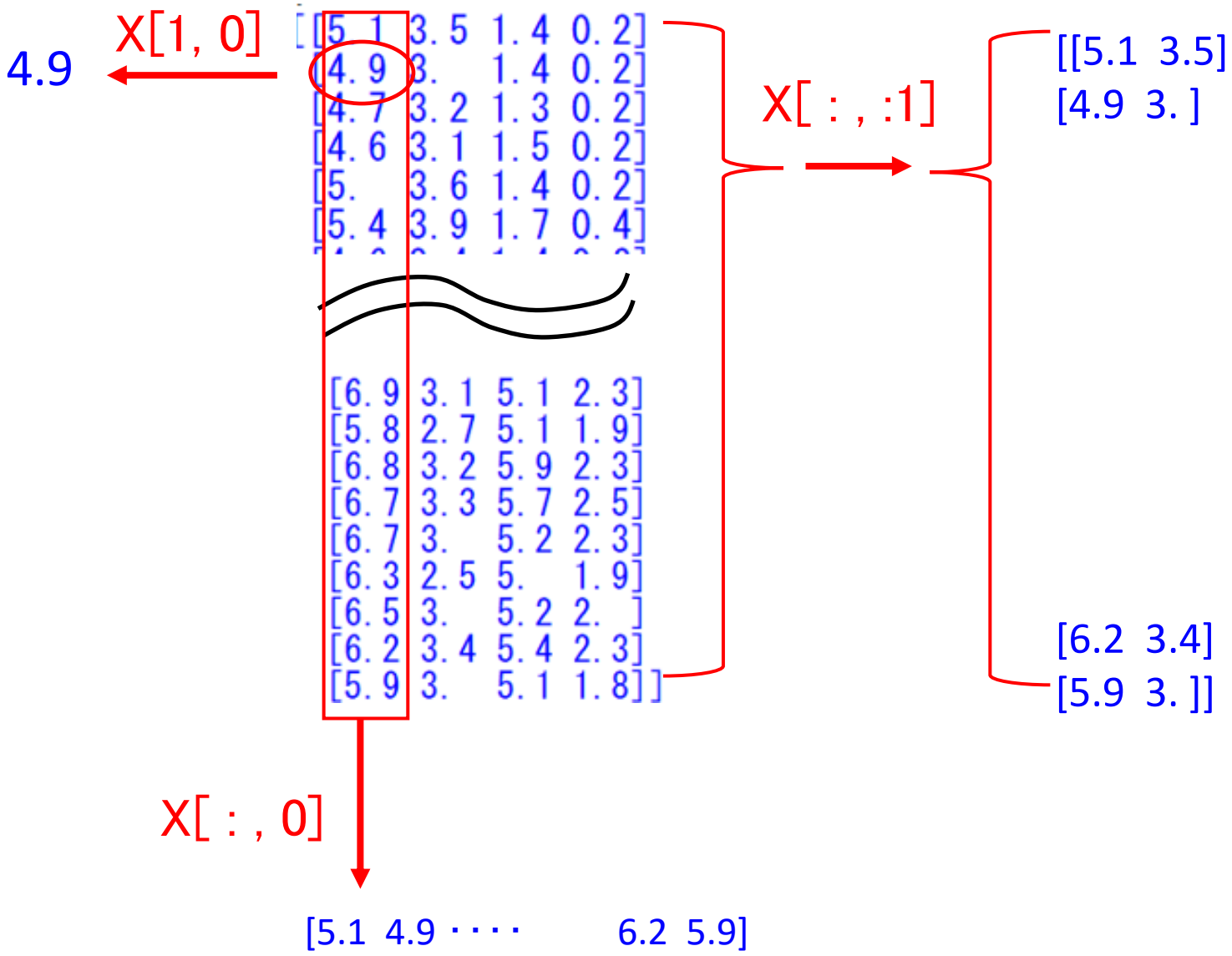
y!=2　はyが2に等しくない

サポートベクトルマシンの分類器SVCを呼び出し、
変数clfに格納。

SVC(Support Vector Machine for Classification)

パラメータC=1.0　kernel='linear'
Cが大きくなるとﾍﾟﾅﾙﾃｨｰが大、1.0はﾃﾞﾌｫﾙﾄ値

分類器のXとyが渡され、境界が決まる



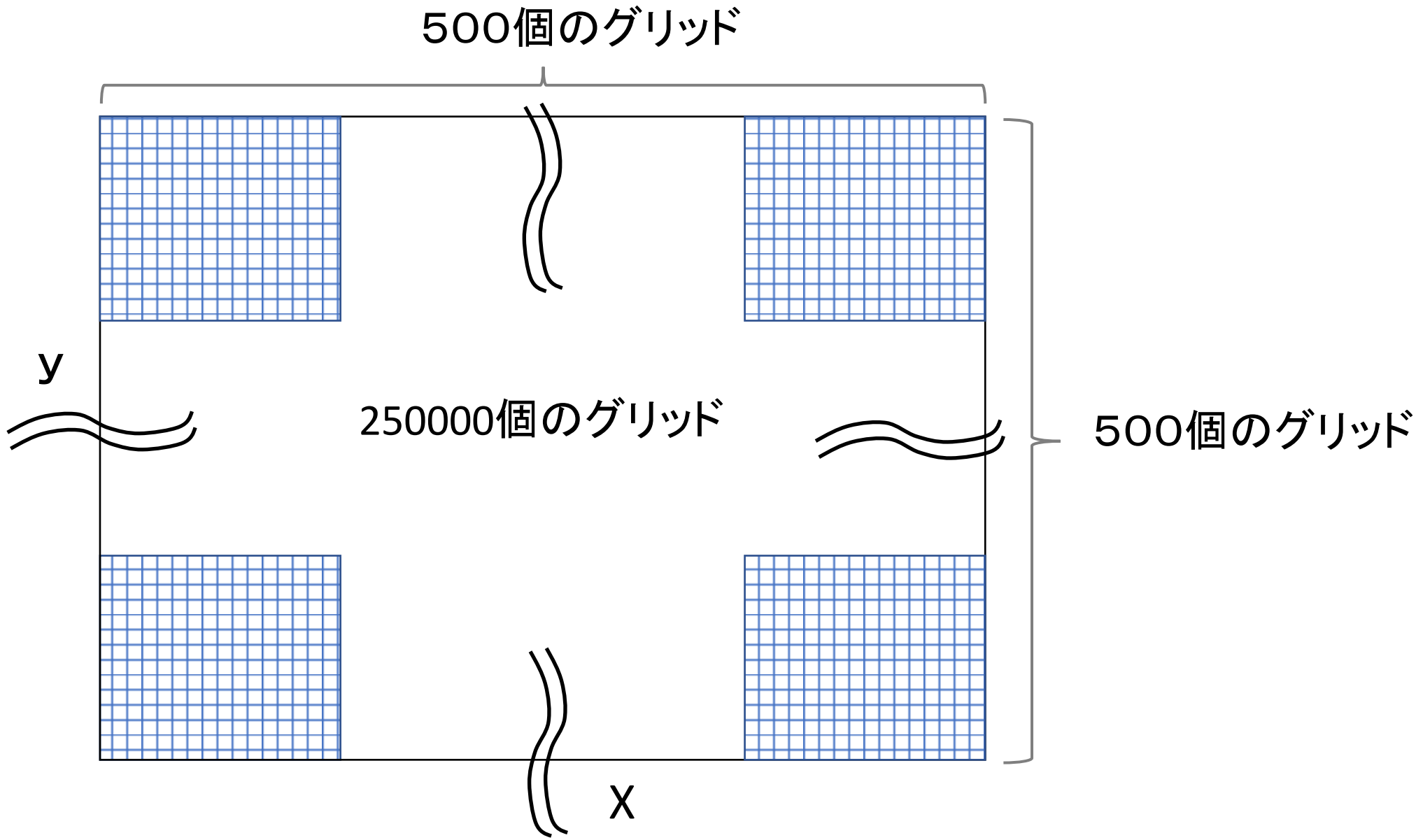
Numpy配列

X[0:150,0:3] と表す

150と3は含まず

0～149、0～2の範囲です

X[: , :]と省略することもある


500個のグリッド
y
250000個のグリッド
500個のグリッド
x

```
# グラフ表示エリアを縦横500ずつのグリッドに区切る
# (分類クラスに応じて背景に色を塗るため)
XX, YY = np.mgrid[x_min:x_max:500j, y_min:y_max:500j]

# グリッドの点をscikit-learn用の入力に並べなおす
Xg = np.c_[XX.ravel(), YY.ravel()]

# 各グリッドの点が属するクラス(0か1)の予測をZに格納
Z = clf.predict(Xg)

# Zをグリッド上に並べなおす
Z = Z.reshape(XX.shape)

# クラス0 (iris setosa) が薄オレンジ (1, 0.93, 0.5, 1)
# クラス1 (iris versicolor) が薄青 (0.5, 1, 1, 1)
cmap01 = ListedColormap([(0.5, 1, 1, 1), (1, 0.93, 0.5, 1)])

# 背景の色を表示
plt.pcolormesh(XX, YY, Z==0, cmap=cmap01)

# 軸ラベルを設定
plt.xlabel('sepal length')
plt.ylabel('sepal width')

##### ターゲットに応じた色付きでデータ点を表示

# iris setosa (y=0) のデータのみを取り出す
Xc0 = X[y==0]
# iris versicolor (y=1) のデータのみを取り出す
Xc1 = X[y==1]

# iris setosa のデータXc0をプロット
plt.scatter(Xc0[:,0], Xc0[:,1], c='#E69F00', linewidths=0.5, edgecolors='black')
# iris versicolor のデータXc1をプロット
plt.scatter(Xc1[:,0], Xc1[:,1], c='#56B4E9', linewidths=0.5, edgecolors='black')

```

250000個のNumpy配列Xgがどちらのクラスになるかを予測してZに格納

2つの色を定義

(赤、緑、青、不透明度の強さ)

Z==0の時オレンジ、それ以外は水色

クラス0(iris setosa)に属する点を集める

クラス1(iris versicolor)に属する点を集める

オレンジ色

Xc0をオレンジ色の点で描画

水色

Xc1を水色の点で描画

```
75 # サポートベクトルを取得
76 SV = clf.support_vectors_
77 # サポートベクトルの点に対し、赤い枠線を表示
78 svgraph = plt.scatter(SV[:,0], SV[:,1], linewidths=1.0, edgecolors='red')
79 svgraph.set_facecolor((0,0,0,0))
80
81 # 描画したグラフを表示
82 plt.show()
83
```

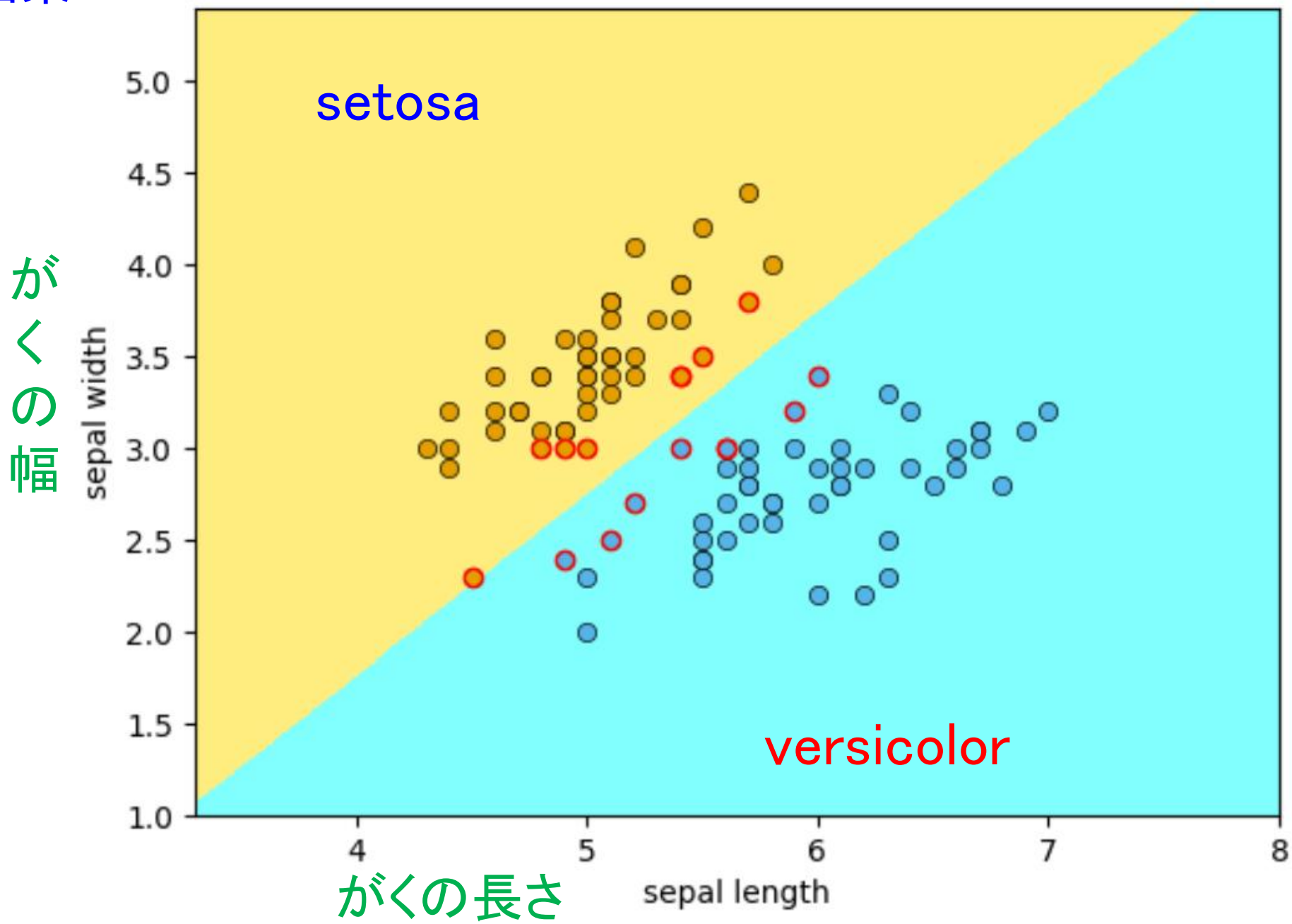
サポートベクトルを取り出し

サポートベクトルを赤い枠を表示

塗りつぶしの色として不透明度が0の透明色を指定

実行結果

setosaとversicolor



がくの幅

がくの長さ