第３回　機械学習

機械学習を実践してみよう！！

本教材を使用した際にはお手数ですが、
下記アンケートフォームにご協力下さい。

https://forms.gle/cgej2DL5PvneRhCp8

統合教育機構
須藤毅顕

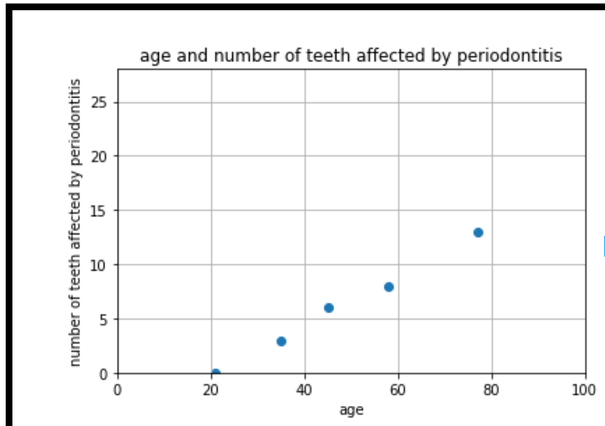
前回までの教師あり機械学習の流れ

①データの準備

No.	年齢	歯周病の歯の本数
1	35	3
2	21	0
3	45	6
4	58	8
5	77	13

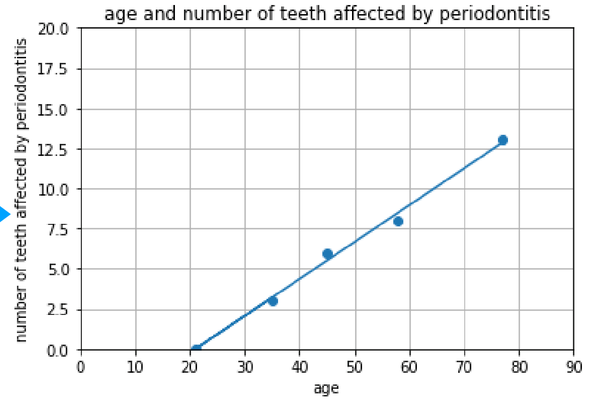
x = (説明変数)
y = (目的変数)

②学習モデルの決定



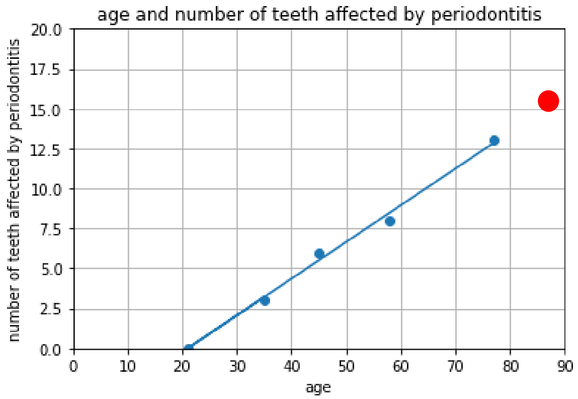
model = LinearRegression()

③学習



model.fit(x,y)

④予測、分類など



model.predict(x,y)

教師あり機械学習ではこのように教師データをもとにいずれかのモデル(ここでは線形回帰)
を適用して学習させ、予測や分類を行う

１）ライブラリの準備とデータの読み込み

新規ファイルを作成し、ファイルを保存しましょう(enshu3.py)
実行場所を設定しましょう(iryoAI)

```
import sklearn
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib import rcParams
rcParams['font.family'] ='sans-serif’
rcParams['font.sans-serif'] = ['Hiragino Maru Gothic Pro', 'Yu Gothic', 'Meirio’]
# 3.csvの読み込み
iris = pd.read_csv(“3.csv", encoding="utf-8")
```

１）ライブラリの準備とデータの読み込み

今回読み込んだ3.csvのデータ（アヤメのデータです）

iris - DataFrame

Index	がく片の長さ	がく片の幅	花びらの長さ	花びらの幅	アヤメの種類	アヤメの種類(0,1,2)
0	5.1	3.5	1.4	0.2	ヒオウギアヤメ	0
1	4.9	3	1.4	0.2	ヒオウギアヤメ	0
2	4.7	3.2	1.3	0.2	ヒオウギアヤメ	0
3	4.6	3.1	1.5	0.2	ヒオウギアヤメ	0
4	5	3.6	1.4	0.2	ヒオウギアヤメ	0
5	5.4	3.9	1.7	0.4	ヒオウギアヤメ	0
6	4.6	3.4	1.4	0.3	ヒオウギアヤメ	0
7	5	3.4	1.5	0.2	ヒオウギアヤメ	0
8	4.4	2.9	1.4	0.2	ヒオウギアヤメ	0
9	4.9	3.1	1.5	0.1	ヒオウギアヤメ	0
10	5.4	3.7	1.5	0.2	ヒオウギアヤメ	0
11	4.8	3.4	1.6	0.2	ヒオウギアヤメ	0
12	4.8	3	1.4	0.1	ヒオウギアヤメ	0
13	4.3	3	1.1	0.1	ヒオウギアヤメ	0
14	5.8	4	1.2	0.2	ヒオウギアヤメ	0

Format Resize Background color Column min/max Save and Close Close

前回までの教師あり機械学習の流れ(2)

線形単回帰
1次直線で近似して連続変数yを予測する

```
df = iris[0:100]
x1 = df[['がく片の長さ']]
y1 = df['がく片の幅']
from sklearn.linear_model import LinearRegression
model1 = LinearRegression()
model1.fit(x1,y1)
print(model1.predict([[6.5]]))
```

前回までの教師あり機械学習の流れ(2)

線形単回帰
1次直線で近似して連続変数yを予測する

```
df = iris[0:100]
x1 = df[['がく片の長さ']]
y1 = df['がく片の幅']
from sklearn.linear_model import LinearRegression
model1 = LinearRegression()
model1.fit(x1,y1)
print(model1.predict([[6.5]]))
```

ロジスティック回帰
ロジスティック関数で近似して(2値分類(0 or 1)を分類する

```
df = iris[0:100]
x2 = df[['がく片の長さ']]
y2 = df['アヤメの種類(0,1,2)']
from sklearn.linear_model import LogisticRegression
model2 = LogisticRegression()
model2.fit(x2, y2)
print(model2.predict([[6.5]]))
```

前回までの教師あり機械学習の流れ(2)

線形単回帰
1次直線で近似して連続変数yを予測する

```
df = iris[0:100]
x1 = df[['がく片の長さ']]
y1 = df['がく片の幅']
from sklearn.linear_model import LinearRegression
model1 = LinearRegression()
model1.fit(x1,y1)
print(model1.predict([[6.5]]))
```

ロジスティック回帰
ロジスティック関数で近似して(2値分類(0 or 1)を分類する

```
df = iris[0:100]
x2 = df[['がく片の長さ']]
y2 = df['アヤメの種類(0,1,2)']
from sklearn.linear_model import LogisticRegression
model2 = LogisticRegression()
model2.fit(x2, y2)
print(model2.predict([[6.5]]))
```

がく片の長さが6.5の時の
がく片の幅

```
[2.94091385]
```

```
In [16]: print(df)
    がく片の長さ  がく片の幅  花びらの長さ  花びらの幅      アヤメの種類  アヤメの種類(0,1,2)
0        5.1     3.5      1.4     0.2  ヒオウギアヤメ                0
1        4.9     3.0      1.4     0.2  ヒオウギアヤメ                0
2        4.7     3.2      1.3     0.2  ヒオウギアヤメ                0
3        4.6     3.1      1.5     0.2  ヒオウギアヤメ                0
4        5.0     3.6      1.4     0.2  ヒオウギアヤメ                0
..       ...     ...      ...     ...         ...              ...
95       5.7     3.0      4.2     1.2  ブルーフラッグ                1
96       5.7     2.9      4.2     1.3  ブルーフラッグ                1
97       6.2     2.9      4.3     1.3  ブルーフラッグ                1
98       5.1     2.5      3.0     1.1  ブルーフラッグ                1
99       5.7     2.8      4.1     1.3  ブルーフラッグ                1
```

がく片の長さが6.5の時の
分類結果

[1]

predict_probaであれば
確率

前回までの教師ありデータの準備

学習データ

	がく片の長さ	がく片の幅	花びらの長さ	花びらの幅	アヤメの種類
0	5.1	3.5	1.4	0.2	ヒオウギアヤメ
1	4.9	3.0	1.4	0.2	ヒオウギアヤメ
2	4.7	3.2	1.3	0.2	ヒオウギアヤメ
3	4.6	3.1	1.5	0.2	ヒオウギアヤメ
4	5.0	3.6	1.4	0.2	ヒオウギアヤメ
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	バージニカ
146	6.3	2.5	5.0	1.9	バージニカ
147	6.5	3.0	5.2	2.0	バージニカ
148	6.2	3.4	5.4	2.3	バージニカ
149	5.9	3.0	5.1	1.8	バージニカ

	がく片の長さ	がく片の幅	花びらの長さ	花びらの幅
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

＋

アヤメの種類
ヒオウギアヤメ
ヒオウギアヤメ
ヒオウギアヤメ
ヒオウギアヤメ
ヒオウギアヤメ
...
バージニカ
バージニカ
バージニカ
バージニカ
バージニカ

特徴量データ(説明変数)　　正解データ(目的変数)

```
iris = pd.read_csv("iris.csv", encoding="utf-8")
```

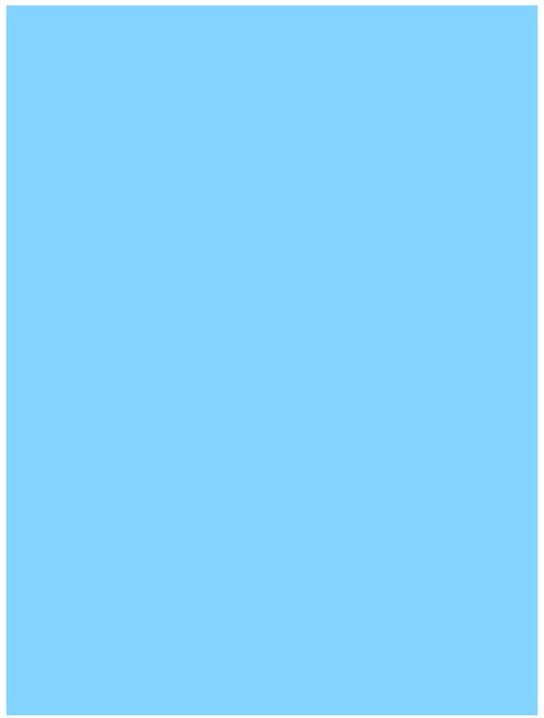
```
x = iris[['がく片の長さ','がく片の幅','花びらの長さ','花びらの幅']]
y = iris['アヤメの種類']
```

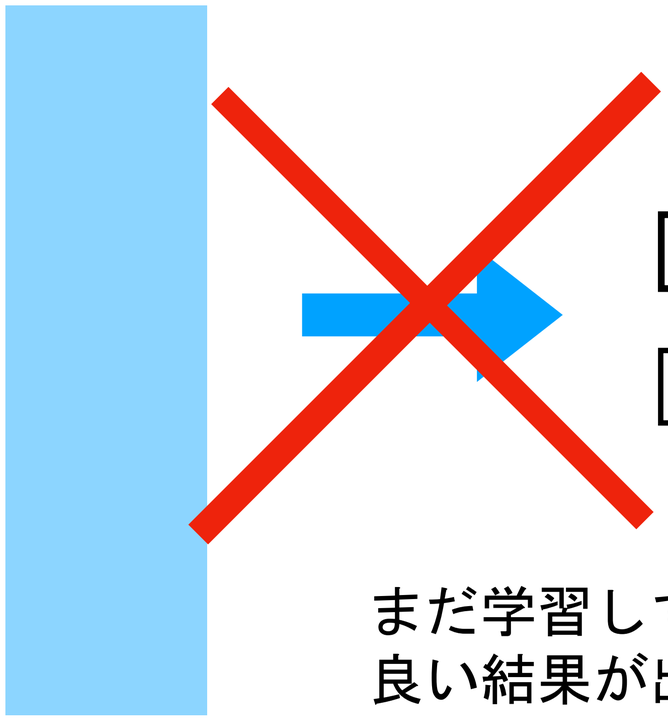
機械学習ではそのままデータを丸ごと学習させない！

→そのままだと実力よりも良すぎる正解率が出る可能性(過学習)
(偏ったデータの可能性を否定するため)

x(特徴量データ)

y(正解データ)



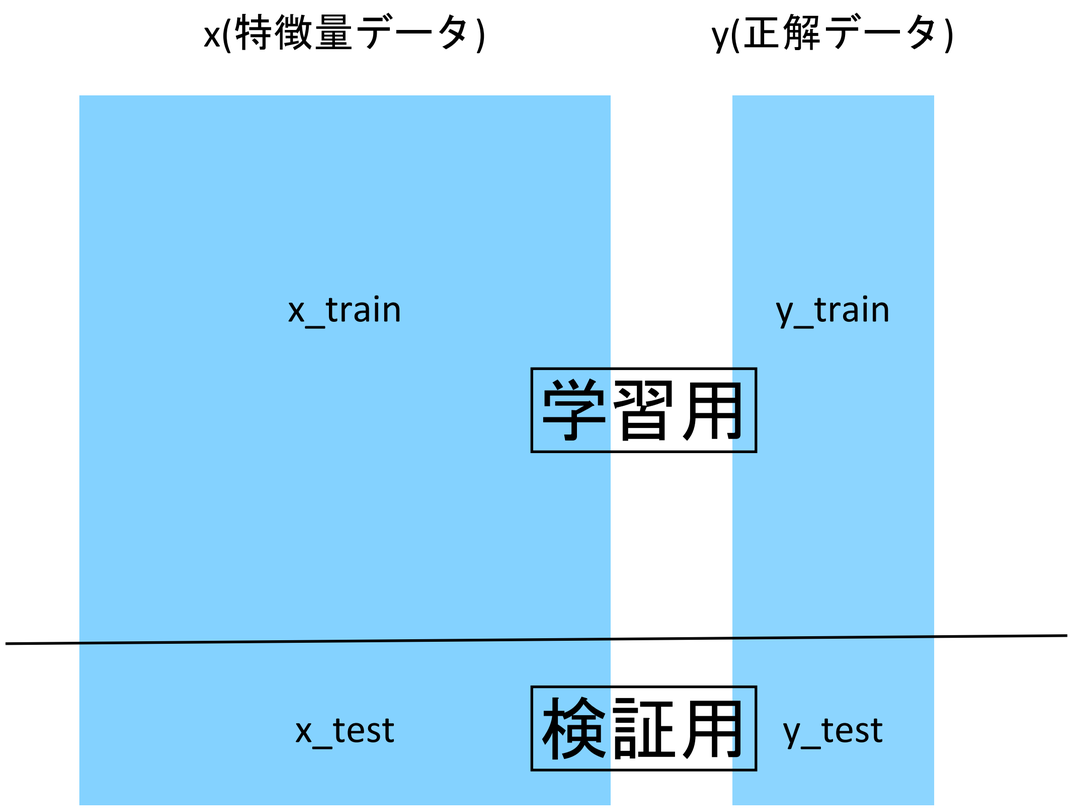


```
model = LinearRegression()
```

```
model.fit(x,y)
```

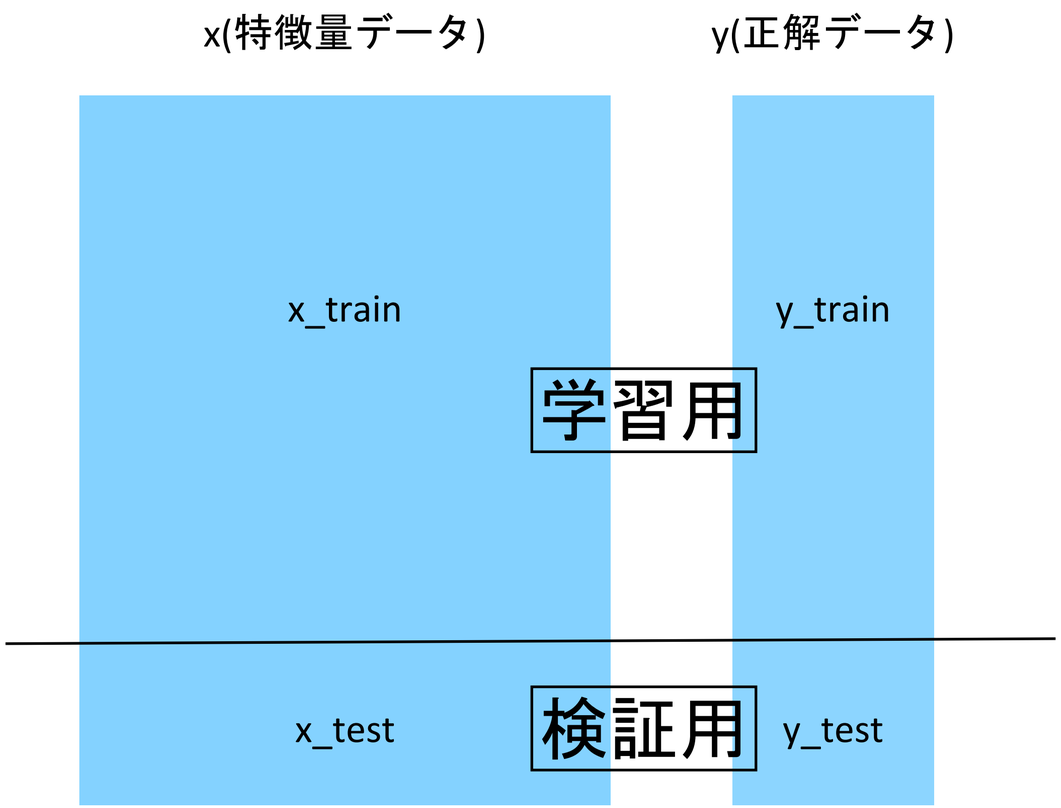
まだ学習していない未知のデータでも
良い結果が出るかどうか検証用データも必要

ホールドアウト法



新たにデータを用意するのではなく、
全データを学習用と検証用に分割する
(20~30%で分割するのが一般的)

ホールドアウト法



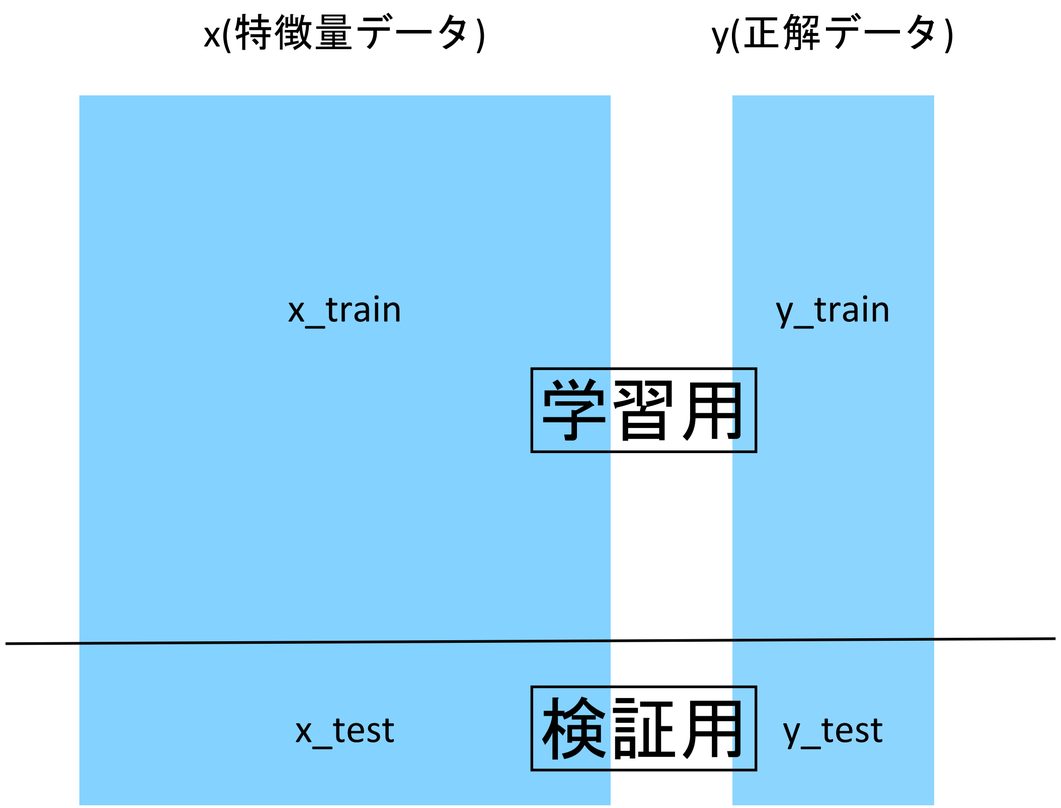
新たにデータを用意するのではなく、
全データを学習用と検証用に分割する
(20~30%で分割するのが一般的)

train_test_split : 学習用データと検証用データに分ける命令文
(下はxとyを検証用データを0.3の割合で分けるように指示している)

x_train, x_test, y_train, y_test
= train_test_split(x,y, test_size=0.3, random_state=0)

x_train： 特徴量の学習用
x_test： 特徴量の検証用
y_train： 正解データの学習用
y_test： 正解データの検証用

ホールドアウト法



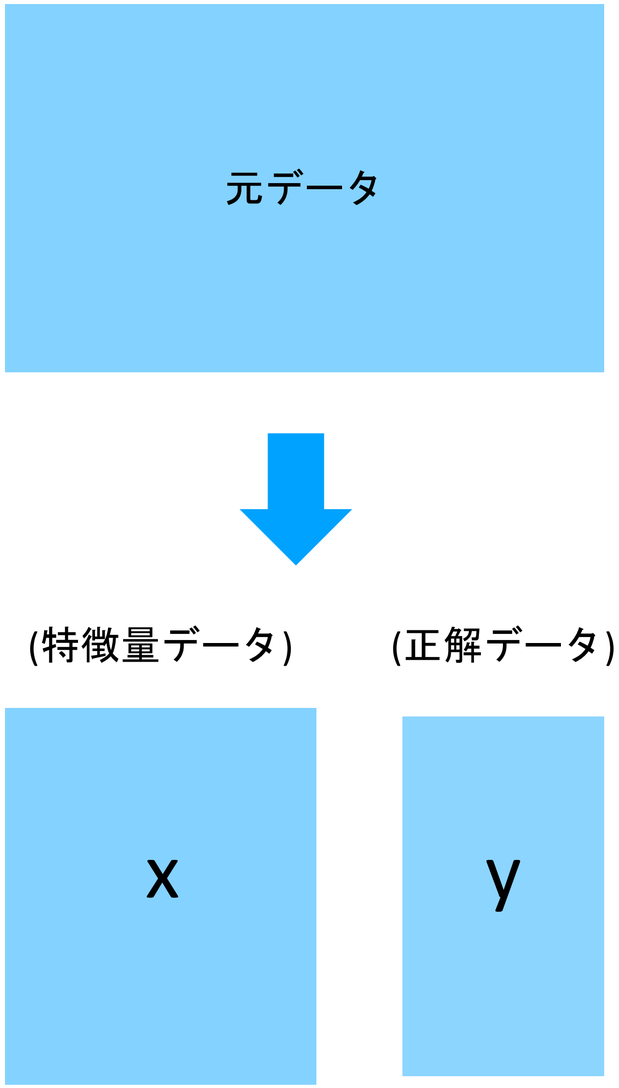
新たにデータを用意するのではなく、
全データを学習用と検証用に分割する
(20~30%で分割するのが一般的)

train_test_split : 学習用データと検証用データに分ける命令文
(下はxとyを検証用データを0.3の割合で分けるように指示している)

x_train, x_test, y_train, y_test
= train_test_split(x,y, test_size=0.3, random_state=0)

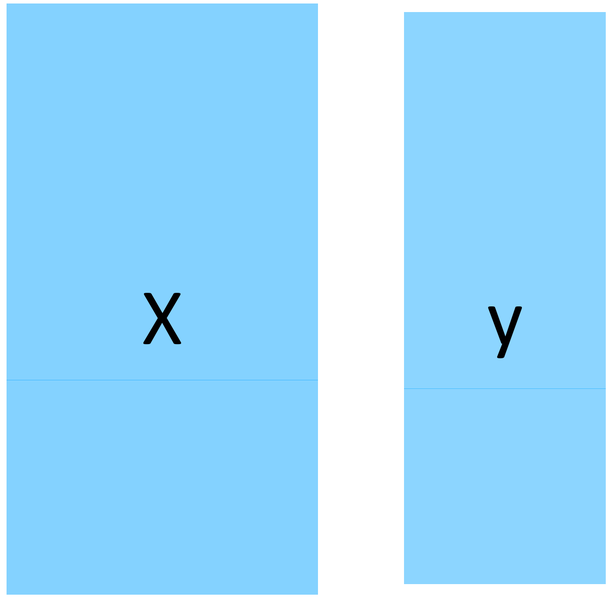
x_train： 特徴量の学習用
x_test： 特徴量の検証用
y_train： 正解データの学習用
y_test： 正解データの検証用

機械学習の流れ(もう少し詳しく)



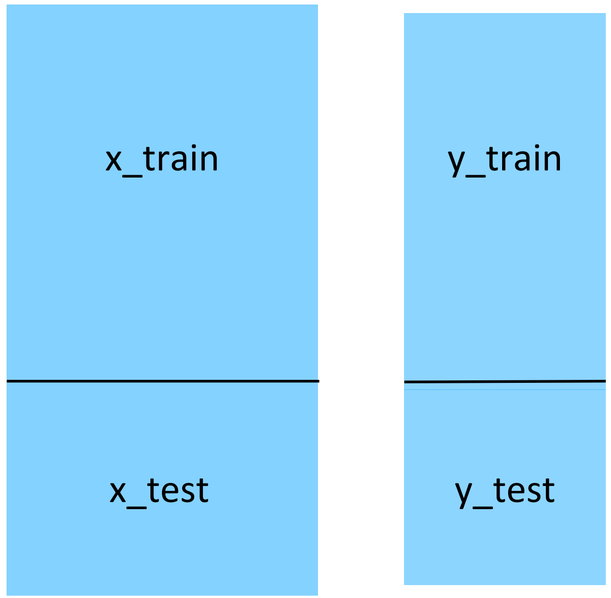
機械学習の流れ(もう少し詳しく)

x(特徴量データ)　　y(正解データ)

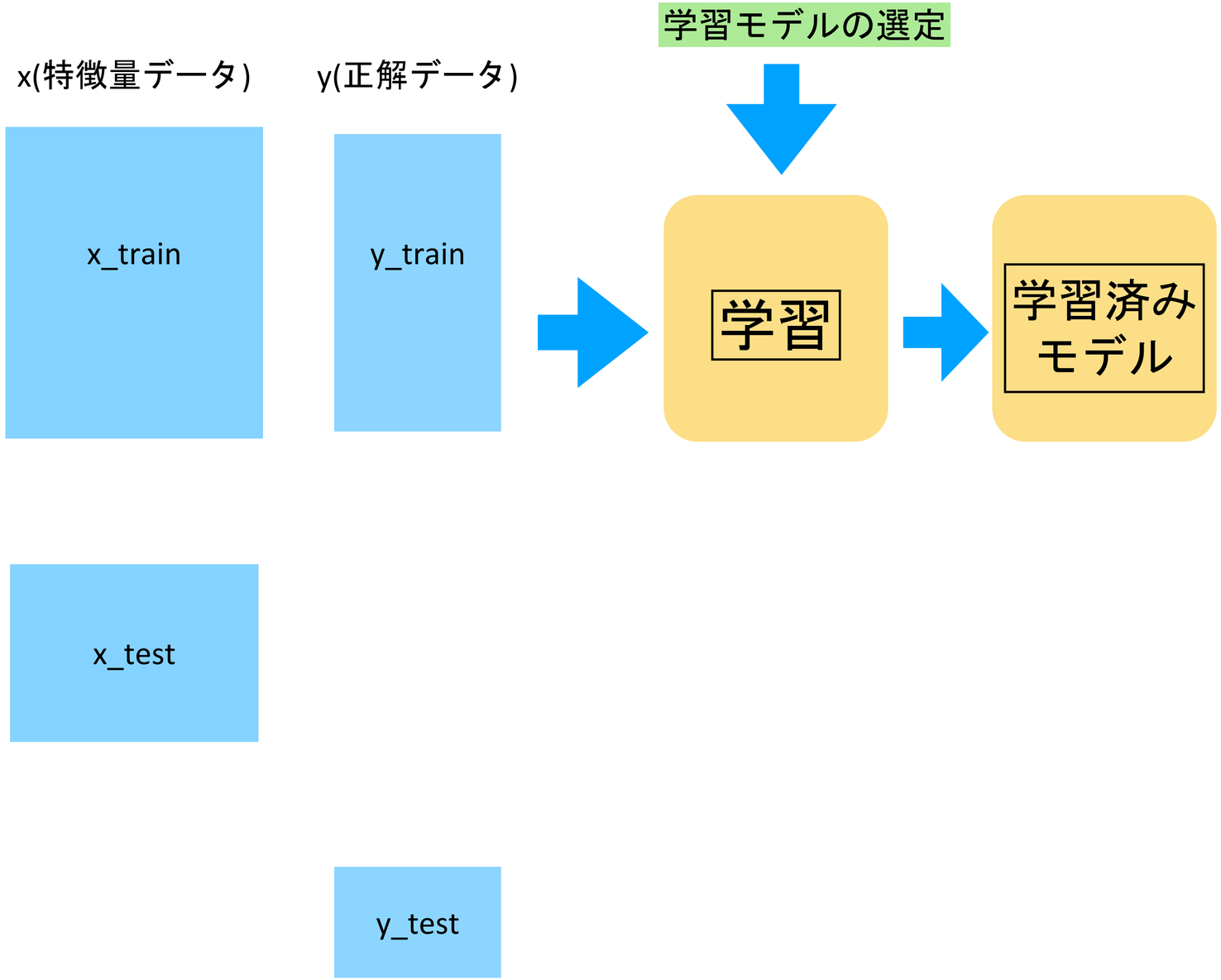


機械学習の流れ(もう少し詳しく)

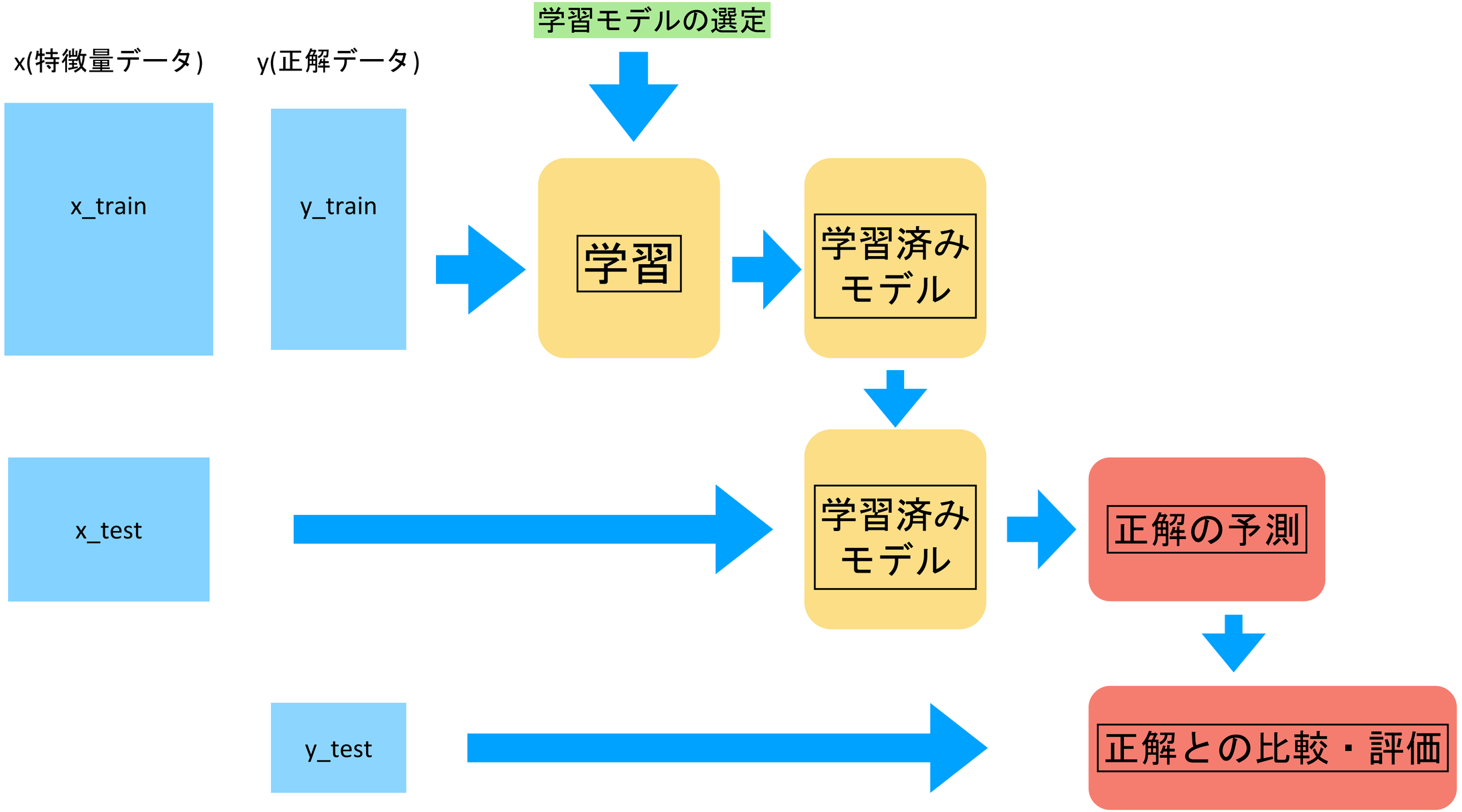
x(特徴量データ)　　y(正解データ)



機械学習の流れ(もう少し詳しく)



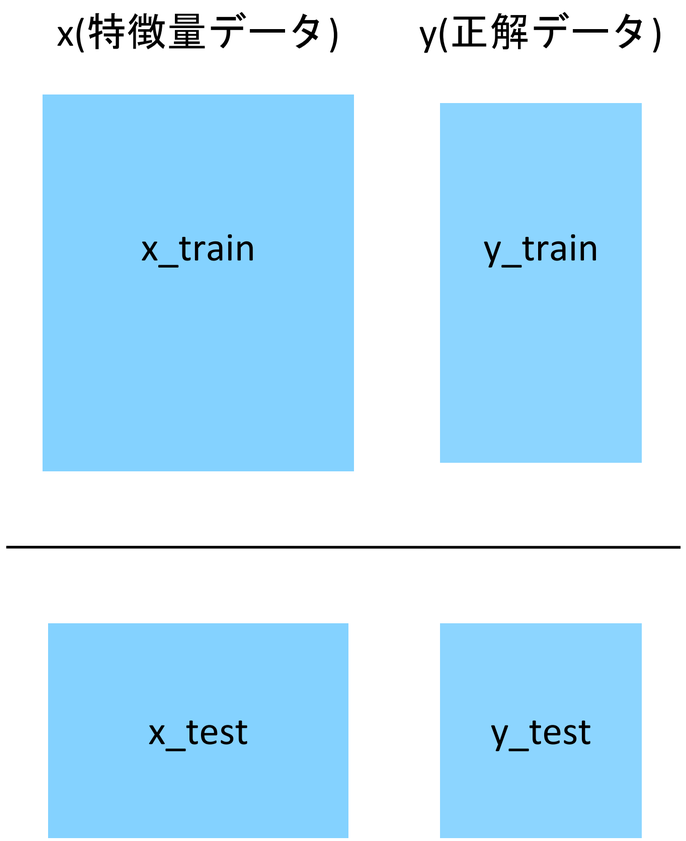
機械学習の流れ(もう少し詳しく)



3)全データを学習用と検証用に分割して学習①

```
# ロジスティック回帰はロジスティック関数で近似して(2値分類(0 or 1)を分類する

df = iris[0:100]
x = df[['がく片の長さ']]
y = df['アヤメの種類(0,1,2)']
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state=0)
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
```

3)全データを学習用と検証用に分割して学習①



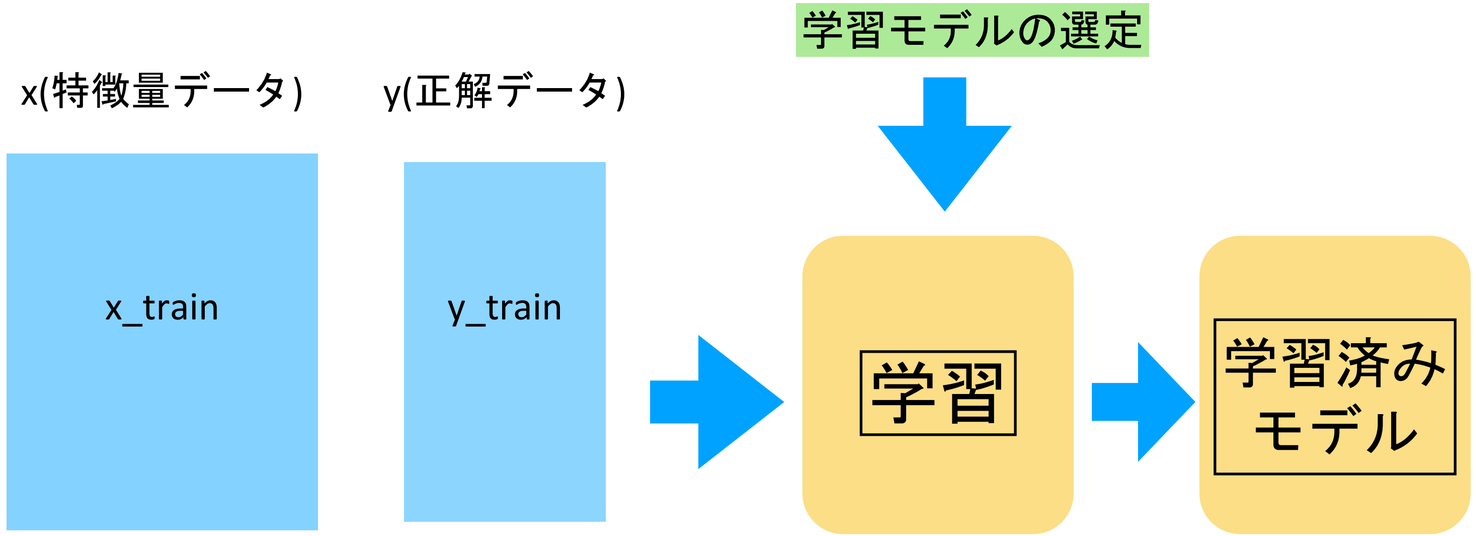
①データを特徴量データ(x)と正解データ(y)に分ける

```
df = iris[0:100]
x = df[['がく片の長さ']]
y = df['アヤメの種類(0,1,2)']
```

②それぞれ学習用データ(train)と検証用データ(test)に分ける

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
                                    x, y, test_size = 0.3, random_state=0)
```

3)全データを学習用と検証用に分割して学習①



学習用データを用いて学習させる

```
#モデルの選択
(変数名) = 学習モデル()
#選択したモデルの学習
(変数名).fit(x_train, y_train)
```

ロジスティック回帰

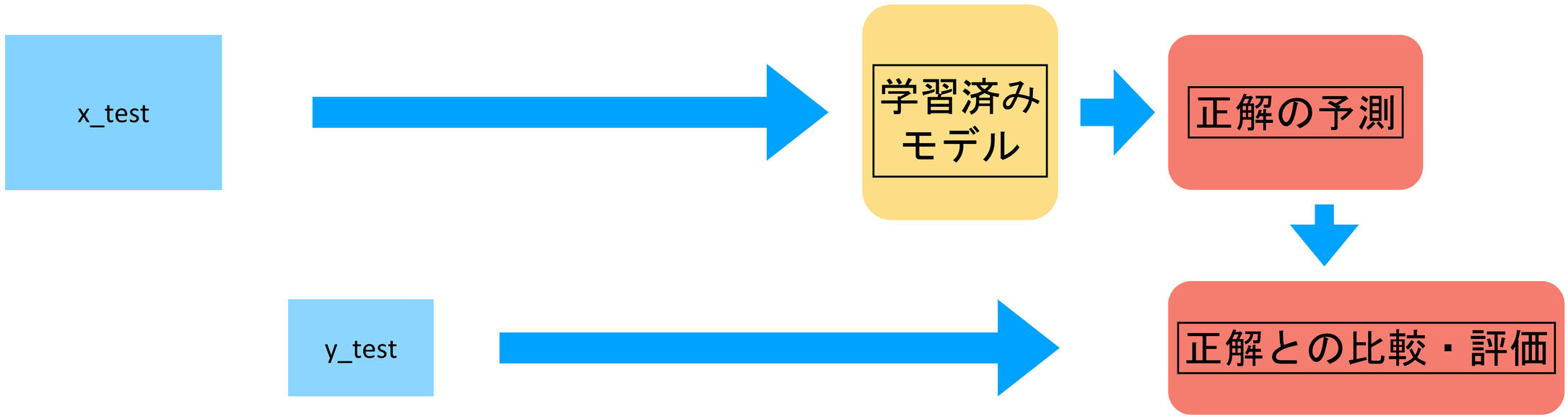
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(x_train, y_train)
```

全データを学習用と検証用に分割して学習②

検証用データを用いて学習結果を評価する

x_test(特徴量データ)を用いて正解を予測して、y_test(実際の正解)との違いを比較・評価する

```
print(model.predict(x_test))
print(np.array(y_test))
```



全データを学習用と検証用に分割して学習②

検証用データを用いて学習結果を評価する

In

```
print(x_test)
print(y_test)
```

Out

```
print(x_test)
    がく片の長さ
26     5.0
86     6.7
2      4.7
55     5.7
75     6.6
93     5.0
16     5.4
73     6.1
54     6.5
95     5.7
53     5.5
92     5.8
78     6.0
13     4.3
7      5.0
30     4.8
22     4.6
24     4.8
33     5.5
8      4.4
43     5.0
62     6.0
3      4.6
71     6.1
45     4.8
48     5.3
6      4.6
99     5.7
82     5.8
76     6.8
```

```
c print(y_test)
26    0
86    1
2     0
55    1
75    1
93    1
16    0
73    1
54    1
95    1
53    1
92    1
78    1
13    0
7     0
30    0
22    0
24    0
33    0
8     0
43    0
62    1
3     0
71    1
45    0
48    0
6     0
99    1
82    1
76    1
```

全データを学習用と検証用に分割して学習②

検証用データを用いて学習結果を評価する

In

```
print(x_test)
print(y_test)
```

In

```
print("テストデータの分類結果")
print(model.predict(x_test))
print("正解ラベル")
print(np.array(y_test))
```

Out

```
テストデータの分類結果
[0 1 0 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 1 0 0 1 0 1 0 0 0 1 1 1]

正解ラベル
[0 1 0 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1 1]
```

Out

```
print(x_test)
   がく片の長さ
26    5.0
86    6.7
2     4.7
55    5.7
75    6.6
93    5.0
16    5.4
73    6.1
54    6.5
95    5.7
53    5.5
92    5.8
78    6.0
13    4.3
7     5.0
30    4.8
22    4.6
24    4.8
33    5.5
8     4.4
43    5.0
62    6.0
3     4.6
71    6.1
45    4.8
48    5.3
6     4.6
99    5.7
82    5.8
76    6.8
```

```
c print(y_test)
26   0
86   1
2    0
55   1
75   1
93   1
16   0
73   1
54   1
95   1
53   1
92   1
78   1
13   0
7    0
30   0
22   0
24   0
33   0
8    0
43   0
62   1
3    0
71   1
45   0
48   0
6    0
99   1
82   1
76   1
```

５）学習モデルの評価

ロジスティック回帰のような分類モデルでは、
model.score()で正解率を求めて性能を評価できる

```
print(model.score(x_test, y_test))
```

５）学習モデルの評価

ロジスティック回帰のような分類モデルでは、
model.score()で正解率を求めて性能を評価できる

```
print(model.score(x_test, y_test))
```

```
0.9333333333333333
```

93%

５）学習モデルの評価

混同行列を用いる

		予測結果	
		positive (正)	Negative (負)
実際の分類結果	positive (正)	真陽性 True Positive	偽陰性 False Negative
	Negative (負)	偽陽性 False Positive	真陰性 True Negative

正解率

$$\frac{\text{真陽性} + \text{真陰性}}{\text{真陽性} + \text{偽陽性} + \text{真陰性} + \text{偽陰性}}$$

他の分類モデルでも試してみよう

５）学習モデルの評価

混同行列を用いる

```
from sklearn.metrics import confusion_matrix
conf = confusion_matrix(y_test, model.predict(x_test))
print(conf)
```

```
[[14  1]
 [ 1 14]]
```

		予測結果	
		positive (正)	Negative (負)
実際の分類結果	positive (正)	14	1
	Negative (負)	1	14

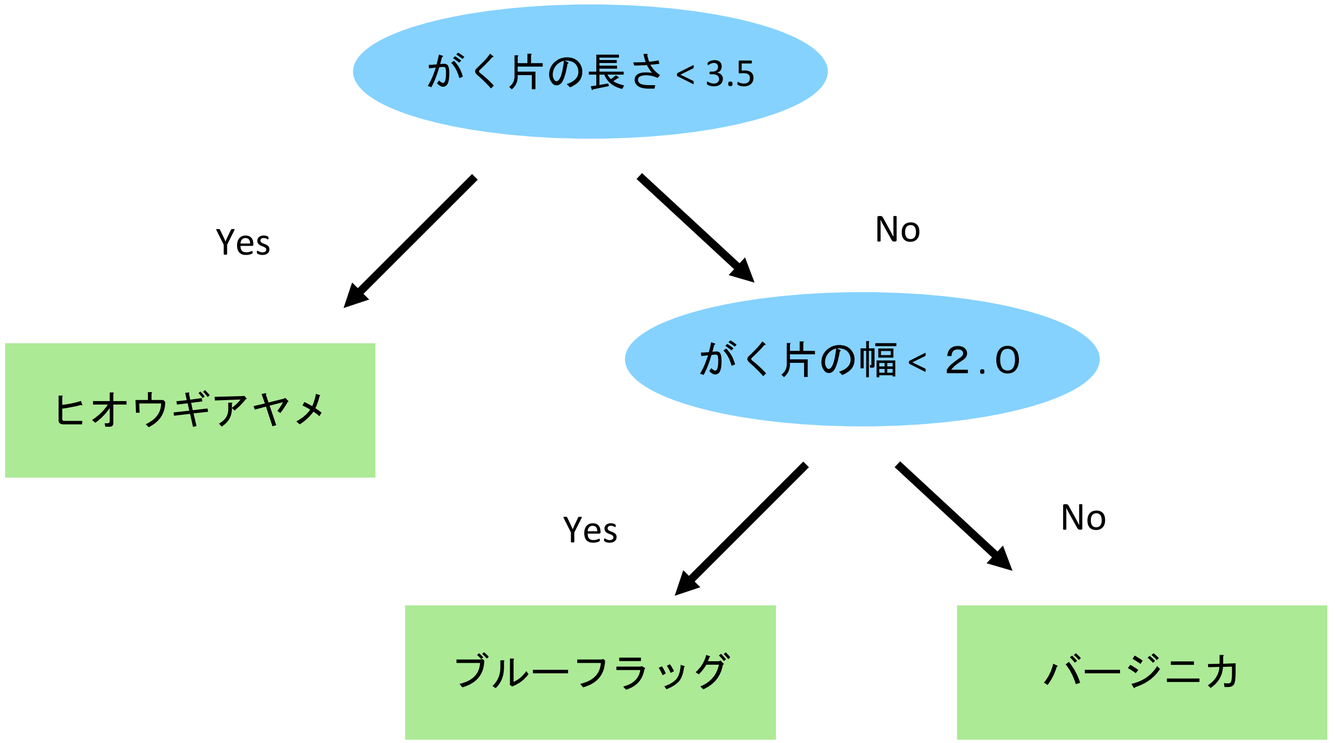
正解率

$$\frac{\text{真陽性} + \text{真陰性}}{\text{真陽性} + \text{偽陽性} + \text{真陰性} + \text{偽陰性}}$$

(14 + 14) / (14 +1 + 14 + 1)
= 0.93333

他の分類モデルでも試してみよう

決定木(decision tree)



決定木(decision tree)は特徴量の列の内容をもとに条件分岐をすることで分類する手法

6) 決定木(decision tree)

```
# 6) 決定木を実践してみよう

# xに4つの特徴量(説明変数)、yに正解ラベル(目的変数)
x2 = iris[['がく片の長さ','がく片の幅','花びらの長さ','花びらの幅']]
y2 = iris['アヤメの種類']
x2_train, x2_test, y2_train, y2_test = train_test_split(x2, y2, test_size = 0.3, random_state=0)

# 決定木(max_depth=2)
from sklearn import tree
model2 = tree.DecisionTreeClassifier(max_depth=2, random_state=0)

# model2.fit()でモデルの学習
model2.fit(x2_train,y2_train)
# model2.score()で学習済みモデルの正解率計算
print(model2.predict(x2_test))
print(model2.score(x2_test,y2_test))
print(np.array(y2_test))
```

ロジスティック回帰と違うのは
ここだけ

5) 決定木(decision tree)

```
print(model2.predict(x2_test))
['バージニカ''ブルーフラッグ''ヒオウギアヤメ''バージニカ''ヒオウギアヤメ''バージニカ''ヒオウギアヤメ''ブルーフラッグ'
 'ブルーフラッグ''ブルーフラッグ''バージニカ''ブルーフラッグ''ブルーフラッグ''ブルーフラッグ''ブルーフラッグ'
 'ヒオウギアヤメ''ブルーフラッグ''ブルーフラッグ''ヒオウギアヤメ''ヒオウギアヤメ''ブルーフラッグ''ブルーフラッグ'
 'ヒオウギアヤメ''ヒオウギアヤメ''ブルーフラッグ''ヒオウギアヤメ''ヒオウギアヤメ''ブルーフラッグ''ブルーフラッグ'
 'ヒオウギアヤメ''バージニカ''ブルーフラッグ''ヒオウギアヤメ''ブルーフラッグ''バージニカ''ブルーフラッグ'
 'ヒオウギアヤメ''バージニカ''ブルーフラッグ''ブルーフラッグ''バージニカ''ヒオウギアヤメ''バージニカ''ヒオウギアヤメ'
 'ヒオウギアヤメ']

print(model2.score(x2_test,y2_test))
0.9111111111111111

print(np.array(y2_test))
['バージニカ''ブルーフラッグ''ヒオウギアヤメ''バージニカ''ヒオウギアヤメ''バージニカ''ヒオウギアヤメ''ブルーフラッグ'
 'ブルーフラッグ''ブルーフラッグ''バージニカ''ブルーフラッグ''ブルーフラッグ''ブルーフラッグ''ブルーフラッグ'
 'ヒオウギアヤメ''ブルーフラッグ''ブルーフラッグ''ヒオウギアヤメ''ヒオウギアヤメ''バージニカ''ブルーフラッグ'
 'ヒオウギアヤメ''ヒオウギアヤメ''バージニカ''ヒオウギアヤメ''ヒオウギアヤメ''ブルーフラッグ''ブルーフラッグ'
 'ヒオウギアヤメ''バージニカ''ブルーフラッグ''ヒオウギアヤメ''バージニカ''バージニカ''ブルーフラッグ'
 'ヒオウギアヤメ''ブルーフラッグ''ブルーフラッグ''ブルーフラッグ''バージニカ''ヒオウギアヤメ''バージニカ''ヒオウギアヤメ'
 'ヒオウギアヤメ']
```

5) 決定木(decision tree)の混同行列

```
conf2 = confusion_matrix(y2_test, model2.predict(x2_test))
print(conf2)
```

```
[[ 8  0  3]
 [ 0 16  0]
 [ 1  0 17]]
```

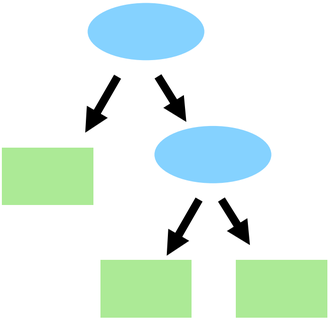
予測

正解		バージニカ	ヒオウギアヤメ	ブルーフラッグ
	バージニカ	8	0	3
	ヒオウギアヤメ	0	16	0
	ブルーフラッグ	1	0	17

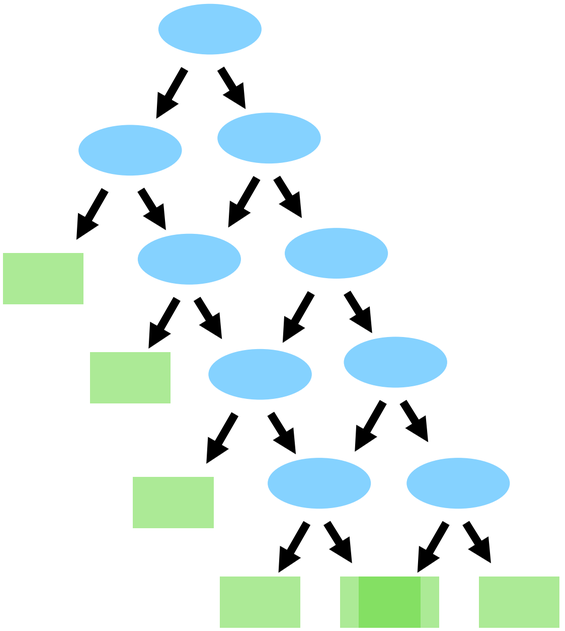
$$\frac{(8 + 16 + 17)}{(8 + 0 + 3 + 0 + 16 + 0 + 1 + 0 + 17)} = 0.91111$$

7) max_depthでフローチャートの数を増やすことが出来る

```
# 決定木(max_depth=2)
from sklearn import tree
model2 = tree.DecisionTreeClassifier(max_depth=2, random_state=0)

# model2.fit()でモデルの学習
model2.fit(x2_train,y2_train)
# model2.score()で学習済みモデルの正解率計算
print(model2.predict(x2_test))
print(model2.score(x2_test,y2_test))
print(np.array(y2_test))
```



```
# 決定木(max_depth=5)
from sklearn import tree
model3 = tree.DecisionTreeClassifier(max_depth=5, random_state=0)

# model3.fit()でモデルの学習
model3.fit(x2_train,y2_train)
# model3.score()で学習済みモデルの正解率計算
print(model3.predict(x2_test))
print(model3.score(x2_test,y2_test))
```



0.9777777777777777

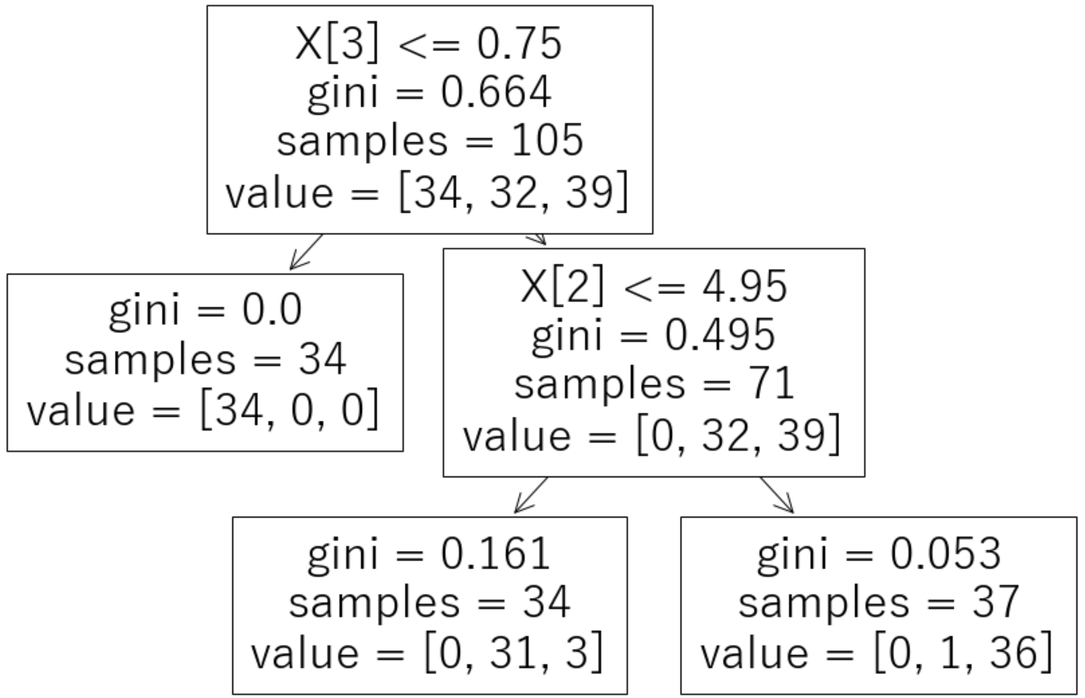
精度が上がっている

8)決定木の図示

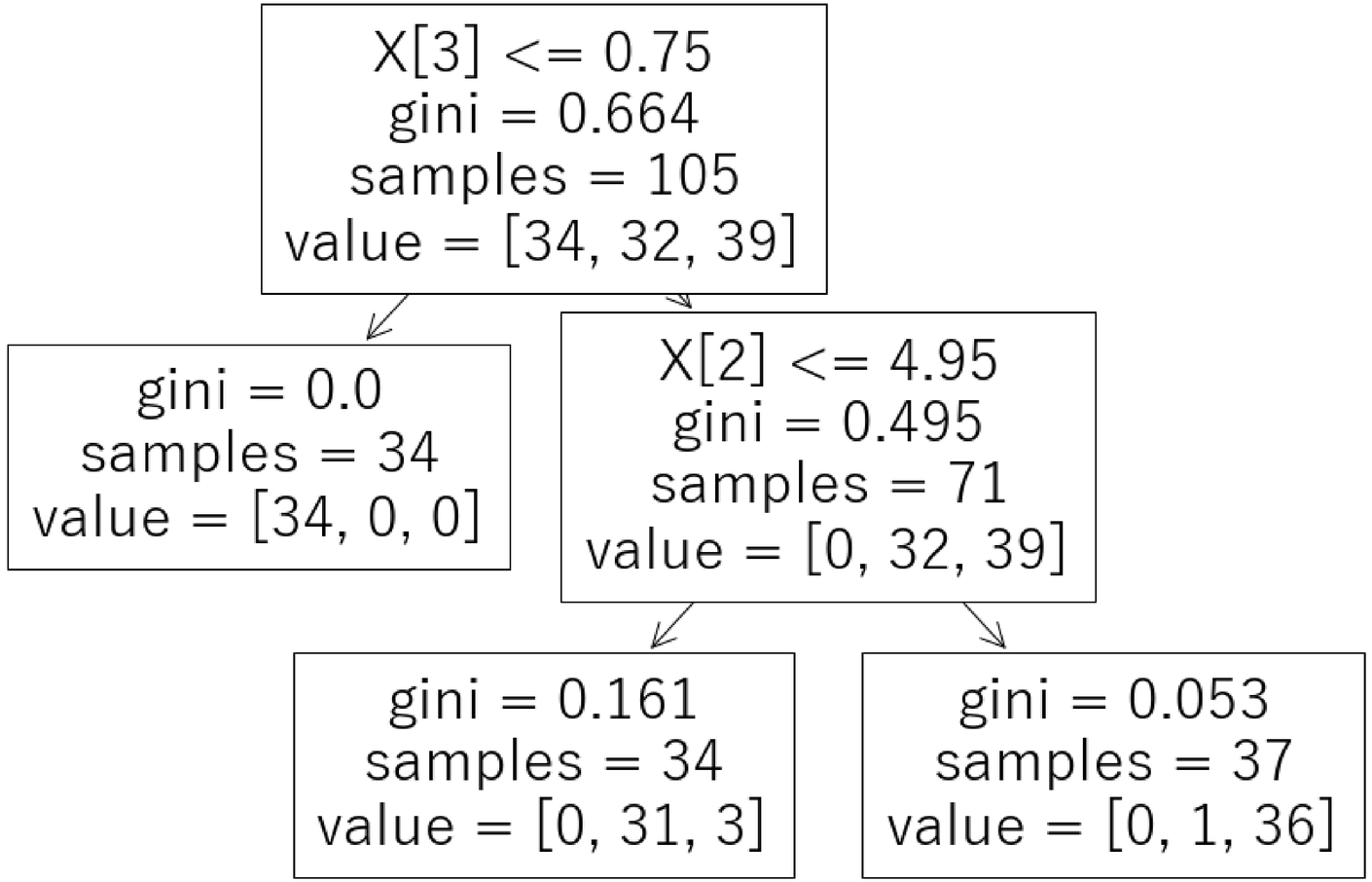
```
# 8) 決定木の図示

x2 = iris[['がく片の長さ','がく片の幅','花びらの長さ','花びらの幅']]
y2 = iris['アヤメの種類(0,1,2)']
x2_train, x2_test, y2_train, y2_test = train_test_split(x2, y2, test_size = 0.3, random_state=0)
model2 = tree.DecisionTreeClassifier(max_depth=2, random_state=0)
model2.fit(x2_train,y2_train)

plt.figure(figsize=(15,10))
tree.plot_tree(model2)
plt.show()
```

8) 決定木の図示

```
# 8) 決定木の図示

x2 = iris[['がく片の長さ','がく片の幅','花びらの長さ','花びらの幅']]
y2 = iris['アヤメの種類(0,1,2)']
x2_train, x2_test, y2_train, y2_test = train_test_split(x2, y2, test_size = 0.3, random_state=0)
model2 = tree.DecisionTreeClassifier(max_depth=2, random_state=0)
model2.fit(x2_train,y2_train)

plt.figure(figsize=(15,10))
tree.plot_tree(model2)
plt.show()
```



8) 決定木の図示



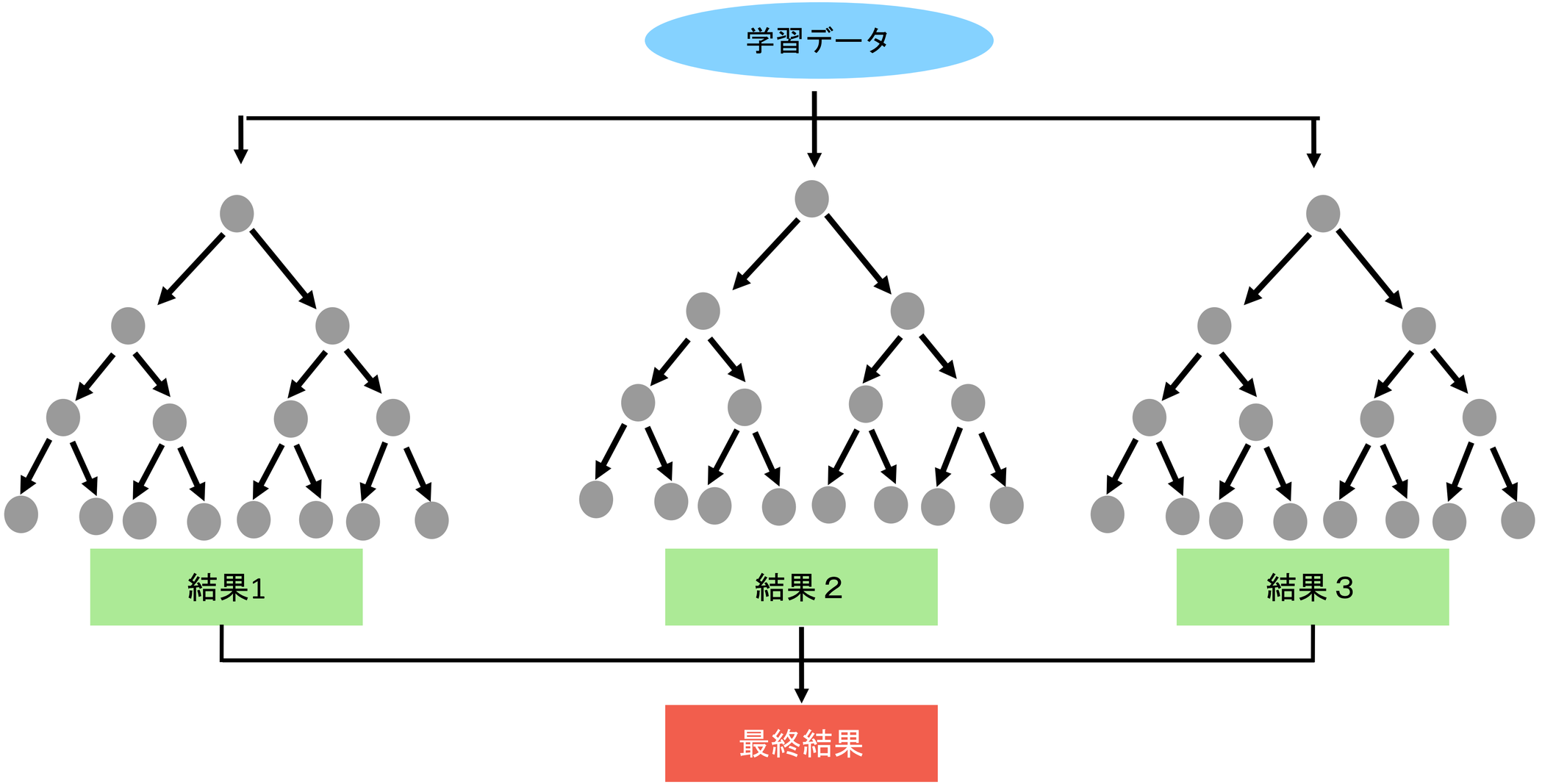
X[0]: がく片の長さ
X[1]: がく片の幅
X[2]: 花びらの長さ
X[3]: 花びらの幅

gini : ジニ係数
誤分類する確率の期待値
(≒誤分類をどれくらいしてしまいそうかの指標)

samples: 150 × 0.7 = 105

Value = [(ヒオウギアヤメ),
(ブルーフラッグ),
(バージニカ)]

9) ランダムフォレスト



ランダムフォレスト(random forests)は複数の分類木を用いて最終的な予測結果を出す手法

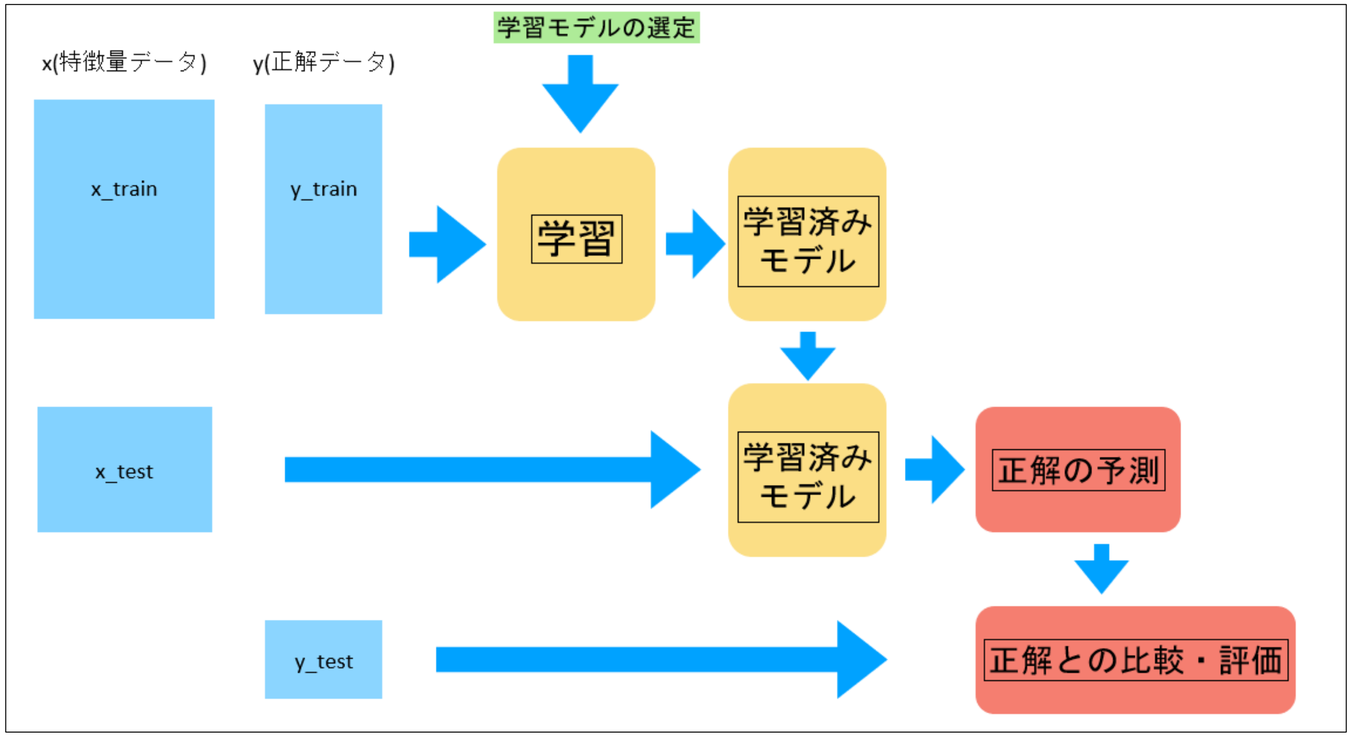
9) ランダムフォレスト

```
# 8) そのほかの教師あり機械学習の実践(ランダムフォレスト)
x = iris[['がく片の長さ','がく片の幅','花びらの長さ','花びらの幅']]
y = iris['アヤメの種類']
x4_train, x4_test, y4_train, y4_test = train_test_split(x, y, test_size = 0.3, random_state=0)

from sklearn.ensemble import RandomForestClassifier
model4 = RandomForestClassifier()
# model4.fit()でモデルの学習
# model4.score()で学習済みモデルの正解率計算
model4.fit(x4_train,y4_train)
print(model4.score(x4_test,y4_test))
```

出力結果：

```
0.9777777777777777
```

まとめ




```
x = (説明変数)
y = (目的変数)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state=0)
model = (使いたい学習モデル)
model.fit(x_train,y_train)
model.score(x_test,y_test)
```

課題

ランダムフォレストをテストサイズを0.1、0.2、0.5で
変えてみて結果を出力してください

ランダムフォレストで作ったモデルに、
がく片の長さと幅：3.3, 4.2
花びらの長さと幅：4.4, 5.7
時のブルーフラッグの確率を求めてください
(テストサイズは0.2)

＋

ここまでのアンケートも課題として提出をお願いします。

アンケート内容

・社会で起きている変化を理解し、数理・データサイエンス・AIを学ぶことの意義を説明できる。
（出来る、少しは出来る、出来ない）

・AIを活用した新しいビジネス/サービスを説明できる
（出来る、少しは出来る、出来ない）

・どんなデータが集められ、どう活用されているかを説明できる。
（出来る、少しは出来る、出来ない）

・データ・AIを活用するために使われている技術を概説できる。
（出来る、少しは出来る、出来ない）

・データ・AIを活用することによって、どの様な価値が生まれているかを説明できる。
（出来る、少しは出来る、出来ない）

・データ・AI利活用における最新動向(ビジネスモデル、テクノロジー)を説明できる。
（出来る、少しは出来る、出来ない）

・データ利活用プロセスを体験し、データを解析して考察できる。
（出来る、少しは出来る、出来ない）

・これまでの授業の率直な感想をお聞かせください