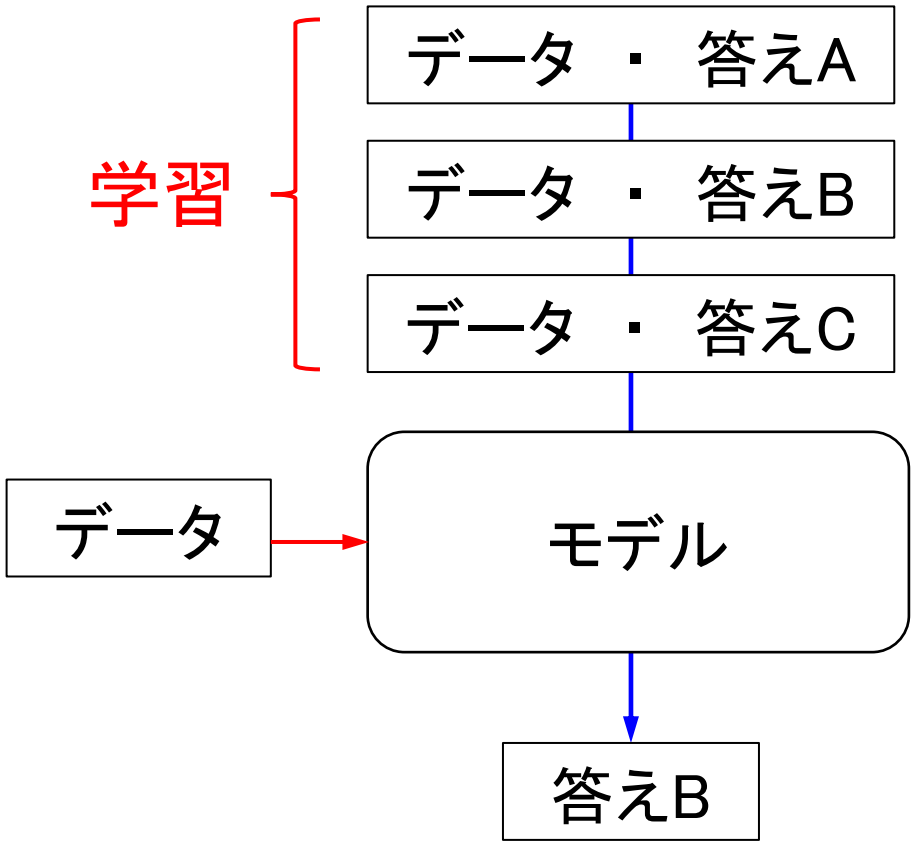
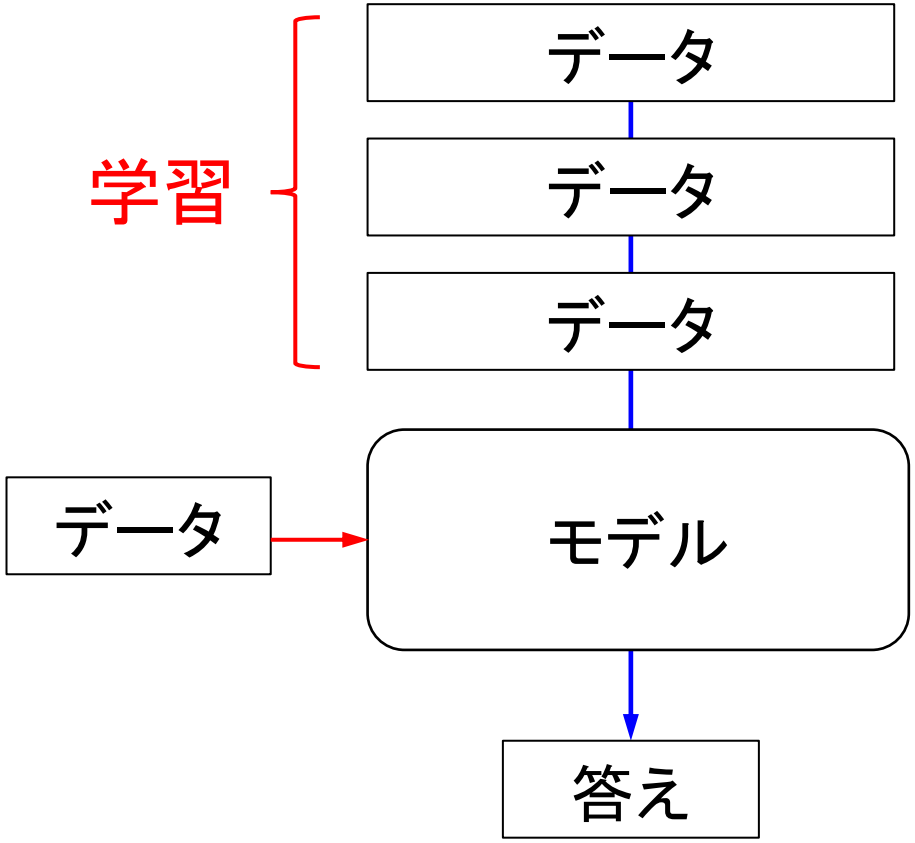
scikit-learnによる機械学習

機械学習＝データを学習→ 一定のパターン抽出→ 新たなデータの結果予測
モデル

教師あり　元データの答えあり



教師なし　元データに答えなし



①scikit-learnのインストール

Windows　Powershell等で＞の後にpy　-m pip install scikit-learnを入力

②Jupyter notebookを起動

③irisデータロード

```
from sklearn.datasets import load_iris
iris=load_iris()
print(iris.data.shape)
print(iris.data[:10])
print(iris.target_names)
```

150個のデータに4個ずつの値

最初のデータ10行を表示

予測されるデータ名称

```
(150, 4)
[[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]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
['setosa' 'versicolor' 'virginica']
```

↑ 0　↑ 1　↑ 2

iris.data　　：irisデータ
iris.target　　：教師用データ(データと答えのリスト)
iris.target_name ：targetの答え

```
from sklearn.model_selection import train_test_split

(train_X,test_X,train_Y,test_Y)=train_test_split(iris.data,iris.target,test_size=0.2)
print(iris.target_names[test_Y])
print(test_Y)
print(test_X)
```

```
['virginica' 'virginica' 'setosa' 'virginica' 'versicolor' 'versicolor'
 'setosa' 'virginica' 'setosa' 'virginica' 'setosa' 'virginica'
 'virginica' 'virginica' 'versicolor' 'virginica' 'versicolor' 'virginica'
 'virginica' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'virginica'
 'versicolor' 'virginica' 'versicolor' 'virginica' 'virginica']
[2 2 0 2 1 1 0 2 0 2 0 2 2 2 1 2 1 2 2 0 0 0 0 0 2 1 2 1 2 2]
```

予測用教師データの名称

←予測用教師データ(答え)

変数＝train_test_split(データ、教師データ、test_size=0.2)

2割を予測
8割は学習用

予測用データ

```
[[6.4 2.8 5.6 2.2]
 [6.8 3.  5.5 2.1]
 [5.2 3.4 1.4 0.2]
 [6.7 3.3 5.7 2.1]
 [6.7 3.  5.  1.7]
 [6.8 2.8 4.8 1.4]
 [5.1 3.4 1.5 0.2]
 [4.9 2.5 4.5 1.7]
 [4.9 3.1 1.5 0.2]
 [6.  2.2 5.  1.5]
 [4.6 3.4 1.4 0.3]
 [5.8 2.7 5.1 1.9]
 [6.5 3.  5.5 1.8]
 [6.7 3.3 5.7 2.5]
 [5.5 2.4 3.8 1.1]
 [6.3 2.8 5.1 1.5]
 [5.9 3.  4.2 1.5]
 [7.7 2.8 6.7 2. ]
 [7.9 3.8 6.4 2. ]
 [4.7 3.2 1.3 0.2]
 [4.6 3.6 1.  0.2]
 [5.1 3.7 1.5 0.4]
 [4.4 3.2 1.3 0.2]
 [5.1 3.3 1.7 0.5]
 [6.7 2.5 5.8 1.8]
 [6.2 2.9 4.3 1.3]
 [5.6 2.8 4.9 2. ]
 [5.6 3.  4.5 1.5]
 [6.5 3.  5.8 2.2]
 [6.4 2.8 5.6 2.1]]
```

変数：　train_X　学習用データ
　　test_X　予測用データ
　　train_Y　学習用教師データ
　　test_Y　　予測用教師データ

Digitsデータ

```
from sklearn.datasets import load_digits
digits=load_digits()
print(digits.data)
print(digits.data.shape)
print(digits.target)
print(digits.target_names)
```

```
[[ 0.   0.   5. ...   0.   0.   0.]
 [ 0.   0.   0. ...  10.   0.   0.]
 [ 0.   0.   0. ...  16.   9.   0.]
 ...
 [ 0.   0.   1. ...   6.   0.   0.]
 [ 0.   0.   2. ...  12.   0.   0.]
 [ 0.   0.  10. ...  12.   1.   0.]]
(1797, 64)
[0 1 2 ... 8 9 8]
[0 1 2 3 4 5 6 7 8 9]
```

1797個のデータに各々８×８＝64個の値

Target（教師データ）は0～9の整数値

```python
from sklearn.model_selection import train_test_split
(train_X,test_X,train_Y,test_Y)=train_test_split(digits.data,digits.target,test_size=0.2,random_state=0)

from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier()
model.fit(train_X,train_Y)
print(model)

from sklearn import metrics
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
pred =model.predict(test_X)
score=accuracy_score(test_Y,pred)
print('score:%s' % score)
print (pred[:20])
print(test_Y[:20])
print(classification_report(test_Y,pred))
print(confusion_matrix(test_Y,pred))
```

データ割り振り

K近傍法というモデルでフィット

←予測

scores： 精度（正解率）

```
KNeighborsClassifier()
score:0.975
[2 8 2 6 6 7 1 9 8 5 2 8 6 6 6 6 1 0 5 8]
[2 8 2 6 6 7 1 9 8 5 2 8 6 6 6 6 1 0 5 8]
```

クラス分けレポート

	Precision 予測中の正解率	recall 正解中の予測正解膣	f1-score 前2者の調和平均	support サンプル数
0	1.00	1.00	1.00	27
1	0.97	0.97	0.97	35
2	1.00	0.97	0.99	36
3	0.91	1.00	0.95	29
4	1.00	0.97	0.98	30
5	0.95	0.97	0.96	40
6	1.00	1.00	1.00	44
7	0.95	1.00	0.97	39
8	1.00	0.90	0.95	39
9	0.98	0.98	0.98	41
accuracy			0.97	360
macro avg	0.98	0.98	0.98	360
weighted avg	0.98	0.97	0.97	360

対角が予測が
正解の部分
それ以外は不正解

予測結果の行列

```
[[27  0  0  0  0  0  0  0  0  0]
 [ 0 34  0  0  0  1  0  0  0  0]
 [ 0  0 35  1  0  0  0  0  0  0]
 [ 0  0  0 29  0  0  0  0  0  0]
 [ 0  0  0  0 29  0  0  1  0  0]
 [ 0  0  0  0  0 39  0  0  0  1]
 [ 0  0  0  0  0  0 44  0  0  0]
 [ 0  0  0  0  0  0  0 39  0  0]
 [ 0  1  0  2  0  0  0  1 35  0]
 [ 0  0  0  0  0  1  0  0  0 40]]
```

```
from sklearn.model_selection import train_test_split
(train_X,test_X,train_Y,test_Y)=train_test_split(digits.data,digits.target,test_size=0.2,random_state=0)
```

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier()
model.fit(train_X,train_Y)
print(model)
```

```
from sklearn import metrics
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
pred =model.predict(test_X)
score=accuracy_score(test_Y,pred)
print('score:%s' % score)
print (pred[:20])
print(test_Y[:20])
print(classification_report(test_Y,pred))
print(confusion_matrix(test_Y,pred))
```

K近傍法(KneighborsClassifier)

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier()
model.fit(train_X,train_Y)
print(model)
```

単純パーセプトロン

```
from sklearn.linear_model import Perceptron
model=Perceptron(max_iter=100)
model.fit(train_X,train_Y)
print(model)
```

ロジスティック回帰

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(train_X,train_Y)
print(model)
```

SVC（サポートベクターマシン）

```
from sklearn import svm
model=svm.SVC(C=100.,gamma=0.001)
model.fit(train_X,train_Y)
print(model)
```

多層パーセプトロン

```
from sklearn.neural_network import MLPClassifier
model=MLPClassifier()
model.fit(train_X,train_Y)
print(model)
```

単純パーセプトロン

```
from sklearn.linear_model import Perceptron
model=Perceptron(max_iter=100)
model.fit(train_X,train_Y)
print(model)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	27
1	0.86	0.91	0.89	35
2	0.97	0.97	0.97	36
3	1.00	0.93	0.96	29
4	0.94	1.00	0.97	30
5	0.95	0.97	0.96	40
6	0.96	1.00	0.98	44
7	1.00	0.95	0.97	39
8	0.78	0.90	0.83	39
9	1.00	0.78	0.88	41
accuracy			0.94	360
macro avg	0.95	0.94	0.94	360
weighted avg	0.94	0.94	0.94	360

K近傍法(KneighborsClassifier)

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier()
model.fit(train_X,train_Y)
print(model)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	27
1	0.97	0.97	0.97	35
2	1.00	0.97	0.99	36
3	0.91	1.00	0.95	29
4	1.00	0.97	0.98	30
5	0.95	0.97	0.96	40
6	1.00	1.00	1.00	44
7	0.95	1.00	0.97	39
8	1.00	0.90	0.95	39
9	0.98	0.98	0.98	41
accuracy			0.97	360
macro avg	0.98	0.98	0.98	360
weighted avg	0.98	0.97	0.97	360

ロジスティック回帰

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(train_X,train_Y)
print(model)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	27
1	0.92	0.97	0.94	35
2	0.97	0.97	0.97	36
3	0.97	1.00	0.98	29
4	0.97	0.97	0.97	30
5	0.97	0.93	0.95	40
6	1.00	0.98	0.99	44
7	0.97	0.97	0.97	39
8	0.97	0.92	0.95	39
9	0.93	0.98	0.95	41
accuracy			0.97	360
macro avg	0.97	0.97	0.97	360
weighted avg	0.97	0.97	0.97	360

多層パーセプトロン

```
from sklearn.neural_network import MLPClassifier
model=MLPClassifier()
model.fit(train_X,train_Y)
print(model)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	27
1	0.94	0.91	0.93	35
2	0.97	1.00	0.99	36
3	1.00	1.00	1.00	29
4	1.00	1.00	1.00	30
5	0.97	0.93	0.95	40
6	0.98	0.98	0.98	44
7	1.00	0.97	0.99	39
8	0.93	0.95	0.94	39
9	0.88	0.93	0.90	41
accuracy			0.96	360
macro avg	0.97	0.97	0.97	360
weighted avg	0.96	0.96	0.96	360

学習モデル	正解率
K近傍法	0.97
単純パーセプトロン	0.94
多層パーセプトロン	0.96
ロジスティック回帰	0.97
SVC	0.99

SVC

```
from sklearn import svm
model=svm.SVC(C=100., gamma=0.001)
model.fit(train_X,train_Y)
print(model)
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	27
1	0.97	1.00	0.99	35
2	1.00	1.00	1.00	36
3	1.00	1.00	1.00	29
4	1.00	1.00	1.00	30
5	0.97	0.97	0.97	40
6	1.00	1.00	1.00	44
7	1.00	1.00	1.00	39
8	1.00	0.97	0.99	39
9	0.98	0.98	0.98	41
accuracy			0.99	360
macro avg	0.99	0.99	0.99	360
weighted avg	0.99	0.99	0.99	360