Iris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)

予測データ表示

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)

デジタルデータ

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))

単純パーセプトロン

from sklearn.datasets import load\_digits

digits=load\_digits()

print(digits.data)

print(digits.data.shape)

print(digits.target)

print(digits.target\_names)

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.linear\_model import Perceptron

model=Perceptron(max\_iter=100)

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))

ロジスティック回帰

from sklearn.datasets import load\_digits

digits=load\_digits()

print(digits.data)

print(digits.data.shape)

print(digits.target)

print(digits.target\_names)

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.linear\_model import LogisticRegression

model=LogisticRegression()

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))

SVC

from sklearn.datasets import load\_digits

digits=load\_digits()

print(digits.data)

print(digits.data.shape)

print(digits.target)

print(digits.target\_names)

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 import svm

model=svm.SVC(C=100.,gamma=0.001)

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))

多層パーセプトロン

from sklearn.datasets import load\_digits

digits=load\_digits()

print(digits.data)

print(digits.data.shape)

print(digits.target)

print(digits.target\_names)

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.neural\_network import MLPClassifier

model=MLPClassifier()

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))