Machine learning HW03 111598067 王上澤

1. 一張含有 文字, 信 的圖片

   自動產生的描述
2. 一張含有 圖表 的圖片

   自動產生的描述
3. 一張含有 文字, 信 的圖片

   自動產生的描述一張含有 文字, 信 的圖片

   自動產生的描述一張含有 文字, 信 的圖片

   自動產生的描述一張含有 文字, 信 的圖片

   自動產生的描述一張含有 文字, 信 的圖片

   自動產生的描述
4. Code:

from sklearn import datasets, model\_selection, metrics

from sklearn.neighbors import KNeighborsClassifier

from sklearn.mixture import GaussianMixture

from numpy import average

import pandas as pd

import numpy as np

iris = datasets.load\_iris()

acc\_arr = []

iter\_times = 10

for i in range(iter\_times):

    # split the datasets

    data\_train, data\_test, target\_train, target\_test = model\_selection.train\_test\_split(

        iris.data, iris.target, test\_size=0.3, random\_state=i+22333333)

    # three classifier for each class

    Setosa\_classifier = GaussianMixture(n\_components=2, random\_state=i+22333333, max\_iter=100)

    Versicolour\_classifier = GaussianMixture(n\_components=2, random\_state=i+22333333, max\_iter=100)

    Virginica\_classifier = GaussianMixture(n\_components=2, random\_state=i+22333333, max\_iter=100)

    # fit the trainning data

    Setosa\_classifier.fit(data\_train[target\_train==0])

    Versicolour\_classifier.fit(data\_train[target\_train==1])

    Virginica\_classifier.fit(data\_train[target\_train==2])

    # get each class' score

    Setosa\_score = Setosa\_classifier.score\_samples(X=data\_test)

    Versicolour\_score = Versicolour\_classifier.score\_samples(X=data\_test)

    Virginica\_score = Virginica\_classifier.score\_samples(X=data\_test)

    # combine three classes' score

    combine\_pred = np.column\_stack((Setosa\_score, Versicolour\_score, Virginica\_score))

    # get total accuracy list by picking out the maxium score's from where they came from

    prediction = np.argmax(combine\_pred, axis=1)

    accur = metrics.accuracy\_score(target\_test, prediction)

    print("{iter} 's train accuracy : {acc}".format(iter=i, acc=accur))

    acc\_arr.append(accur)

print("average accuracy of {iter} times' GMM classification : {aver\_acc}".format(iter=iter\_times,aver\_acc=average(acc\_arr)))

result :  
0 's train accuracy : 0.9777777777777777

1 's train accuracy : 1.0

2 's train accuracy : 0.9777777777777777

3 's train accuracy : 1.0

4 's train accuracy : 0.9555555555555556

5 's train accuracy : 1.0

6 's train accuracy : 0.9777777777777777

7 's train accuracy : 0.9555555555555556

8 's train accuracy : 0.9111111111111111

9 's train accuracy : 0.9777777777777777

average accuracy of 10 times' GMM classification : 0.9733333333333334

1. Code:

from sklearn import datasets, model\_selection, metrics

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

from numpy import average

import pandas as pd

import numpy as np

from mlxtend.feature\_selection import SequentialFeatureSelector as SFS

from mlxtend.feature\_selection import ColumnSelector as CSR

cancer\_data = pd.read\_csv("./data/breast-cancer-wisconsin.data", names=[i for i in range(11)])

# print(len(cancer\_data))

cancer\_data = cancer\_data.drop(cancer\_data.columns[0], axis=1)

cancer\_data = cancer\_data.replace(to\_replace='?', value=pd.NA)

cancer\_data.dropna(inplace = True) ## remove the row that miss the attribute

cancer\_data = cancer\_data.reset\_index(drop=True)

# print(cancer\_data[[1,2,3,4,5,6,7,8,9]])

acc\_arr = []

iter\_time = 10

for i in range(iter\_time):

    # split the datasets

    data\_train, data\_test, target\_train, target\_test = model\_selection.train\_test\_split(

        cancer\_data[[i for i in range(1, 10)]], cancer\_data[10], test\_size=0.2

        )

    data\_train, data\_valid, target\_train, target\_valid = model\_selection.train\_test\_split(

        data\_train, target\_train, test\_size=0.25

        )

    model = GaussianNB()

    sfs = SFS(model, forward=False, cv=0, k\_features=3, scoring='accuracy', verbose=False, n\_jobs=-1)

    sfs.fit(data\_valid, target\_valid) # 20%

    print(f"Best score achieved: {sfs.k\_score\_}, \nFeature's names: {sfs.k\_feature\_names\_}")

    model.fit(CSR(cols=sfs.k\_feature\_names\_).transform(data\_train),target\_train) # 60%

    prediction = model.predict(CSR(cols=sfs.k\_feature\_names\_).transform(data\_test)) # 20%

    accur = metrics.accuracy\_score(target\_test, prediction)

    acc\_arr.append(accur)

print(f"\nthe average accuracy of 10 times trial =",average(acc\_arr))

result:

Best score achieved: 0.9708029197080292,

Feature's names: (1, 3, 6)

Best score achieved: 0.9854014598540146,

Feature's names: (1, 2, 6)

Best score achieved: 0.9708029197080292,

Feature's names: (1, 2, 7)

Best score achieved: 0.9562043795620438,

Feature's names: (1, 2, 6)

Best score achieved: 0.948905109489051,

Feature's names: (1, 3, 8)

Best score achieved: 0.9781021897810219,

Feature's names: (2, 3, 4)

Best score achieved: 0.9708029197080292,

Feature's names: (1, 3, 6)

Best score achieved: 0.9635036496350365,

Feature's names: (1, 4, 6)

Best score achieved: 0.9708029197080292,

Feature's names: (1, 5, 6)

Best score achieved: 0.9781021897810219,

Feature's names: (2, 3, 6)

the average accuracy of 10 times trial = 0.9547445255474452