From IPython.core.pylabtools import figsize

From sklearn.datasets import load\_iris

Import matplotlib.pyplot as plt

Import pandas as pd

From sklearn.cluster import KMeans

I = load\_iris()

X, y = i.data[:,:2], i.target

Plt.scatter(x[:,0],x[:,1],c=y,cmap=’prism’)

Km = KMeans(n\_clusters=3, init=’k-means++’).fit(x)

C = km.cluster\_centers\_

Print©

N = km.labels\_

Print(n)

Print(y)

Fig, axes = plt.subplots(1,2,figsize=(16,8))

Axes[0].scatter(x[:,0],x[:,1],c=y,cmap=’prism’,edgecolor=’k’,s=75)

Axes[1].scatter(x[:,0],x[:,1],c=n,cmap=’jet’,edgecolor=’k’,s=75)

Axes[0].tick\_params(colors=’k’,labelsize=15)

Axes[1].tick\_params(colors=’k’,labelsize=15)

2import matplotlib.pyplot as plt

From sklearn import datasets

From sklearn.linear\_model import LinearRegression

From sklearn.metrics import mean\_squared\_error,r2\_score

Import numpy as np

I = datasets.load\_diabetes()

X,y = datasets.load\_diabetes(return\_X\_y=True)

X = x[:,np.newaxis,2]

Xtr = x[:-20]

Xte = x[-20:]

Ytr = y[:-20]

Yte = y[-20:]

R = LinearRegression().fit(xtr,ytr)

Y\_pred = r.predict(xte)

Print(r.coef\_)

Print(mean\_squared\_error(yte,y\_pred))

Print(r2\_score(yte,y\_pred))

Plt.scatter(xte,yte,color=’black’)

Plt.plot(xte,y\_pred,color=’blue’,linewidth=5)

Plt.show()

3

From sklearn.datasets import load\_iris

From sklearn.metrics import accuracy\_score

From sklearn.model\_selection import train\_test\_split

From sklearn.tree import DecisionTreeClassifier

I = load\_iris()

X, y = i.data,i.target

Xtr, xte, ytr, yte = train\_test\_split(x,y,train\_size=0.7)

C = DecisionTreeClassifier().fit(xtr, ytr)

Ypred = c.predict(xte)

Print(accuracy\_score(ytr,c.predict(xtr)))

Print(accuracy\_score(yte,ypred))

E = DecisionTreeClassifier(criterion=’entropy’).fit(xtr,ytr)

Ypred\_e = e.predict(xte)

Print(accuracy\_score(ytr,e.predict(xtr)))

Print(accuracy\_score(yte,ypred\_e))

E2 = DecisionTreeClassifier(criterion=’entropy’, min\_samples\_split=50).fit(xtr, ytr)

Yp = e2.predict(xte)

Print(accuracy\_score(ytr,e2.predict(xtr)))

Print(accuracy\_score(yte,yp))

4

From sklearn import preprocessing

From sklearn.model\_selection import train\_test\_split

From sklearn.naive\_bayes import BernoulliNB, GaussianNB

From sklearn.metrics import confusion\_matrix, accuracy\_score

From sklearn.datasets import load\_iris

I = load\_iris()

X,y = i.data[:,:],i.target

Xtr, xte, ytr, yte = train\_test\_split(x, y, train\_size = 0.7)

S = preprocessing.StandardScaler().fit(xtr)

Xtr = s.transform(xtr)

Xte = s.transform(xte)

# sc = []

B = BernoulliNB()

b.fit(xtr, ytr)

y\_pred = b.predict(xte)

print(confusion\_matrix(yte,y\_pred))

g = GaussianNB()

g.fit(xtr, ytr)

y\_pred = g.predict(xte)

print(confusion\_matrix(yte,y\_pred))

5

From sklearn import preprocessing, neighbors

From sklearn.datasets import load\_digits

From sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

From sklearn.model\_selection import train\_test\_split

I = load\_digits()

X, y = i.data[:,:], i.target

X\_tr, x\_te, y\_tr, y\_te = train\_test\_split(x, y, train\_size=0.7)

Scaler = preprocessing.StandardScaler().fit(x\_tr)

X\_tr = scaler.transform(x\_tr)

X\_te = scaler.transform(x\_te)

For k in (1,15):

Knn = neighbors.KNeighborsClassifier(n\_neighbors=k)

Knn.fit(x\_tr,y\_tr)

Y\_pred = knn.predict(x\_te)

Print(accuracy\_score(y\_te,y\_pred))

Print(confusion\_matrix(y\_te,y\_pred))

Print(classification\_report(y\_te,y\_pred))