|  |  |
| --- | --- |
| In[1] | import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  from sklearn import datasets  from sklearn.model\_selection import train\_test\_split  import sklearn.metrics as metrics |

|  |  |
| --- | --- |
| In[2] | def load\_data(infile):  dt1 = pd.read\_excel(infile)  x\_can=dt1.iloc[:,1:8]  y\_can= dt1.CAN  x\_train, x\_test, y\_train, y\_test=train\_test\_split(x\_can,y\_can,test\_size=0.2,random\_state=0)  return x\_train, y\_train, x\_test,y\_test |

|  |  |
| --- | --- |
| In[3] | infile="./data/can\_data1.xlsx"  dts = load\_data(infile) |

|  |  |
| --- | --- |
| In[4] | print(dts[0].shape, dts[1].shape, dts[2].shape, dts[3].shape) |
| Out[4] | (800, 7) (800,) (200, 7) (200,) |

|  |  |
| --- | --- |
| In[5] | dts[0].head() |
| Out[5] | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  | Age | HR | SBP | FPG | PBG | HTN | DM | | 687 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | | 500 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | | 332 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | | 979 | 2 | 0 | 1 | 0 | 0 | 1 | 0 | | 817 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |

|  |  |
| --- | --- |
| In[6] | dts[1].head() |
| Out[6] | 687 0  500 0  332 1  979 1  817 0  Name: CAN, dtype: int64 |

|  |  |
| --- | --- |
| In[7] | from sklearn.tree import DecisionTreeClassifier  def test\_decisontreecls(dts):  clf=DecisionTreeClassifier()  clf.fit(dts[0],dts[1])  y\_predict=clf.predict(dts[2])  print("acc:",metrics.accuracy\_score(y\_predict,dts[3])) |

|  |  |
| --- | --- |
| In[8] | test\_decisontreecls(dts) |
| Out[8] | acc: 0.865 |

|  |  |
| --- | --- |
| In[9] | from sklearn import svm  def test\_svc(dts):  cls=svm.SVC()  cls.fit(dts[0],dts[1])  y\_predict=cls.predict(dts[2])  print("acc:",metrics.accuracy\_score(y\_predict,dts[3])) |

|  |  |
| --- | --- |
| In[10] | test\_svc(dts) |
| Out[10] | acc: 0.87 |

|  |  |
| --- | --- |
| In[11] | from sklearn.linear\_model import LogisticRegression  def test\_logsticreg(dts):  cls=LogisticRegression()  cls.fit(dts[0],dts[1])  y\_predict=cls.predict(dts[2])  print("acc:",metrics.accuracy\_score(y\_predict,dts[3])) |

|  |  |
| --- | --- |
| In[12] | test\_logsticreg(dts) |
| Out[12] | acc: 0.87 |

|  |  |
| --- | --- |
| In[13] | from sklearn.ensemble import RandomForestClassifier  def test\_rf(dts):  cls = RandomForestClassifier(n\_estimators=1000)  cls.fit(dts[0],dts[1])  y\_predict=cls.predict(dts[2])  print("acc:",metrics.accuracy\_score(y\_predict,dts[3])) |

|  |  |
| --- | --- |
| In[14] | test\_rf(dts) |
| Out[14] | acc: 0.885 |

|  |  |
| --- | --- |
| In[15] | from sklearn.neighbors import KNeighborsClassifier  def test\_knn(dts):  cls = KNeighborsClassifier()  cls.fit(dts[0],dts[1])  y\_predict=cls.predict(dts[2])  print("acc:",metrics.accuracy\_score(y\_predict,dts[3])) |

|  |  |
| --- | --- |
| In[16] | test\_knn(dts) |
| Out[16] | acc: 0.865 |

|  |  |
| --- | --- |
| In[17] | from sklearn.naive\_bayes import GaussianNB  def test\_gnb(dts):  cls = GaussianNB()  cls.fit(dts[0],dts[1])  y\_predict=cls.predict(dts[2])  print("acc:",metrics.accuracy\_score(y\_predict,dts[3])) |

|  |  |
| --- | --- |
| In[18] | test\_gnb(dts) |
| Out[18] | acc: 0.85 |

|  |  |
| --- | --- |
| In[19] | from xgboost import XGBClassifier  def test\_xgboost(dts):  cls = XGBClassifier()  cls.fit(dts[0],dts[1])  y\_predict=cls.predict(dts[2])  print("acc:",metrics.accuracy\_score(y\_predict,dts[3])) |

|  |  |
| --- | --- |
| In[20] | test\_xgboost(dts) |
| Out[20] | acc: 0.87 |

|  |  |
| --- | --- |
| In[21] | from sklearn.ensemble import GradientBoostingClassifier  def test\_gbdt(dts):  cls = GradientBoostingClassifier(n\_estimators=256)  cls.fit(dts[0],dts[1])  y\_predict=cls.predict(dts[2])  print("acc:",metrics.accuracy\_score(y\_predict,dts[3])) |

|  |  |
| --- | --- |
| In[22] | test\_gbdt(dts) |
| Out[22] | acc: 0.855 |

|  |  |
| --- | --- |
| In[23] | from sklearn.ensemble import AdaBoostClassifier  def test\_abc(dts):  cls = AdaBoostClassifier()  cls.fit(dts[0],dts[1])  y\_predict=cls.predict(dts[2])  print("acc:",metrics.accuracy\_score(y\_predict,dts[3])) |

|  |  |
| --- | --- |
| In[24] | test\_abc(dts) |
| Out[24] | acc: 0.86 |

|  |  |
| --- | --- |
| In[25] | from sklearn.neural\_network import MLPClassifier  def test\_ann(dts):  cls=MLPClassifier(alpha=1e-5,hidden\_layer\_sizes=(16,8),random\_state=1)  cls.fit(dts[0],dts[1])  y\_predict=cls.predict(dts[2])  print("acc:",metrics.accuracy\_score(y\_predict,dts[3])) |

|  |  |
| --- | --- |
| In[26] | test\_ann(dts) |
| Out[26] | acc: 0.87 |