# 《企业实训》机器学习作业提交

大学：内蒙古大学 学院：计算机学院 专业： 软件工程

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 姓名 | | 李纪元 | | 学号 | 0191121383 | 班级 | 软工二班 |
| 实验任务名称 | | **机器学习实验一：使用特征分析和可视化方法理解机器学习的iris数据集**。 | | | | | |
| 实验内容 | | 通过对数据集的各个特征单独及分组进行分析和编程，将分析结果进行可视化来更好地理解机器学习的iris数据集，更好地理解特征本身的分布特性、特征与目标的关系等。 | | | | | |
| 实验代码和结果 | | import matplotlib.pyplot as plt  x\_axis = [i for i in range(50)]  plt.figure(figsize = (16,8))  plt.subplot(2,5,1)  plt.scatter(x\_axis, X[:50, 0])  #特征1  plt.subplot(2,5,2)  x\_axis = [i for i in range(50)]  plt.scatter(x\_axis, X[:50, 0])  plt.scatter(x\_axis,X[50:100,0])  plt.scatter(x\_axis,X[100:,0])  #特征2  plt.subplot(2,5,3)  x\_axis = [i for i in range(50)]  plt.scatter(x\_axis, X[:50, 1])  plt.scatter(x\_axis,X[50:100,1])  plt.scatter(x\_axis,X[100:,1])  #特征1折线  plt.subplot(2,5,4)  x\_axis = [i for i in range(50)]  plt.plot(x\_axis, X[:50, 0])  plt.plot(x\_axis,X[50:100,0])  plt.plot(x\_axis,X[100:,0])  #特征2折线  plt.subplot(2,5,5)  x\_axis = [i for i in range(50)]  plt.plot(x\_axis, X[:50, 1])  plt.plot(x\_axis,X[50:100,1])  plt.plot(x\_axis,X[100:,1])  #特征3折线  plt.subplot(2,5,6)  x\_axis = [i for i in range(50)]  plt.plot(x\_axis, X[:50, 2])  plt.plot(x\_axis,X[50:100,2])  plt.plot(x\_axis,X[100:,2])  #特征4折线  plt.subplot(2,5,7)  x\_axis = [i for i in range(50)]  plt.plot(x\_axis, X[:50, 3])  plt.plot(x\_axis,X[50:100,3])  plt.plot(x\_axis,X[100:,3])  #特征3  plt.subplot(2,5,8)  x\_axis = [i for i in range(50)]  plt.scatter(x\_axis, X[:50, 2])  plt.scatter(x\_axis,X[50:100,2])  plt.scatter(x\_axis,X[100:,2])  #特征4  plt.subplot(2,5,9)  x\_axis = [i for i in range(50)]  plt.scatter(x\_axis, X[:50, 3])  plt.scatter(x\_axis,X[50:100,3])  plt.scatter(x\_axis,X[100:,3])  #特征2，4  plt.subplot(2,5,10)  plt.scatter(X[:50,1], X[:50, 3])  plt.scatter(X[50:100,1],X[50:100,3])  plt.scatter(X[100:,1],X[100:,3])  plt.xlabel(iris.feature\_names[0])  plt.ylabel(iris.feature\_names[1])  plt.show()    #总体密度图  sns.kdeplot(X[:,0],color = 'purple',label='sepal length(cm)')  sns.kdeplot(X[:,1],color = 'pink',label='sepal width (cm)')  sns.kdeplot(X[:,2],label='petal length (cm))')  sns.kdeplot(X[:,3],label='petal width (cm)')  plt.legend()  plt.title('Total density')  plt.show()    #双变量图  import numpy as np  import matplotlib  import pandas as pd  plot\_data = pd.DataFrame(X,columns=['sepal length(cm)', 'sepal width(cm)', 'petal length(cm)', 'petal width(cm)'])  plot\_data = plot\_data.replace({np.inf: np.nan,-np.inf:np.nan})  plot\_data.dropna  def corr\_func(x,y,\*\*kwargs):  r = np.corrcoef(x,y)[0][1]  ax = plt.gca()  ax.annotate("r = {:.2f}".format(r),xy = (.2,.8),xycoords = ax.transAxes,size = 20)    grid = sns.PairGrid(data = plot\_data,size = 3)  grid.map\_upper(plt.scatter,color = 'red',alpha = 0.6)  grid.map\_diag(plt.hist,color = 'red',edgecolor = 'black')  grid.map\_lower(corr\_func)  grid.map\_lower(sns.kdeplot,cmap = plt.cm.Reds)  plt.suptitle('Pairs Plot of Iris',size = 36, y = 1.2) | | | | | |
| 实验任务名称 | | **机器学习实验二：使用线性回归和逻辑回归算法对iris数据集进行分类。** | | | | | |
| 实验内容 | | 通过对线性回归和逻辑回归算法编程实现iris数据集的分类，及使用单特征和多特征进行训练和评估分类结果，帮助学生巩固和掌握线性回归算法以及在其上增加了非线性函数的逻辑回归算法。 | | | | | |
| 实验代码和结果 | | from sklearn.datasets import load\_iris  iris = load\_iris()  print(iris.keys)  print(iris.feature\_names)  X = iris.data  y = iris.target  print("y = ",y.shape,y)  print("x = ",X.shape,X)    #线性回归  from sklearn import linear\_model  linear = linear\_model.LinearRegression()  linear.fit(X[::,1:2:], y)  print("train score: ", linear.score(X[::,1:2:],y))  print(linear.coef\_)  print(linear.intercept\_)  print("predict: ", linear.predict([[2],[7]]))    import numpy as np  import matplotlib.pyplot as plt  linear.fit(X[::, 0:1:], y)  print(linear.coef\_)  print(linear.intercept\_)  print("predict:", linear.predict([[7],[7.5]]))  plt.scatter(X[::, 0:1:], y)  plt.scatter(X[::, 0:1:], np.dot(X[::, 0:1:], linear.coef\_) + linear.intercept\_)  plt.plot(X[::, 0:1:],np.dot(X[::,0:1:], linear.coef\_) + linear.intercept\_)  plt.show()    #逻辑回归  from sklearn.linear\_model import LogisticRegression  lr = linear\_model.LogisticRegression()  lr.fit(X, y)  print("train score: ", lr.score(X,y))  print(lr.coef\_)  print(lr.intercept\_)  print("predict: ", lr.predict([[7, 5, 2, 0.5],[7.5, 4, 7, 2]]))    #特征全排列线性回归  test1 = [7, 5, 2, 0.5]  test2 = [7.5, 4, 7, 2]  def dfs(cur, arr, idx, c1, c2):  if cur == 4:  if(idx):  print("result of :",idx)  ln = linear\_model.LinearRegression()  ln.fit(arr, y)  print("train score: ", ln.score(arr,y))  print(ln.coef\_)  print(ln.intercept\_)  #print("predict: ", linear.predict([c1,c2]))  return  arr = np.insert(arr, arr.shape[1], X[::, cur:cur + 1:].T, axis = 1)  idx.append(cur)  c1.append(test1[cur])  c2.append(test2[cur])  dfs(cur + 1, arr, idx, c1, c2)  arr = np.delete(arr, arr.shape[1] - 1, axis = 1)  idx.pop()  c1.pop()  c2.pop()  dfs(cur + 1, arr, idx, c1, c2)  return  for i in range(0, 4):  arr = X[::, i: i + 1:]  idx = [i]  c1 = [test1[i]]  c2 = [test2[i]]  dfs(i + 1, arr, idx, c1, c2) | | | | | |
| 实验任务名称 | | **机器学习实验三：使用SVM支持向量机算法对iris数据集进行分类。** | | | | | |
| 实验内容 | | 通过使用SVM支持向量机算法编程实现对iris数据集的分类，并练习画图来可视化理解SVM算法和支持向量机的核函数，帮助学生巩固和掌握SVM算法并重点理解RBF径向基核函数的使用。 | | | | | |
| 实验代码和结果 | | from sklearn.datasets import load\_iris  iris = load\_iris()  print(iris.keys)  print(iris.feature\_names)  X = iris.data  y = iris.target  print("y = ",y.shape,y)  print("x = ",X.shape,X)    from sklearn import svm  svm1 = svm.SVC()  svm1.fit(X,y)  print("training score:",svm1.score(X,y))  print("predict:",svm1.predict([[7,5,2,0.5],[7.5,4,7,2]]))    #只使用特征1  svm2 = svm.SVC()  svm2.fit(X[:,:1],y)  print("training score:",svm2.score(X[:,:1],y))  print("predict:",svm2.predict([[7],[7.5]]))    #只使用特征1、2  from sklearn import svm  import numpy as np  import matplotlib.pyplot as plt  svm3 = svm.SVC(kernel='linear', C=1, gamma=1/2) # 1/f  svm3.fit(X[:, :2],y)  print("training score: ",svm3.score(X[:, :2], y))  print("predict: ",svm3.predict([[7, 5],[7.5, 4]]))  x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1  y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  h = (x\_max / x\_min)/100  xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))  plt.subplot(1, 1, 1)  Z = svm3.predict(np.c\_[xx.ravel(), yy.ravel()])  Z = Z.reshape(xx.shape)  plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)  plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)  plt.xlabel('Sepal length')  plt.ylabel('Sepal width')  plt.xlim(xx.min(), xx.max())  plt.title('SVC with linear kernel')  plt.show()    #支持向量机3：只使用特征1、2  def calc\_xy(vGamma):  from sklearn import svm  svm = svm.SVC(kernel='rbf', C=1, gamma=vGamma)  svm.fit(X[:, :2],y)  print("training score: ",svm.score(X[:, :2], y))  print("predict: ",svm.predict([[7, 5],[7.5, 4]]))  x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1  y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1  h = (x\_max / x\_min)/100  xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))  Z = svm.predict(np.c\_[xx.ravel(), yy.ravel()])  Z = Z.reshape(xx.shape)  plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)  plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)  plt.xlabel('Sepal length')  plt.ylabel('Sepal width')  plt.xlim(xx.min(), xx.max())  plt.title('SVC with linear kernel')  return xx,yy,Z  plt.figure(figsize = (12,6))  plt.subplot(1,3,1)  calc\_xy(1/2)  plt.subplot(1,3,2)  calc\_xy(10)  plt.subplot(1,3,3)  calc\_xy(100)  plt.show() | | | | | |
| 实验任务名称 | | **机器学习实验四：使用决策树算法对iris数据集进行分类。** | | | | | |
| 实验内容 | | 通过使用决策树算法编程实现iris数据集的分类，并练习手动计算决策树算法的信息熵和基尼系数，帮助学生巩固和掌握决策树的ID3算法、C4.5算法并重点练习和理解CART决策树算法。 | | | | | |
| 实验代码和结果 | | #决策树  from sklearn import tree  clf = tree.DecisionTreeClassifier(criterion='entropy')  lenses = clf.fit(X,y)  print("training score: ", clf.score(X,y))  print("predict: ", clf.predict([[7,5,2,0.5],[7.5,4,7,2]]))  from six import StringIO  dot\_data = StringIO()  tree.export\_graphviz(clf, out\_file = dot\_data,  feature\_names = iris.feature\_names,  class\_names = iris.target\_names,  filled = True, rounded = True,  special\_characters = True)  import pydotplus  graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())  from IPython.display import display, Image  display(Image(graph.create\_png()))    ID3算法：    根节点信息熵：  E(S) = -[(9/14)log2(9/14) + (5/14)log2(5/14)] = 0.94  对outlook：    outlook加权平均熵：  E(S,outlook) = (5/14)\*E(3,2) + (4/14)\*E(4,0) + (5/14)\*E(2,3)  = (5/14)(-(3/5)log2(3/5)-(2/5)log2(2/5) + (4/14)(0) + (5/14)(-(2/5)log2(2/5)-(3/5)log2(3/5)) = 0.69  信息增益：  IG(S,outlook) = 0.94 – 0.69 = 0.25  对temperature:    Temperature加权平均熵：  E(temperature) = (4/14)E(2,2) + (6/14)E(4,2) + (4/14)E(3,1)  = (4/14) \* (-(2/4)\*log2(2/4)-(2/4)\*log2(2/4)) + (6/14)\*(-(4/6)\*log2(4/6)-(2/6)\*log2(2/6)) + (4/14)\*(-(3/4)\*log2(3/4)-(1/4)\*log2(1/4))  = 0.91  信息增益：  E(S,temperature) = 0.94 – 0.91 = 0.03  对humidity:    Humdity加权平均熵：  E(humidity) = (7/14)E(3,4) + (7/14)E(6,1)  = (7/14) \* (-(3/7)\*log2(3/7)-(4/7)\*log2(4/7)) + (7/14)\*(-(6/7) \* log2(6/7) – (1/7) \* log2(1/7))  = 0.78  信息增益：  IG(S,humidity) = 0.94 – 0.78 = 0.16  对wind:    Wind加权平均熵：  E(wind) = (6/14)E(3,3) + (8/14)E(6,2)  = (6/14) \* (-(3/6)\*log2(3/6) –(3/6)\*log2(3/6)) + (8/14) \* (-(6/8)\*log2(6/8) – (2/8)\*log2(2/8))  = 0.89  信息增益：  IG(S,wind) = 0.94 – 0.89 = 0.05  选择outlook作为下一个节点，由于outlook为overcast时均为yes，故该分支可直接确定。  对于sunny：  E(sunny) = -(2/5)log2(2/5)-(3/5)log2(3/5) = 0.97    Temperature的加权平均熵：  E(sunny,temperature) = (2/5)E(0,2) + (2/5)E(1,1) + (1/5)E(1,0)  = (2/5)\*(-(1/2)\*log2(1/2)-(1/2)\*log2(1/2))  = 0.40  信息增益：  IG(sunny,teperature) = 0.97 – 0.40 = 0.57  对humidity:    E(sunny,humidity) = (3/5)E(0,3) + (2/5)E(2,0)  = 0  IG(sunny,humidity) = 0.97  对wind:    E(sunny,wind) = (2/5)E(1,1) + (3/5)E(1,2)  = (2/5)(-(1/2)log2(1/2)-(1/2)log2(1/2)) +(3/5)(-(1/3)log2(1/3)-(2/3)log2(2/3)) = 0.95  IG(sunny,wind) = 0.97 – 0.95 = 0.02  我们选择humidity作为sunny的分支节点，由于sunny条件下humidity为high时均为no，为normal时均为yes，故可直接确定。  对rain:  E(rain) - -(2/5)log2(2/5) – (3/5)log2(3/5) = 0.97    E(rain,temperature) = 0 + (3/5)E(2,1) + (2/5)E(1,1)  = (2/5)(-(1/2)log2(1/2)-(1/2)log2(1/2)) + (3/5)(-(2/3)log2(2/3)-(1/3)log2(1/3))  =0.95  IG(rain,temperature) = 0.97 – 0.95 = 0.02    E(rain,humidity) = (2/5)E(1,1) + (3/5)E(2,1)  = (2/5)(-(1/2)log2(1/2)-(1/2)log2(1/2))+(3/5)(-(1/3)log2(1/3)-(2/3)log2(2/3))  = 0.95  IG(rain,humidity) = 0.97 – 0.95 = 0.02    E(rain,wind) = 0  IG(rain,wind) = 0.97  选择wind作为rain分支节点，由于rain条件下wind为weak均为yes，为strong均为no，故可直接确定。    CART 算法：  Gini(S) = 1 – [(9/14)^2 + (5/14)^2] = 0.46    Gini(S,outlook) = (5/14)gini(3,2) + (4/14)gini(4,0)+(5/14)gini(2,3)  =(5/14)(1-(3/5)^2-(2/5)^2) + 0 + (5/14)(1 – (2/5)^2 – (3/5)^2)  =0.34  Ginigain(S,outlook) = 0.46 – 0.34 = 0.12    Gini(S,temperature) = (4/14)gini(2,2) + (6/14)gini(4,2) + (4/14)gini(3,1)  =(4/14)(1-(2/4)^2-(2/4)^2) + (6/14)(1-(4/6)^2-(2/6)^2)+(4/14)(1-(3/4)^2-(1/4)^2)  =0.44  Ginigain(S,temperature) = 0.46 – 0.44 = 0.02    Gini(S,humidity) = (7/14)gini(3,4) + (7/14)gini(6,1)  = (7/14)(1-(3/7)^2 – (4/7)^2) + (7/14)(1 – (6/7)^2 – (1/7)^2)  = 0.37  Ginigain(S,humidity) = 0.46 – 0.37 = 0.09    Gini(S,wind) = (6/14)gini(3,3) + (8/14)gini(6,2)  (6/14)(1-(3/6)^2-(3/6)^2) + (8/14)(1-(6/8)^2-(2/8)^2)  = 0.43  Ginigain(S,wind) = 0.46 – 0.43 = 0.03  选择outlook作为根节点，由于outlook为overcast时均为yes，故可直接确定。  Gini(sunny) = 1 – (3/5)^2 – (2/5)^2 = 0.48    Gini(sunny,temperature)  = (2/5)gini(0,2)+(2/5)gini(1,1)+(1/5)gini(1,0)  =0+(2/5)(1-(1/2)^2-(1/2)^2)+0  =0.2  Ginigain(sunny,temperature)=0.48-0.2=0.28    Gini(sunny,humidity)=(3/5)gini(0,3)+(2/5)gini(2,0)  =0  Ginigain(sunny,humidity)=0.48    Gini(sunny,wind)=(2/5)gini(1,1)+(3/5)\*gini(1,2)         ==(2/5)(1-(1/2)^2-(1/2)^2)+(3/5)(1-(1/3)^2-(2/3)^2)  =0.47 Ginigain(sunny,wind)=0.48-0.47=0.01  选择humidity为sunny分支节点，由于sunny条件下high均为no，normal均为yes，故可直接确定。  Gini(rain)=1-(3/5)^2-(2/5)^2=0.48    Gini(rain,temperature)= 0+(3/5)gini(2,1)+(2/5)\*gini(1,1)  =(3/5)(1-(2/3)^2-(1/3)^2)+(2/5)(1-(1/2)^2-(1/2)^2)  =0.47  Ginigain(rain,temperature)=0.48-0.47=0.01    Gini(rain,humidity)=(3/5)gini(2,1)+(2/5)gini(1,1)  =(3/5)(1-(2/3)^2-(1/3)^2)+(2/5)(1-(1/2)^2-(1/2)^2)  =0.47 Ginigain(rain,humidity)=0.48–0.47=0.01    Gini(rain,wind)=(2/5)gini(0,2)+(3/5)gini(3,0)  = 0  Ginigain(rain,wind)=0.48  选择Wind作为rain分支节点，由于rain条件下strong均为no，weak均为yes，故可直接确定。  决策树的结构同上，此处略。 | | | | | |
| 实验任务名称 | | **机器学习实验五：使用朴素贝叶斯算法对iris数据集进行分类。** | | | | | |
| 实验内容 | | 使用朴素贝叶斯算法编程实现iris数据集的分类, 并练习画图来可视化理解朴素贝叶斯算法，以及依据算法练习手动计算各项先验概率值，帮助学生巩固和掌握朴素贝叶斯算法和先验概率。 | | | | | |
| 实验代码和结果 | | #朴素贝叶斯，高斯分布  from sklearn import naive\_bayes  bayes = naive\_bayes.GaussianNB()  bayes.fit(X,y)  print("training score: ",bayes.score(X,y))  print("predict: ",bayes.predict([[7,5,2,0.5],[7.5,4,7,2]]))    #朴素贝叶斯，多项式分布  from sklearn import naive\_bayes  bayes = naive\_bayes.MultinomialNB()  bayes.fit(X,y)  print("training score: ",bayes.score(X,y))  print("predict: ",bayes.predict([[7,5,2,0.5],[7.5,4,7,2]]))    # 朴素贝叶斯算法  from sklearn.naive\_bayes import GaussianNB  import numpy as np  import pandas as pd  from pandas import Series,DataFrame  import matplotlib.pyplot as plt  from sklearn.datasets import load\_iris  from matplotlib.colors import ListedColormap  %matplotlib inline  #导入函数  muNB = GaussianNB()  #读取数据  iris = load\_iris()  #取出数据中的data  data = iris.data  #取出数据中的target  target = iris.target  #取data中所有行前两列为训练数据  samples = data[:,:2]  #训练数据  muNB.fit(samples,target)  #取出训练数据中第一列中的最大与最小值  xmin,xmax = samples[:,0].min(),samples[:,0].max()  #取出训练数据中第二列中的最大与最小值  ymin,ymax = samples[:,1].min(),samples[:,1].max()  #在最大与最小值的区间分成300个数据  x = np.linspace(xmin,xmax,300)  y = np.linspace(ymin,ymax,300)  #然后使这些数据组成一个平面  xx,yy = np.meshgrid(x,y)  #生成90000个坐标点  X\_test = np.c\_[xx.ravel(),yy.ravel()]  #预测训练数据  y\_ = muNB.predict(X\_test)  #导入三种不同的颜色  colormap = ListedColormap(['#00aaff','#aa00ff','#ffaa00'])  #生成三个不同颜色的模块，第一列为x轴坐标，第二列为y轴坐标，预测之后，不同的点分成不同的三类  plt.scatter(X\_test[:,0],X\_test[:,1],c=y\_)  plt.scatter(samples[:,0],samples[:,1],c=target,cmap=colormap)  plt.show()    from sklearn.naive\_bayes import GaussianNB  import numpy as np  import pandas as pd  from pandas import Series,DataFrame  import matplotlib.pyplot as plt  from sklearn.datasets import load\_iris  from matplotlib.colors import ListedColormap  %matplotlib inline  #导入函数  muNB = GaussianNB()  #读取数据  iris = load\_iris()  #取出数据中的data  data = iris.data  #取出数据中的target  target = iris.target  #取data中所有行后两列为训练数据  samples = data[:,-2:]  #训练数据  muNB.fit(samples,target)  #取出训练数据中第一列中的最大与最小值  xmin,xmax = samples[:,0].min(),samples[:,0].max()  #取出训练数据中第二列中的最大与最小值  ymin,ymax = samples[:,1].min(),samples[:,1].max()  #在最大与最小值的区间分成300个数据  x = np.linspace(xmin,xmax,300)  y = np.linspace(ymin,ymax,300)  #然后使这些数据组成一个平面  xx,yy = np.meshgrid(x,y)  #生成90000个坐标点  X\_test = np.c\_[xx.ravel(),yy.ravel()]  #预测训练数据  y\_ = muNB.predict(X\_test)  #导入三种不同的颜色  colormap = ListedColormap(['#00aaff','#aa00ff','#ffaa00'])  #生成三个不同颜色的模块，第一列为x轴坐标，第二列为y轴坐标，预测之后，不同的点分成不同的三类  plt.scatter(X\_test[:,0],X\_test[:,1],c=y\_)  plt.scatter(samples[:,0],samples[:,1],c=target,cmap=colormap)  plt.show() | | | | | |
| 实验任务名称 | | 机器学习实验六：使用KNN聚类算法对iris数据集进行分类。 | | | | | |
| 实验内容 | | 通过使用KNN聚类算法编程实现iris数据集的分类，并练习画图来可视化理解KNN聚类算法，帮助学生巩固和掌握KNN聚类算法、以及相关的机器学习的准确度、召回率和精度等评估值的计算。 | | | | | |
| 实验代码和结果 | | # KNN 算法  import numpy as np  import matplotlib.pylab as pyb  %matplotlib inline  from sklearn.neighbors import KNeighborsClassifier  from sklearn import datasets  iris = load\_iris()  #取出数据中的data  X = iris.data  #取出数据中的target  y = iris.target  # pyb.scatter(X[:,0],X[:,1],c = y)  from sklearn import neighbors  KNN = neighbors.KNeighborsClassifier(n\_neighbors = 5)  X = X[:, :2]  KNN.fit(X,y)  print("training score: ",KNN.score(X,y))  x1 = np.linspace(4,8,100)  y1 = np.linspace(2,4.5,80)  X1,Y1 = np.meshgrid(x1,y1)  # 平铺，一维化，reshape  X\_test = np.c\_[X1.ravel(),Y1.ravel()]  print('X\_test.shape = ', X\_test.shape)  # 8000个样本,每个样本2个属性[鸢尾花萼的长度和宽度]。  # 使用算法进行预测，可视化  y\_ = KNN.predict(X\_test)  # 导入颜色包  from matplotlib.colors import ListedColormap  # 画图  lc = ListedColormap(['#FFAAAA','#AAFFAA','#AAAAFF'])  lc2 = ListedColormap(['#FF0000','#00FF00','#0000FF'])  pyb.scatter(X\_test[:,0],X\_test[:,1],c = y\_,cmap = lc)  pyb.scatter(X[:,0],X[:,1],c = y,cmap = lc2)    import numpy as np  import matplotlib.pylab as pyb  %matplotlib inline  from sklearn.neighbors import KNeighborsClassifier  from sklearn import datasets  iris = load\_iris()  #取出数据中的data  X = iris.data  #取出数据中的target  y = iris.target  from sklearn import neighbors  KNN = neighbors.KNeighborsClassifier(n\_neighbors = 5)  X = X[:, 2:4]  KNN.fit(X,y)  print("training score: ",KNN.score(X,y))  x1 = np.linspace(4,8,100)  y1 = np.linspace(2,4.5,80)  X1,Y1 = np.meshgrid(x1,y1)  # 平铺，一维化，reshape  X\_test = np.c\_[X1.ravel(),Y1.ravel()]  print('X\_test.shape = ', X\_test.shape)  # 8000个样本,每个样本2个属性[鸢尾花萼的长度和宽度]。  # 使用算法进行预测，可视化  y\_ = KNN.predict(X\_test)  # 导入颜色包  from matplotlib.colors import ListedColormap  # 画图  lc = ListedColormap(['#FFAAAA','#AAFFAA','#AAAAFF'])  lc2 = ListedColormap(['#FF0000','#00FF00','#0000FF'])  pyb.scatter(X\_test[:,0],X\_test[:,1],c = y\_,cmap = lc)  pyb.scatter(X[:,0],X[:,1],c = y,cmap = lc2)    from sklearn.metrics import confusion\_matrix, accuracy\_score, recall\_score, precision\_score, f1\_score  from sklearn.model\_selection import train\_test\_split  from sklearn import preprocessing  # 数据预处理：对变色鸢尾花和维吉尼亚鸢尾花按列归一化  iris\_X=preprocessing.scale(X[50:])  # 对变色鸢尾花和维吉尼亚鸢尾花切分数据集：测试集占 30%  iris\_X\_train,iris\_X\_test,iris\_y\_train,iris\_y\_test=train\_test\_split(iris\_X,y[50:],test\_size=0.3,random\_state=0)  # 导入KNN 分类模型  from sklearn import neighbors  # 选定邻居数量，创建KNN模型对象  KNN = neighbors.KNeighborsClassifier(n\_neighbors = 5)  # 模型训练  KNN.fit(iris\_X\_train,iris\_y\_train)  # 模型预测  y\_pred = KNN.predict(iris\_X\_test)  # 显示混淆矩阵  print(confusion\_matrix(iris\_y\_test,y\_pred))  # 计算准确率  print("准确率：%.3f"%accuracy\_score(iris\_y\_test,y\_pred))  # 计算召回率  print("召回率：%.3f"%recall\_score(iris\_y\_test,y\_pred))  # 计算精度  print("精度：%.3f"%precision\_score(iris\_y\_test,y\_pred))  # 计算f1值  print("f1：%.3f"%f1\_score(iris\_y\_test,y\_pred)) | | | | | |