**Python语言与数据分析 石家庄铁道大学信息学院**

**实验1：机器学习案例实现**

**本次实验属于验证型实验，通过本次实验学生将掌握以下内容：**

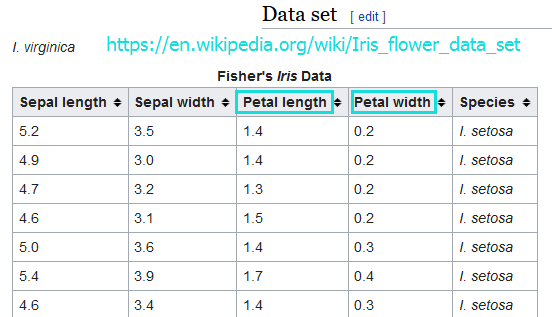
1、了解scikit-learn库的安装调用；

2、掌握熟悉感知器、 Logistic回归、SVM、决策树、随机森林、k-近邻等传统机器学习算法。

# [实验任务]：

**sklearn**官⽹**API**查询

**scikit-learn**中⾃带了⼀些数据集，⽐如说最著名的**Iris**数据集。**（实验中用其他数据集：日常生活中的数据或scikit-learn其他自带数据集）**



数据集中第**3**列和第**4**列数据表⽰花瓣的长度和宽度，类别标签列已经转成了数字，⽐如**0=Iris-Setosa,1=Iris- Versicolor,2=Iris-Virginica.**

1.导⼊python库和实验数据集 （参考代码）

**from IPython.display import Image**

**%matplotlib inline**

**# Added version check for recent scikit-learn 0.18 checks**

**from distutils.version import LooseVersion as Version from sklearn import \_\_version\_\_ as sklearn\_version**

**from sklearn import datasets import numpy as np**

**iris = datasets.load\_iris()**

**#**[**http://scikit-learn.org/stable/auto\_examples/datasets/plot\_iris\_dataset.html**](http://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html) **X = iris.data[:, [2, 3]]**

**y = iris.target #**取**species**列，类别

**print('Class labels:', np.unique(y))**

**#Output:Class labels: [0 1 2]**

# 2. 数据集切分 （参考代码）

## 把数据集切分成训练集和测试集，这⾥**70%**的训练集，**30%**的测试集

**if Version(sklearn\_version) < '0.18':**

**from sklearn.cross\_validation import train\_test\_split else:**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**X, y, test\_size=0.3, random\_state=0) #train\_test\_split**⽅法分割数据集

**X\_train.shape**

**#Output:(105, 2) X\_test.shape**

**#Output:(45, 2) X.shape**

**#Output:(150, 2) y\_train.shape**

**#Output: (105,) y\_test.shape**

**#Output: (45,)**

3. 对特征做标准化（参考代码）

⾮树形模型⼀般都要对特征数据进⾏标准化处理，避免数据波动的影响。处理后各维特征有**0**均值，单位⽅差。也叫**z-scor**规范化（零均值规范化）。计算方式是特征值减去均值，除以标准差。

**#scaler = sklearn.preprocessing.StandardScaler().fit(train)**

**#scaler.transform(train);scaler.transform(test)**

**#fit()**⽅法建模，**transform()**⽅法转换

**from sklearn.preprocessing import StandardScaler**

**sc = StandardScaler() #**初始化⼀个对象**sc**去对数据集作变换 **sc.fit(X\_train) #**⽤对象去拟合数据集**X\_train**，并且存下来拟合参数

**#Output:StandardScaler(copy=True, with\_mean=True, with\_std=True)**

**#type(sc.fit(X\_train))**

**#Output:sklearn.preprocessing.data.StandardScaler sc.scale\_ #sc.std\_**同样输出结果

**#Output:array([ 1.79595918, 0.77769705])**

**sc.mean\_**

**#Output:array([ 3.82857143, 1.22666667])**

**import numpy as np**

**X\_train\_std = sc.transform(X\_train) X\_test\_std = sc.transform(X\_test)**

**#test**标准化原理

**at=X\_train\_std[:5]\*sc.scale\_+sc.mean\_ a=X\_train[:5]**

**at==a**

**#Output:**

**#array([[ True, True],**

**# [ True, True],**

**# [ True, True],**

**# [ True, True],**

**# [ True, True]], dtype=bool)**

1. 各种算法分类及可视化 （参考代码）

4.1 构造可视化函数

下⾯各算法中，通过**plot*\_decision\_*region**函数可视化，⽅便直观看分类结果

#构造可视化函数 def **plot*\_decision\_*region**

**from matplotlib.colors import ListedColormap import matplotlib.pyplot as plt**

**import warnings**

**def versiontuple(v):#Numpy**版本检测函数 **return tuple(map(int, (v.split("."))))**

**def plot\_decision\_regions(X, y, classifier, test\_idx=None, resolution=0.02):**

**#**画决策边界**,X**是特征，**y**是标签，**classifier**是分类器，**test\_idx**是测试集序号

**# setup marker generator and color map markers = ('s', 'x', 'o', '^', 'v')**

**colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan') cmap = ListedColormap(colors[:len(np.unique(y))])**

**# plot the decision surface**

**x1\_min, x1\_max = X[:, 0].min() - 1, X[:, 0].max() + 1 #**第⼀个特征取值范围作为横轴 **x2\_min, x2\_max = X[:, 1].min() - 1, X[:, 1].max() + 1 #**第⼆个特征取值范围作为纵轴 **xx1, xx2 = np.meshgrid(np.arange(x1\_min, x1\_max, resolution),**

**np.arange(x2\_min, x2\_max, resolution)) #reolution**是⽹格剖分粒度，**xx1**和**xx2**数组维度⼀样 **Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)**

**#classifier**指定分类器，**ravel**是数组展平；**Z**的作⽤是对组合的⼆种特征进⾏预测 **Z = Z.reshape(xx1.shape) #Z**是列向量

**plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)**

**#contourf(x,y,z)**其中**x**和**y**为两个等长⼀维数组，**z**为⼆维数组，指定每⼀对**xy**所对应的**z**值。

**#**对等⾼线间的区域进⾏填充（使⽤不同的颜⾊） **plt.xlim(xx1.min(), xx1.max())**

**plt.ylim(xx2.min(), xx2.max())**

**for idx, cl in enumerate(np.unique(y)): plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],**

**alpha=0.8, c=cmap(idx),**

**marker=markers[idx], label=cl) #**全数据集，不同类别样本点的特征作为坐标**(x,y)**，⽤不同颜⾊画散点图

**# highlight test samples if test\_idx:**

**# plot all samples**

**if not versiontuple(np.\_\_version\_\_) >= versiontuple('1.9.0'): X\_test, y\_test = X[list(test\_idx), :], y[list(test\_idx)] warnings.warn('Please update to NumPy 1.9.0 or newer')**

**else:**

**X\_test, y\_test = X[test\_idx, :], y[test\_idx] #X\_test**取测试集样本两列特征，**y\_test**取测试集标签

**plt.scatter(X\_test[:, 0],**

**X\_test[:, 1], c='',**

**alpha=1.0, linewidths=1, marker='o',**

**s=55, label='test set') #c**设置颜⾊，测试集不同类别的实例点画图不区别颜⾊

4.2感知器分类算法（三分类） （参考代码）

**from sklearn.linear\_model import Perceptron**

**#**[**http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.Perceptron.html   
#sklearn.linear\_model.Perceptron**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html%20#sklearn.linear_model.Perceptron)

**#ppn = Perceptron(n\_iter=40, eta0=0.1, random\_state=0) ppn = Perceptron() #y=w.x+b**

**ppn.fit(X\_train\_std, y\_train)**

**#Output:Perceptron(alpha=0.0001, class\_weight=None, eta0=1.0, fit\_intercept=True,**

**# n\_iter=5, n\_jobs=1, penalty=None, random\_state=0, shuffle=True,**

**# verbose=0, warm\_start=False) ppn.coef\_ #**分类决策函数中的特征系数**w**

**#Output:array([[-1.48746619, -1.1229737 ],**

**# [ 3.0624304 , -2.18594118],**

**# [ 2.9272062 , 2.64027405]])**

**ppn.intercept\_ #**分类决策函数中的偏置项**b**

**#Output:array([-1., 0., -2.])**

**y\_pred = ppn.predict(X\_test\_std) #**对测试集做类别预测 **y\_pred**

**#Output:array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1, 0,**

**# 0, 2, 0, 0, 1, 0, 0, 2, 1, 0, 2, 2, 1, 0, 2, 1, 1, 2, 0, 2, 0, 0])**

**y\_test**

**#Output:array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1, 0,**

**# 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 1, 1, 1, 2, 0, 2, 0, 0])**

**y\_pred == y\_test**

**#Output:array([ True, True, True, True, True, True, True, True, True,**

**# True, False, True, True, True, True, True, True, True,**

**# True, True, True, True, True, True, True, True, True,**

**# True, False, True, True, True, True, True, True, True,**

**# True, False, True, True, True, True, True, True, True], dtype=bool) print('Misclassified samples: %d' % (y\_test != y\_pred).sum())**

**#Output:Misclassified samples: 3**

**from sklearn.metrics import accuracy\_score**

**print('Accuracy: %.2f' % accuracy\_score(y\_test, y\_pred)) #**预测准确度**,(len(y\_test)-3)/len(y\_test):0.9333333333333333**

**#Output:Accuracy: 0.93**

## ⽤标准化的数据做⼀个感知器分类器

**%matplotlib inline**

**X\_combined\_std = np.vstack((X\_train\_std, X\_test\_std)) #shape**是**(150,2) y\_combined = np.hstack((y\_train, y\_test)) #shape**是**(150,)**

**plot\_decision\_regions(X=X\_combined\_std, y=y\_combined, classifier=ppn, test\_idx=range(105, 150))**

**plt.xlabel('petal length [standardized]') plt.ylabel('petal width [standardized]') plt.legend(loc='upper left')**

**plt.tight\_layout() #**紧凑显⽰图⽚，居中显⽰；避免出现叠影

**# plt.savefig('./figures/iris\_perceptron\_scikit.png', dpi=300) plt.show()**

4.3 Logistic回归分类算法（三分类） （参考代码）

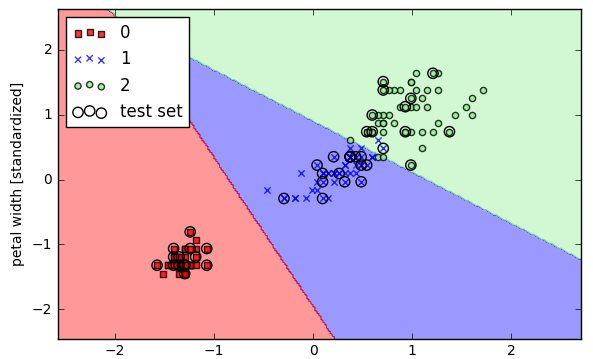
**from sklearn.linear\_model import LogisticRegression lr = LogisticRegression(C=1000.0, random\_state=0)**

**#**[**http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html#sklearn.linear\_model.LogisticRegression**](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression) **lr.fit(X\_train\_std, y\_train)**

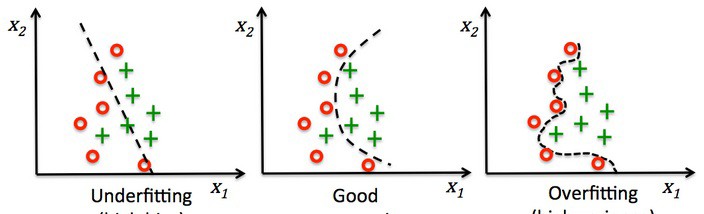
**plot\_decision\_regions(X\_combined\_std, y\_combined, classifier=lr, test\_idx=range(105, 150)) plt.xlabel('petal length [standardized]')**

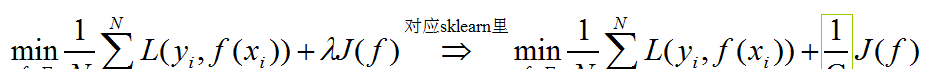
**plt.ylabel('petal width [standardized]') plt.legend(loc='upper left') plt.tight\_layout()**

**# plt.savefig('./figures/logistic\_regression.png', dpi=300) plt.show()**



过拟合**/overfittin**与正则化**/regularization**





**weights, params = [], []**

**for c in range(-5,6):**

**lr = LogisticRegression(C=10\*\*c, random\_state=0) #**默认**L2**正则，**C**是正则化系数的倒数**(C**越⼩，特征权重越⼩**) lr.fit(X\_train\_std, y\_train)**

**weights.append(lr.coef\_[1]) ##############################lr.coef\_[1] params.append(10\*\*c)**

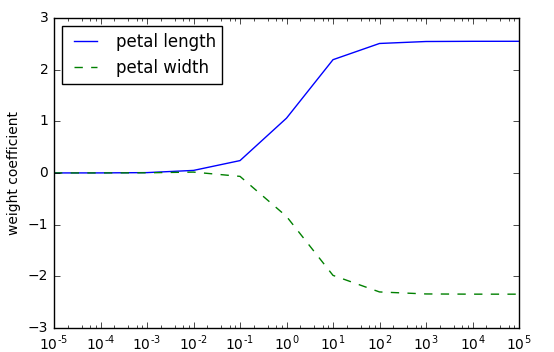
**weights = np.array(weights) plt.plot(params,weights[:, 0],label='petal length')**

**plt.plot(params,weights[:, 1],linestyle='--',label='petal width') plt.ylabel('weight coefficient')**

**plt.xlabel('C') plt.legend(loc='upper left')**

**plt.xscale('log') #**在**x**轴上画对数坐标轴

**# plt.savefig('./figures/regression\_path.png', dpi=300) plt.show()**



4.4 SVM分类算法（三分类） （参考代码）

**from sklearn.svm import SVC**

**#**[**http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC**](http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC) **svm = SVC(kernel='linear', C=1.0, random\_state=0)**

**svm.fit(X\_train\_std, y\_train)**

**plot\_decision\_regions(X\_combined\_std, y\_combined, classifier=svm, test\_idx=range(105, 150)) plt.xlabel('petal length [standardized]')**

**plt.ylabel('petal width [standardized]') plt.legend(loc='upper left') plt.tight\_layout()**

**# plt.savefig('./figures/support\_vector\_machine\_linear.png', dpi=300) plt.show()**

# 

## 上图利⽤的是线性**SVM**分类，但是从上图可见有些点被分错类了，进⼀步，考虑利⽤核函数进⾏⾮线性分类

**from sklearn.svm import SVC**

**svm = SVC(kernel='rbf',random\_state=0,gamma=0.2,C=1.0) svm.fit(X\_train\_std,y\_train)**

**plot\_decision\_regions(X\_combined\_std, y\_combined, classifier=svm, test\_idx=range(105, 150)) plt.xlabel('petal length [standardized]')**

**plt.ylabel('petal width [standardized]') plt.legend(loc='upper left') plt.tight\_layout()**

**# plt.savefig('./figures/support\_vector\_machine\_rbf\_iris\_1.png', dpi=300) plt.show()**

# 

4.5 决策树分类算法（三分类） （参考代码）

**from sklearn.tree import DecisionTreeClassifier**

**#**[**http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier**](http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier) **tree = DecisionTreeClassifier(criterion='entropy', max\_depth=3, random\_state=0)**

**#**特征选择的度量，**entropy**是信息增益；**max\_depth**参数是数的最⼤深度

**tree.fit(X\_train, y\_train)**

**X\_combined = np.vstack((X\_train, X\_test)) y\_combined = np.hstack((y\_train, y\_test))**

**plot\_decision\_regions(X\_combined, y\_combined, classifier=tree, test\_idx=range(105, 150)) plt.xlabel('petal length [cm]')**

**plt.ylabel('petal width [cm]') plt.legend(loc='upper left') plt.tight\_layout()**

**# plt.savefig('./figures/decision\_tree\_decision.png', dpi=300) plt.show()**

# 

4.6 随机森林分类算法（三分类） （参考代码）

**from sklearn.ensemble import RandomForestClassifier**

**forest = RandomForestClassifier(criterion='entropy', n\_estimators=10, random\_state=1, n\_jobs=2)**

**#**[**http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier**](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier)

**#criterion**特征选择度量，**n\_estimators**随机森林中单棵树数⽬，**n\_jobs**设置并⾏⽣成树模型得数⽬ **forest.fit(X\_train, y\_train)**

**plot\_decision\_regions(X\_combined, y\_combined, classifier=forest, test\_idx=range(105, 150)) plt.xlabel('petal length [cm]')**

**plt.ylabel('petal width [cm]') plt.legend(loc='upper left') plt.tight\_layout()**

**# plt.savefig('./figures/random\_forest.png', dpi=300) plt.show()**

# 

4.7 k近邻分类算法（三分类） （参考代码）

**from sklearn.neighbors import KNeighborsClassifier**

**knn = KNeighborsClassifier(n\_neighbors=5, p=2, metric='minkowski')**

**#**[**http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier**](http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier)

**#p**是度量范数，**metric='minkowski'**距离度量标准**2**范数 **knn.fit(X\_train\_std, y\_train)**

**plot\_decision\_regions(X\_combined\_std, y\_combined, classifier=knn, test\_idx=range(105, 150)) plt.xlabel('petal length [standardized]')**

**plt.ylabel('petal width [standardized]') plt.legend(loc='upper left') plt.tight\_layout()**

**# plt.savefig('./figures/k\_nearest\_neighbors.png', dpi=300) plt.show()**

# 

# 简述题：结合自己的专业谈谈你对机器学习和scikit-learn库的认识