北京郵電大學

Beijing University of Posts and Telecommunications

数据科学

题目：第二次实验

学院：信息与通信工程学院

班级：2019211127

姓名：王兴睿

学号：2018211756

目录

复现课件中线性 SVM、决策树、朴素贝叶斯分类的示例，并相对课件代码作出如下作图修改（必做）

(1) 设定支持向量分类器的惩罚为 0.05

In [2]:
```
#设置显示结果的图的标题
names = ["Linear SVM","Decision Tree","Naive Bayes"]
#设置分类器，用到线性SVM，决策树，朴素贝叶斯
classifiers = [
    SVC(kernel="linear",C=0.05),
    DecisionTreeClassifier(random_state=0,max_depth=5),
    GaussianNB(priors=[0.5,0.5])
]
```

(2) 对朴素贝叶斯分类器的先验概率进行设定（可随机设定）

In [2]:
```
#设置显示结果的图的标题
names = ["Linear SVM","Decision Tree","Naive Bayes"]
#设置分类器，用到线性SVM，决策树，朴素贝叶斯
classifiers = [
    SVC(kernel="linear",C=0.05),
    DecisionTreeClassifier(random_state=0,max_depth=5),
    GaussianNB(priors=[0.45,0.55])
]
```

(3) 在每张结果图上展示图例

```
    ax.set_ylim(yy.min(),yy.max())
    ax.set_xticks(())
    ax.set_yticks(())
    plt.legend(loc=2)
    i+=1
    #分别对每个分类器做训练测试
    for name,clf in zip(names,classifiers):
        ax=plt.subplot(len(datasets),len(classifiers)+1,i)
        clf.fit(X_train,y_train)#训练集训练分类器
        score=clf.score(X_test,y_test)#测试集测试分类器
        if hasattr (clf,"decision_function"):
            Z=clf.decision_function(np.c_[xx.ravel(),yy.ravel()])
```

```
        if ds_cnt==0:
            #画子图标题
            ax.set_title(name)
            #显示测试正确率
        ax.text(xx.max()-.3,yy.min()+.3,('%.3f'% score).lstrip('0'),
                size=20,horizontalalignment='right')
        plt.legend(loc=2)#左上角加图例
        i+=1
plt.tight_layout()
plt.show()
```

(4) 修改散点颜色为黄和绿

```
    #先展示输入数据集
    cm=ListedColormap(['red','blue'])#设置分割面颜色
    cm_bright=ListedColormap(['yellow','green'])#设置散点颜色为黄和绿
    ax=plt.subplot(len(datasets),len(classifiers)+1,i)#划分子面
    if ds_cnt==0:
        ax.set_title("Input data")
    #画训练集散点
```

(5) 测试结果的正确率保留三位小数展示

```
        if ds_cnt==0:
            #画子图标题
            ax.set_title(name)
            #显示测试正确率
        ax.text(xx.max()-.3,yy.min()+.3,('%.3f'% score).lstrip('0'),
                size=20,horizontalalignment='right')
        i+=1
plt.tight_layout()
plt.show()
```

(6) 复现课件中的代码和结果如下：

源代码：

```
#导入包
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
#随机划分训练集、测试集
from sklearn.preprocessing import StandardScaler
#标准化数数据集
#导入三种分别的数据集，都是具体数据函数生成
from sklearn.datasets import make_moons,make_circles,make_classification
from  sklearn.model_selection import train_test_split
#导入三分类器
from sklearn.svm import SVC  #支持向量机
from sklearn.tree import DecisionTreeClassifier  #决策树分类器
from sklearn.naive_bayes import GaussianNB  #先验为高斯分布的朴素贝叶斯分类器
#设置显示结果的图的标题
names = ["Linear SVM","Decision Tree","Naive Bayes"]
#设置分类器，用到线性 SVM，决策树，朴素贝叶斯
classifiers = [
    SVC(kernel="linear",C=0.05),
    DecisionTreeClassifier(random_state=0,max_depth=5),
    GaussianNB(priors=[0.45,0.55])
]
#设置一个用于分类的数据集，这里设置为线型可分的数据集，输入变量设置为两个特征
X,y  =  make_classification(n_features=2,n_redundant=0,n_informative=2,random_state=1,n_clusters_per
_class=1)
rng = np.random.RandomState(2)#设置一个伪随机种子
X+=2*rng.uniform(size=X.shape)#对 x 变量上加随机扰动
linearly_separable=(X,y)#将上述得到的 x,y 构建为一个线性可分的数据集
datasets=[make_moons(noise=0.1,random_state=0),
          make_circles(noise=0.1,factor=0.5,random_state=1),
         linearly_separable
          ]
#设置显示结果图的大小
figure = plt.figure(figsize=(27,9))
```

```
i=1
#对每个数据集做训练及测试
for ds_cnt,ds in enumerate(datasets):
    X,y=ds
    X=StandardScaler().fit_transform(X)#标准化数据
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.4,random_state=42)#分割数据集
    x_min,x_max=X[:,0].min()-.5,X[:,0].max()+.5
    y_min,y_max=X[:,1].min()-.5,X[:,1].max()+.5
    h=.02#设置网络的步长
    xx,yy=np.meshgrid(np.arange(x_min,x_max,h),
                      np.arange(y_min,y_max,h)
                      )
    #先展示输入数据集
    cm=ListedColormap(['red','blue'])#设置分割面颜色
    cm_bright=ListedColormap(['yellow','green'])#设置散点颜色为黄和绿
    ax=plt.subplot(len(datasets),len(classifiers)+1,i)#划分子面
    if ds_cnt==0:
        ax.set_title("Input data")
    #画训练集散点
    ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
               edgecolors='k',marker='o',label='train set')
    #画测试集散点
    ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,alpha=0.6,
               edgecolors='k',marker='x',label='test set')
    #画坐标轴
    ax.set_xlim(xx.min(),xx.max())
    ax.set_ylim(yy.min(),yy.max())
    ax.set_xticks(())
    ax.set_yticks(())
    plt.legend(loc=2)
    i+=1
    #分别对每个分类器做训练测试
    for name,clf in zip(names,classifiers):
        ax=plt.subplot(len(datasets),len(classifiers)+1,i)
        clf.fit(X_train,y_train)#训练集训练分类器
        score=clf.score(X_test,y_test)#测试集测试分类器
        if hasattr (clf,"decision_function"):
            Z=clf.decision_function(np.c_[xx.ravel(),yy.ravel()])
        else:
            Z=clf.predict_proba(np.c_[xx.ravel(),yy.ravel()])[:,1]
        Z=Z.reshape(xx.shape) #分类结果用等高线函数画出
        ax.contourf(xx,yy,Z,cmap=cm,alpha=.4)
        #画训练集点
        ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
```

```
                    edgecolors='k',marker='o',label='train set')
            #画测试集点
            ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,
                    edgecolors='k',marker='x',label='test set')
            #画坐标轴
            ax.set_xlim(xx.min(),xx.max())
            ax.set_ylim(yy.min(),yy.max())
            ax.set_xticks(())
            ax.set_yticks(())
            if ds_cnt==0:
                #画子图标题
                ax.set_title(name)
                #显示测试正确率
            ax.text(xx.max()-.3,yy.min()+.3,('%.3f'% score).lstrip('0'),
                    size=20,horizontalalignment='right')
            plt.legend(loc=2)#左上角加图例
            i+=1
    plt.tight_layout()
    plt.show()
```

最终结果：

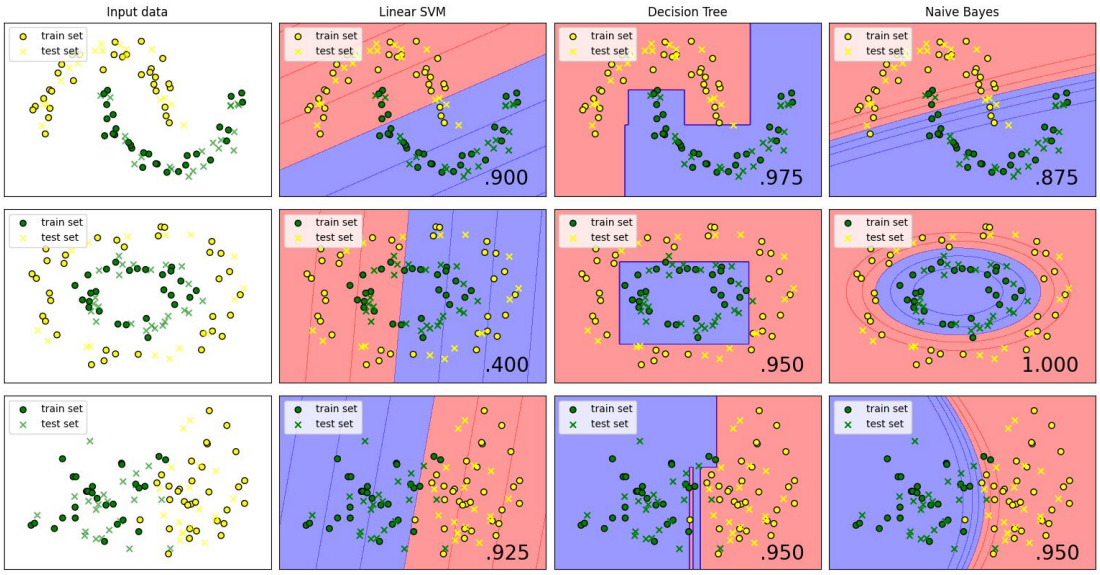


图 1 结果图

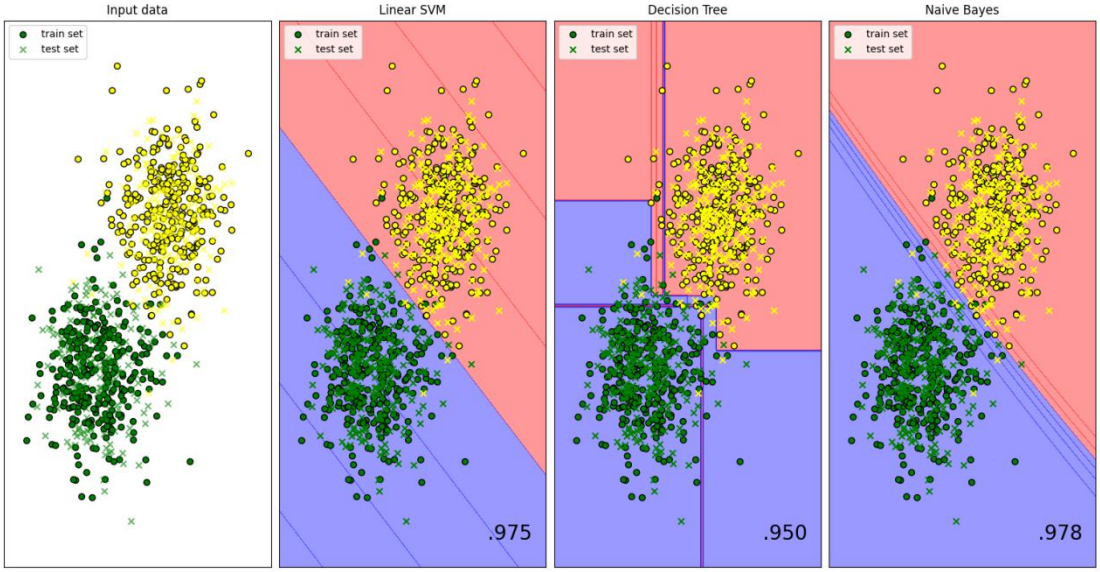
创新与拓展（选做）：

(1) 自主选取其他的数据集，采用上述三类分类器进行分类，展示分类结果（源代码见附录）

```
#标准化数数据集
#导入三种分别的数据集，都是具体数据函数生成
from sklearn.datasets import make_blobs,make_classification
from  sklearn.model_selection import train_test_split
#导入三分类器
from sklearn.svm import SVC  #支持向量机
from sklearn.tree import DecisionTreeClassifier  #决策树分类器
from sklearn.naive_bayes import GaussianNB  #先验为高斯分布的朴素
```

```
X,y = make_classification(n_features=2,n_redundant=0,n_informative=2,random_state=1,n_c
rng = np.random.RandomState(2)#设置一个伪随机种子
X+=2*rng.uniform(size=X.shape)#对x变量上加随机扰动
linearly_separable=(X,y)#将上述得到的x,y构建为一个线性可分的数据集
datasets=[make_blobs(n_samples=1000,n_features=2,centers=2,cluster_std=1.5)
          ]
#设置显示结果图的大小
figure = plt.figure(figsize=(27,9))
i=1
#对每个数据集做训练及测试
for ds_cnt,ds in enumerate(datasets):
```

分类结果如下：



(2) 探究分类器的参数对于分类结果的影响并进行文字分析（选做）

✓ 支持向量分类器不同核函数对于结果的影响（源代码见附录）

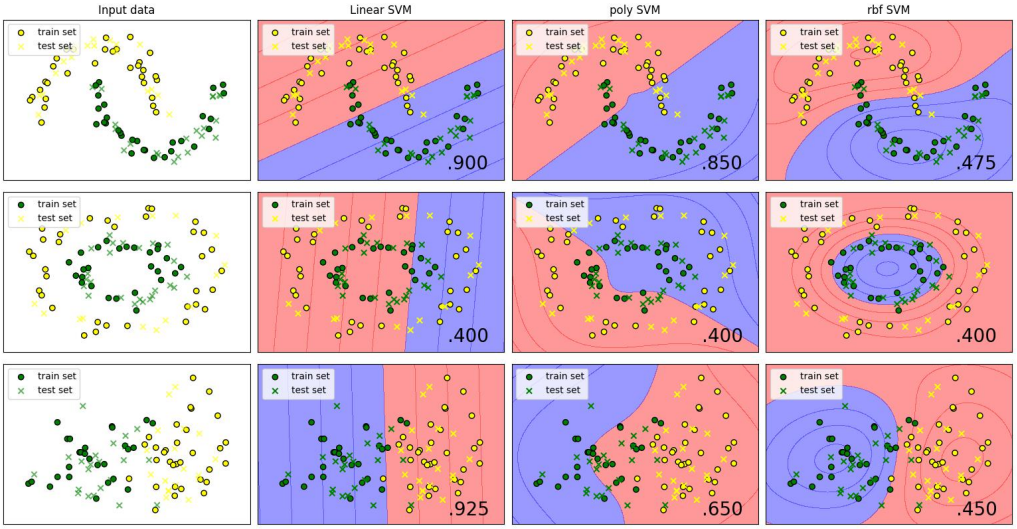
采用核方法，能够很方便地产生非线性分类边界。

Linear：线性核，会产生线性分类边界。一般来说它的计算效率最高，而且需要数据最少。

poly ：多项式核，会产生多项式分类边界。

rbf：径向基函数，也就是高斯核，是根据与每一个支持向量的距离来决定分类边界的。它的映射到无线维的。它是最灵活的方法，但是也需要最多的数据。

```
names = ["Linear SVM","poly SVM","rbf SVM"]
#设置分类器，用到线性SVM，决策树，朴素贝叶斯
classifiers = [
    SVC(kernel="linear",C=0.025),
    SVC(kernel="poly",C=0.025),
    SVC(kernel="rbf",C=0.025)
]
#设置一个用于分类的数据集，这里设置为线型可分的数据集，输入变量设置
X,y = make_classification(n_features=2,n_redundant=0,n_informative
rng = np.random.RandomState(2)#设置一个伪随机种子
```



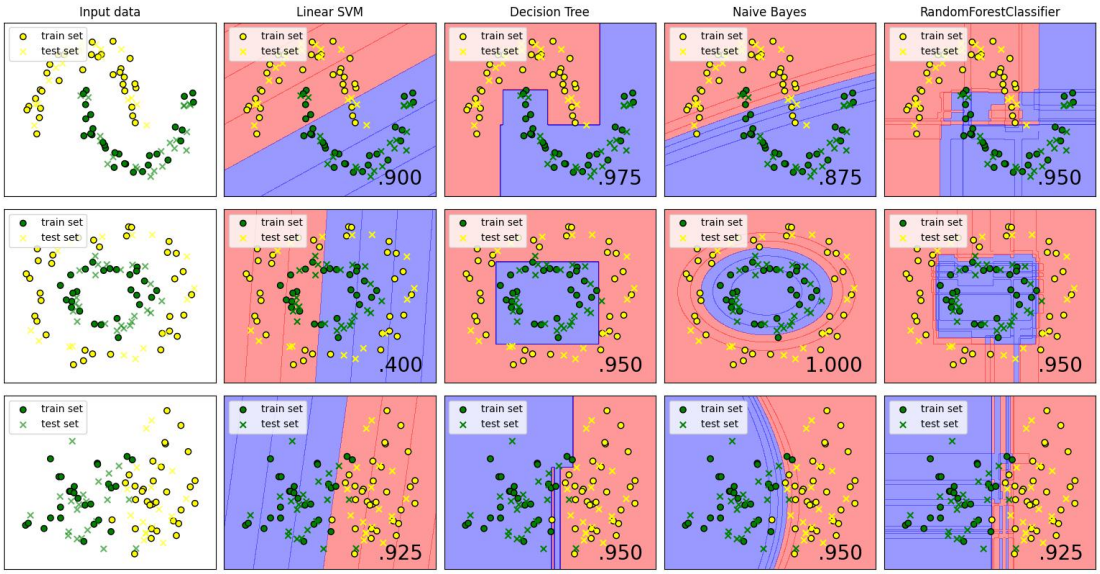
核方法挑选了线性核、多项式核和高斯核，从上图的结果来看，线性核的正确率是三者当中最高的，因为线性核需要数据最少，而高斯核更灵活，所以在训练集上效果是最好的，但是其需要最多的数据，并且需要担心过拟合的问题，其在测试集上的正确率并不是最高的。

(3) 其他分类方法的效果的对比分析（K 近邻，随机森林等）（源代码见附录）

在这一问，我主要对比了随机森林与其它分类方法，

```
14 from sklearn.naive_bayes import GaussianNB  #先验为高斯分布的朴素贝叶斯分类器
15 from sklearn.ensemble import RandomForestClassifier #随机森林
16 #设置显示结果的图的标题
17 names = ["Linear SVM","Decision Tree","Naive Bayes","RandomForestClassifier"]
18 #设置分类器，用到线性SVM，决策树，朴素贝叶斯
19 classifiers = [
20     SVC(kernel="linear",C=0.05),
```

结果如下所示：



从上图可以看出，随机森林在测试集上的分类准确率与决策树和朴素贝叶斯分类相近，整体效果优于线性 SVM。

附录

创新与拓展(1)源代码：

```
#导入包
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
#随机划分训练集、测试集
from sklearn.preprocessing import StandardScaler
#标准化数数据集
#导入三种分别的数据集，都是具体数据函数生成
from sklearn.datasets import make_moons,make_circles,make_classification
from  sklearn.model_selection import train_test_split
#导入三分类器
from sklearn.svm import SVC  #支持向量机
from sklearn.tree import DecisionTreeClassifier  #决策树分类器
from sklearn.naive_bayes import GaussianNB  #先验为高斯分布的朴素贝叶斯分类器
#设置显示结果的图的标题
names = ["Linear SVM","Decision Tree","Naive Bayes"]
#设置分类器，用到线性 SVM，决策树，朴素贝叶斯
classifiers = [
    SVC(kernel="linear",C=0.025),
    DecisionTreeClassifier(random_state=0,max_depth=5),
    GaussianNB(priors=[0.5,0.5])
]
#设置一个用于分类的数据集，这里设置为线型可分的数据集，输入变量设置为两个特征
X,y  =  make_classification(n_features=2,n_redundant=0,n_informative=2,random_state=1,n_clusters_per
_class=1)
rng = np.random.RandomState(2)#设置一个伪随机种子
X+=2*rng.uniform(size=X.shape)#对 x 变量上加随机扰动
linearly_separable=(X,y)#将上述得到的 x,y 构建为一个线性可分的数据集
datasets=[make_moons(noise=0.1,random_state=0),
          make_circles(noise=0.1,factor=0.5,random_state=1),
         linearly_separable
          ]
#设置显示结果图的大小
figure = plt.figure(figsize=(27,9))
i=1
#对每个数据集做训练及测试
for ds_cnt,ds in enumerate(datasets):
    X,y=ds
    X=StandardScaler().fit_transform(X)#标准化数据
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.4,random_state=42)#分割数据集
    x_min,x_max=X[:,0].min()-.5,X[:,0].max()+.5
```

```
    y_min,y_max=X[:,1].min()-.5,X[:,1].max()+.5
    h=.02#设置网络的步长
    xx,yy=np.meshgrid(np.arange(x_min,x_max,h),
                      np.arange(y_min,y_max,h)
                      )
    #先展示输入数据集
    cm=ListedColormap(['red','blue'])#设置分割面颜色
    cm_bright=ListedColormap(['yellow','green'])#设置散点颜色为黄和绿
    ax=plt.subplot(len(datasets),len(classifiers)+1,i)#划分子面
    if ds_cnt==0:
        ax.set_title("Input data")
    #画训练集散点
    ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
               edgecolors='k',marker='o',label='train set')
    #画测试集散点
    ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,alpha=0.6,
               edgecolors='k',marker='x',label='test set')
    #画坐标轴
    ax.set_xlim(xx.min(),xx.max())
    ax.set_ylim(yy.min(),yy.max())
    ax.set_xticks(())
    ax.set_yticks(())
    plt.legend(loc=2)
    i+=1
    #分别对每个分类器做训练测试
    for name,clf in zip(names,classifiers):
        ax=plt.subplot(len(datasets),len(classifiers)+1,i)
        clf.fit(X_train,y_train)#训练集训练分类器
        score=clf.score(X_test,y_test)#测试集测试分类器
        if hasattr (clf,"decision_function"):
            Z=clf.decision_function(np.c_[xx.ravel(),yy.ravel()])
        else:
            Z=clf.predict_proba(np.c_[xx.ravel(),yy.ravel()])[:,1]
        Z=Z.reshape(xx.shape) #分类结果用等高线函数画出
        ax.contourf(xx,yy,Z,cmap=cm,alpha=.4)
        #画训练集点
        ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
                   edgecolors='k',marker='o',label='train set')
        #画测试集点
        ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,
                   edgecolors='k',marker='x',label='test set')
        #画坐标轴
        ax.set_xlim(xx.min(),xx.max())
        ax.set_ylim(yy.min(),yy.max())
```

```
            ax.set_xticks(())
            ax.set_yticks(())
            if ds_cnt==0:
                #画子图标题
                ax.set_title(name)
                #显示测试正确率
            ax.text(xx.max()-.3,yy.min()+.3,('%.3f'% score).lstrip('0'),
                    size=20,horizontalalignment='right')
            plt.legend(loc=2)#左上角加图例
            i+=1
    plt.tight_layout()
    plt.show()
```

创新与拓展(2)源代码

```
#导入包
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
#随机划分训练集、测试集
from sklearn.preprocessing import StandardScaler
#标准化数数据集
#导入三种分别的数据集，都是具体数据函数生成
from sklearn.datasets import make_moons,make_circles,make_classification
from  sklearn.model_selection import train_test_split
#导入三分类器
from sklearn.svm import SVC  #支持向量机
from sklearn.tree import DecisionTreeClassifier  #决策树分类器
from sklearn.naive_bayes import GaussianNB  #先验为高斯分布的朴素贝叶斯分类器
#设置显示结果的图的标题
names = ["Linear SVM","poly SVM","rbf SVM"]
#设置分类器，用到线性 SVM，决策树，朴素贝叶斯
classifiers = [
    SVC(kernel="linear",C=0.025),
    SVC(kernel="poly",C=0.025),
    SVC(kernel="rbf",C=0.025)
]
#设置一个用于分类的数据集，这里设置为线型可分的数据集，输入变量设置为两个特征
X,y  =  make_classification(n_features=2,n_redundant=0,n_informative=2,random_state=1,n_clusters_per
_class=1)
rng = np.random.RandomState(2)#设置一个伪随机种子
X+=2*rng.uniform(size=X.shape)#对 x 变量上加随机扰动
linearly_separable=(X,y)#将上述得到的 x,y 构建为一个线性可分的数据集
datasets=[make_moons(noise=0.1,random_state=0),
```

```
          make_circles(noise=0.1,factor=0.5,random_state=1),
          linearly_separable
          ]
#设置显示结果图的大小
figure = plt.figure(figsize=(27,9))
i=1
#对每个数据集做训练及测试
for ds_cnt,ds in enumerate(datasets):
    X,y=ds
    X=StandardScaler().fit_transform(X)#标准化数据
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.4,random_state=42)#分割数据集
    x_min,x_max=X[:,0].min()-.5,X[:,0].max()+.5
    y_min,y_max=X[:,1].min()-.5,X[:,1].max()+.5
    h=.02#设置网络的步长
    xx,yy=np.meshgrid(np.arange(x_min,x_max,h),
                      np.arange(y_min,y_max,h)
                      )
    #先展示输入数据集
    cm=ListedColormap(['red','blue'])#设置分割面颜色
    cm_bright=ListedColormap(['yellow','green'])#设置散点颜色为黄和绿
    ax=plt.subplot(len(datasets),len(classifiers)+1,i)#划分子面
    if ds_cnt==0:
        ax.set_title("Input data")
    #画训练集散点
    ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
               edgecolors='k',marker='o',label='train set')
    #画测试集散点
    ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,alpha=0.6,
               edgecolors='k',marker='x',label='test set')
    #画坐标轴
    ax.set_xlim(xx.min(),xx.max())
    ax.set_ylim(yy.min(),yy.max())
    ax.set_xticks(())
    ax.set_yticks(())
    plt.legend(loc=2)
    i+=1
    #分别对每个分类器做训练测试
    for name,clf in zip(names,classifiers):
        ax=plt.subplot(len(datasets),len(classifiers)+1,i)
        clf.fit(X_train,y_train)#训练集训练分类器
        score=clf.score(X_test,y_test)#测试集测试分类器
        if hasattr (clf,"decision_function"):
            Z=clf.decision_function(np.c_[xx.ravel(),yy.ravel()])
        else:
```

```
            Z=clf.predict_proba(np.c_[xx.ravel(),yy.ravel()])[:,1]
        Z=Z.reshape(xx.shape) #分类结果用等高线函数画出
        ax.contourf(xx,yy,Z,cmap=cm,alpha=.4)
        #画训练集点
        ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
                   edgecolors='k',marker='o',label='train set')
        #画测试集点
        ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,
                   edgecolors='k',marker='x',label='test set')
        #画坐标轴
        ax.set_xlim(xx.min(),xx.max())
        ax.set_ylim(yy.min(),yy.max())
        ax.set_xticks(())
        ax.set_yticks(())
        if ds_cnt==0:
            #画子图标题
            ax.set_title(name)
            #显示测试正确率
        ax.text(xx.max()-.3,yy.min()+.3,('%.3f'% score).lstrip('0'),
                size=20,horizontalalignment='right')
        plt.legend(loc=2)#左上角加图例
        i+=1
plt.tight_layout()
plt.show()
```

创新与拓展(3)：

```
#导入包
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
#随机划分训练集、测试集
from sklearn.preprocessing import StandardScaler
#标准化数数据集
#导入三种分别的数据集，都是具体数据函数生成
from sklearn.datasets import make_moons,make_circles,make_classification
from  sklearn.model_selection import train_test_split
#导入三分类器
from sklearn.svm import SVC  #支持向量机
from sklearn.tree import DecisionTreeClassifier  #决策树分类器
from sklearn.naive_bayes import GaussianNB  #先验为高斯分布的朴素贝叶斯分类器
from sklearn.ensemble import RandomForestClassifier #随机森林
#设置显示结果的图的标题
names = ["Linear SVM","Decision Tree","Naive Bayes","RandomForestClassifier"]
```

```
#设置分类器，用到线性 SVM，决策树，朴素贝叶斯
classifiers = [
    SVC(kernel="linear",C=0.05),
    DecisionTreeClassifier(random_state=0,max_depth=5),
    GaussianNB(priors=[0.45,0.55]),
    RandomForestClassifier()
]
#设置一个用于分类的数据集，这里设置为线型可分的数据集，输入变量设置为两个特征
X,y = make_classification(n_features=2,n_redundant=0,n_informative=2,random_state=1,n_clusters_per_class=1)
rng = np.random.RandomState(2)#设置一个伪随机种子
X+=2*rng.uniform(size=X.shape)#对 x 变量上加随机扰动
linearly_separable=(X,y)#将上述得到的 x,y 构建为一个线性可分的数据集
datasets=[make_moons(noise=0.1,random_state=0),
          make_circles(noise=0.1,factor=0.5,random_state=1),
         linearly_separable
          ]
#设置显示结果图的大小
figure = plt.figure(figsize=(27,9))
i=1
#对每个数据集做训练及测试
for ds_cnt,ds in enumerate(datasets):
    X,y=ds
    X=StandardScaler().fit_transform(X)#标准化数据
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.4,random_state=42)#分割数据集
    x_min,x_max=X[:,0].min()-.5,X[:,0].max()+.5
    y_min,y_max=X[:,1].min()-.5,X[:,1].max()+.5
    h=.02#设置网络的步长
    xx,yy=np.meshgrid(np.arange(x_min,x_max,h),
                      np.arange(y_min,y_max,h)
                      )
    #先展示输入数据集
    cm=ListedColormap(['red','blue'])#设置分割面颜色
    cm_bright=ListedColormap(['yellow','green'])#设置散点颜色为黄和绿
    ax=plt.subplot(len(datasets),len(classifiers)+1,i)#划分子面
    if ds_cnt==0:
        ax.set_title("Input data")
    #画训练集散点
    ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
               edgecolors='k',marker='o',label='train set')
    #画测试集散点
    ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,alpha=0.6,
               edgecolors='k',marker='x',label='test set')
    #画坐标轴
```

```
    ax.set_xlim(xx.min(),xx.max())
    ax.set_ylim(yy.min(),yy.max())
    ax.set_xticks(())
    ax.set_yticks(())
    plt.legend(loc=2)
    i+=1
    #分别对每个分类器做训练测试
    for name,clf in zip(names,classifiers):
        ax=plt.subplot(len(datasets),len(classifiers)+1,i)
        clf.fit(X_train,y_train)#训练集训练分类器
        score=clf.score(X_test,y_test)#测试集测试分类器
        if hasattr (clf,"decision_function"):
            Z=clf.decision_function(np.c_[xx.ravel(),yy.ravel()])
        else:
            Z=clf.predict_proba(np.c_[xx.ravel(),yy.ravel()])[:,1]
        Z=Z.reshape(xx.shape) #分类结果用等高线函数画出
        ax.contourf(xx,yy,Z,cmap=cm,alpha=.4)
        #画训练集点
        ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
                   edgecolors='k',marker='o',label='train set')
        #画测试集点
        ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,
                   edgecolors='k',marker='x',label='test set')
        #画坐标轴
        ax.set_xlim(xx.min(),xx.max())
        ax.set_ylim(yy.min(),yy.max())
        ax.set_xticks(())
        ax.set_yticks(())
        if ds_cnt==0:
            #画子图标题
            ax.set_title(name)
            #显示测试正确率
        ax.text(xx.max()-.3,yy.min()+.3,('%.3f'% score).lstrip('0'),
                size=20,horizontalalignment='right')
        plt.legend(loc=2)#左上角加图例
        i+=1
plt.tight_layout()
plt.show()
```