

北京邮电大学

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）	3
(1) 设定支持向量分类器的惩罚为 0.05.....	3
(2) 对朴素贝叶斯分类器的先验概率进行设定（可随机设定）	3
(3) 在每张结果图上展示图例.....	3
(4) 修改散点颜色为黄和绿.....	3
(5) 测试结果的正确率保留三位小数展示.....	4
(6) 复现课件中的代码和结果如下：	4
创新与拓展（选做）：	6
(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）	6
(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）	7
(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）	8
附录.....	9

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]: 1 #设置显示结果的图的标题
2 names = ["Linear SVM", "Decision Tree", "Naive Bayes"]
3 #设置分类器，用到线性SVM，决策树，朴素贝叶斯
4 classifiers = [
5     SVC(kernel="linear", C=0.05),
6     DecisionTreeClassifier(random_state=0, max_depth=5),
7     GaussianNB(priors=[0.5, 0.5])
8 ]
```

(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]: 1 #设置显示结果的图的标题
2 names = ["Linear SVM", "Decision Tree", "Naive Bayes"]
3 #设置分类器，用到线性SVM，决策树，朴素贝叶斯
4 classifiers = [
5     SVC(kernel="linear", C=0.05),
6     DecisionTreeClassifier(random_state=0, max_depth=5),
7     GaussianNB(priors=[0.45, 0.55])
8 ]
```

(3) 在每张结果图上展示图例

```
26 ax.set_ylim(yy.min(), yy.max())
27 ax.set_xticks(())
28 ax.set_yticks(())
29 plt.legend(loc=2)
30 i+=1
31 #分别对每个分类器做训练测试
32 for name, clf in zip(names, classifiers):
33     ax=plt.subplot(len(datasets), len(classifiers)+1, i)
34     clf.fit(X_train, y_train) #训练集训练分类器
35     score=clf.score(X_test, y_test) #测试集测试分类器
36     if hasattr(clf, "decision_function"):
37         Z=clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
38
53 if ds_cnt==0:
54     #画子图标题
55     ax.set_title(name)
56     #显示测试正确率
57     ax.text(xx.max()-3, yy.min()+.3, ('%.3f'% score).lstrip('0'),
58            size=20, horizontalalignment='right')
59     plt.legend(loc=2) #左上角加图例
60     i+=1
61 plt.tight_layout()
62 plt.show()
```

(4) 修改散点颜色为黄和绿

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

(5) 测试结果的正确率保留三位小数展示

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

(6) 复现课件中的代码和结果如下：

源代码：	
1.	#导入包
2.	import numpy as np
3.	import matplotlib.pyplot as plt
4.	from matplotlib.colors import ListedColormap
5.	#随机划分训练集、测试集
6.	from sklearn.preprocessing import StandardScaler
7.	#标准化数据集
8.	#导入三种分别的数据集，都是具体数据函数生成
9.	from sklearn.datasets import make_moons,make_circles,make_classification
10.	from sklearn.model_selection import train_test_split
11.	#导入三分类器
12.	from sklearn.svm import SVC #支持向量机
13.	from sklearn.tree import DecisionTreeClassifier #决策树分类器
14.	from sklearn.naive_bayes import GaussianNB #先验为高斯分布的朴素贝叶斯分类器
15.	#设置显示结果的图的标题
16.	names = ["Linear SVM","Decision Tree","Naive Bayes"]
17.	#设置分类器，用到线性SVM，决策树，朴素贝叶斯
18.	classifiers = [
19.	SVC(kernel="linear",C=0.05),
20.	DecisionTreeClassifier(random_state=0,max_depth=5),
21.	GaussianNB(priors=[0.45,0.55])
22.]
23.	#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
24.	x,y = make_classification(n_features=2,n_redundant=0,n_informative=2,random_state=1,n_clusters_per_class=1)
25.	rng = np.random.RandomState(2)#设置一个伪随机种子
26.	x+=2*rng.uniform(size=x.shape)#对 x 变量上加随机扰动
27.	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
28.	datasets=[make_moons(noise=0.1,random_state=0),
29.	make_circles(noise=0.1,factor=0.5,random_state=1),
30.	linearly_separable
31.]
32.	#设置显示结果图的大小
33.	figure = plt.figure(figsize=(27,9))

```

34.     i=1
35.     #对每个数据集做训练及测试
36.     for ds_cnt,ds in enumerate(datasets):
37.         X,y=ds
38.         X=StandardScaler().fit_transform(X)#标准化数据
39.         X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.4,random_state=42)#分割数据集
40.         x_min,x_max=X[:,0].min()-.5,X[:,0].max()+.5
41.         y_min,y_max=X[:,1].min()-.5,X[:,1].max()+.5
42.         h=.02#设置网络的步长
43.         xx,yy=np.meshgrid(np.arange(x_min,x_max,h),
44.                             np.arange(y_min,y_max,h)
45.                             )
46.         #先展示输入数据集
47.         cm=ListedColormap(['red','blue'])#设置分割面颜色
48.         cm_bright=ListedColormap(['yellow','green'])#设置散点颜色为黄和绿
49.         ax=plt.subplot(len(datasets),len(classifiers)+1,i)#划分子面
50.         if ds_cnt==0:
51.             ax.set_title("Input data")
52.             #画训练集散点
53.             ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
54.                       edgecolors='k',marker='o',label='train set')
55.             #画测试集散点
56.             ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,alpha=0.6,
57.                       edgecolors='k',marker='x',label='test set')
58.             #画坐标轴
59.             ax.set_xlim(xx.min(),xx.max())
60.             ax.set_ylim(yy.min(),yy.max())
61.             ax.set_xticks(())
62.             ax.set_yticks(())
63.             plt.legend(loc=2)
64.             i+=1
65.         #分别对每个分类器做训练测试
66.         for name,clf in zip(names,classifiers):
67.             ax=plt.subplot(len(datasets),len(classifiers)+1,i)
68.             clf.fit(X_train,y_train)#训练集训练分类器
69.             score=clf.score(X_test,y_test)#测试集测试分类器
70.             if hasattr (clf,"decision_function"):
71.                 Z=clf.decision_function(np.c_[xx.ravel(),yy.ravel()])
72.             else:
73.                 Z=clf.predict_proba(np.c_[xx.ravel(),yy.ravel()])[:,1]
74.                 Z=Z.reshape(xx.shape) #分类结果用等高线函数画出
75.                 ax.contourf(xx,yy,Z,cmap=cm,alpha=.4)
76.                 #画训练集点
77.                 ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,

```

```

78.         edgecolors='k',marker='o',label='train set')
79.         #画测试集点
80.         ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,
81.                    edgecolors='k',marker='x',label='test set')
82.         #画坐标轴
83.         ax.set_xlim(xx.min(),xx.max())
84.         ax.set_ylim(yy.min(),yy.max())
85.         ax.set_xticks(())
86.         ax.set_yticks(())
87.         if ds_cnt==0:
88.             #画子图标题
89.             ax.set_title(name)
90.             #显示测试正确率
91.             ax.text(xx.max()-.3,yy.min()+.3,('%0.3f' % score).rstrip('0'),
92.                    size=20,horizontalalignment='right')
93.             plt.legend(loc=2)#左上角加图例
94.             i+=1
95. plt.tight_layout()
96. plt.show()

```

最终结果：

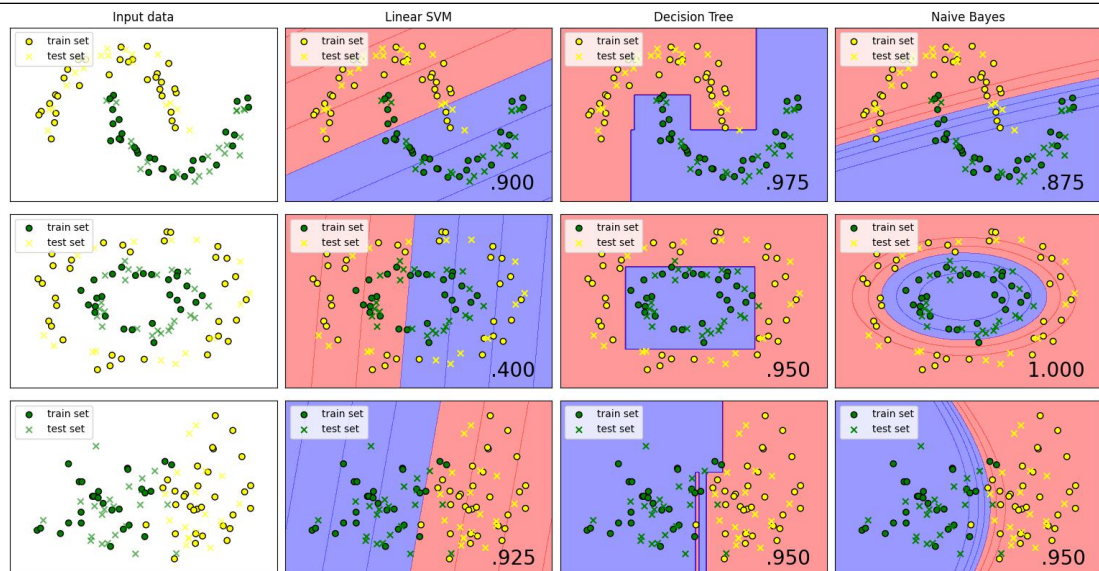


图 1 结果图

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）

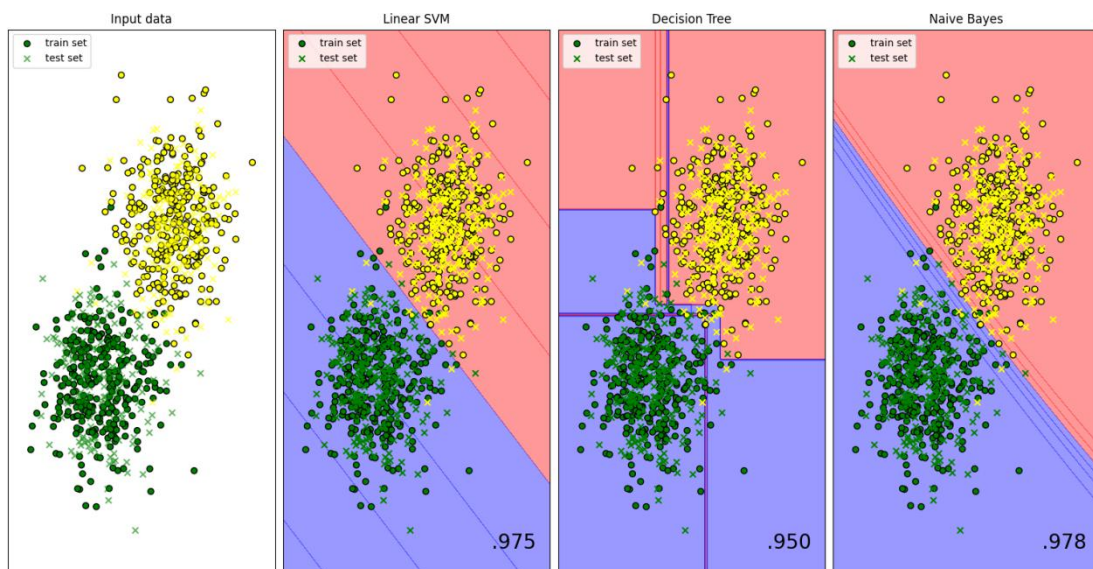

```

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

24 X, y = make_classification(n_features=2, n_redundant=0, n_informative=2, random_state=1, n_c
25 rng = np.random.RandomState(2) #设置一个伪随机种子
26 X+=2*rng.uniform(size=X.shape) #对x变量上加随机扰动
27 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
28 datasets=[make_blobs(n_samples=1000, n_features=2, centers=2, cluster_std=1.5)
29 ]
30 #设置显示结果图的大小
31 figure = plt.figure(figsize=(27,9))
32 i=1
33 #对每个数据集做训练及测试
34 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。

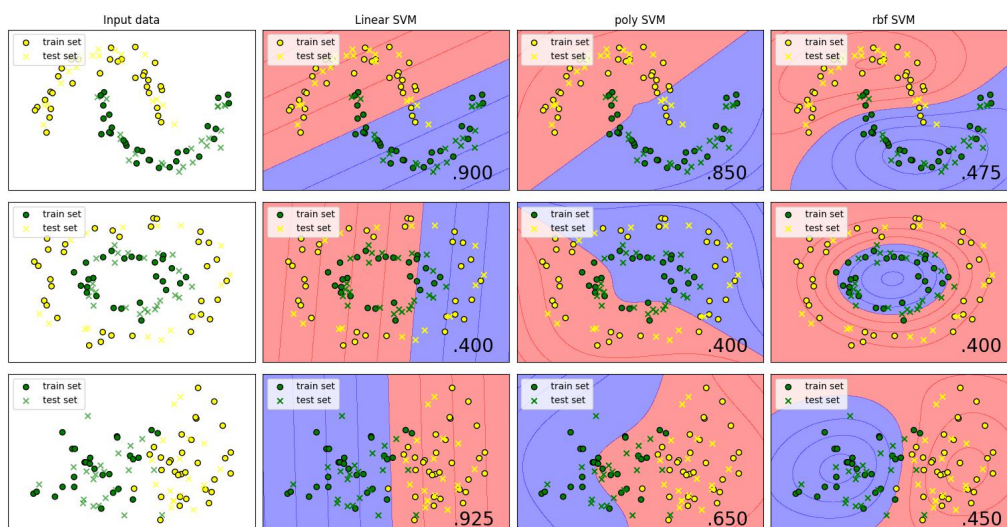
poly : 多项式核，会产生多项式分类边界。

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。

```

15 #设置显示结果图的大小
16 names = ["Linear SVM", "poly SVM", "rbf SVM"]
17 #设置分类器，用到线性SVM，决策树，朴素贝叶斯
18 classifiers = [
19     SVC(kernel="linear", C=0.025),
20     SVC(kernel="poly", C=0.025),
21     SVC(kernel="rbf", C=0.025)
22 ]
23 #设置一个用于分类的数据集，这里设置为线性可分的数据集，输入变量设置
24 X, y = make_classification(n_features=2, n_redundant=0, n_informative=
25 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）

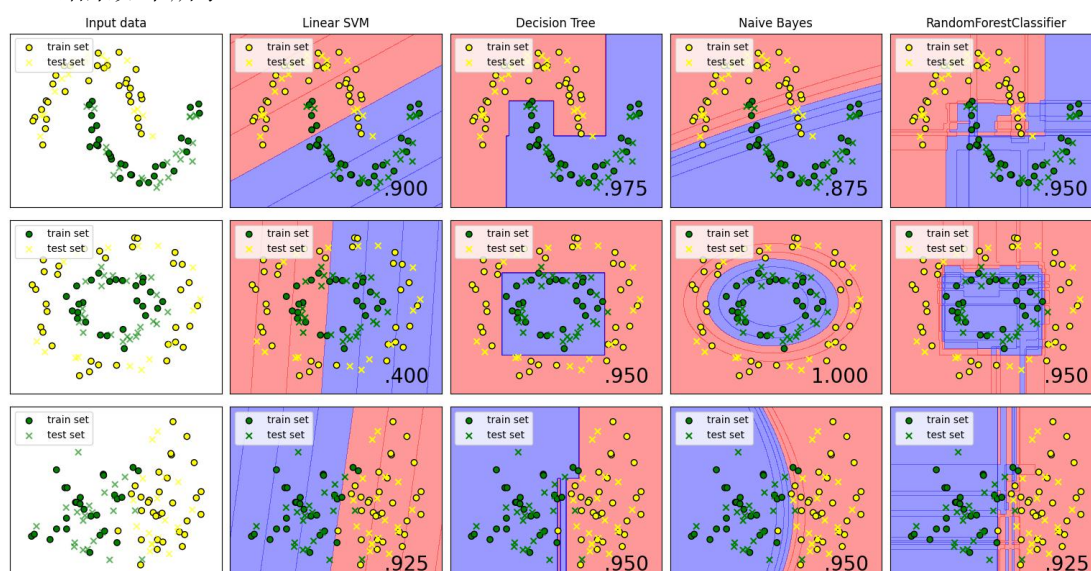
在这一问，我主要对比了随机森林与其它分类方法，

```

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),
21     DecisionTreeClassifier(random_state=0, max_depth=5)

```

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:

```
1.  #导入包
2.  import numpy as np
3.  import matplotlib.pyplot as plt
4.  from matplotlib.colors import ListedColormap
5.  #随机划分训练集、测试集
6.  from sklearn.preprocessing import StandardScaler
7.  #标准化数据数据集
8.  #导入三种分别的数据集，都是具体数据函数生成
9.  from sklearn.datasets import make_moons,make_circles,make_classification
10. from sklearn.model_selection import train_test_split
11. #导入三分类器
12. from sklearn.svm import SVC #支持向量机
13. from sklearn.tree import DecisionTreeClassifier #决策树分类器
14. from sklearn.naive_bayes import GaussianNB #先验为高斯分布的朴素贝叶斯分类器
15. #设置显示结果的图的标题
16. names = ["Linear SVM","Decision Tree","Naive Bayes"]
17. #设置分类器，用到线性SVM，决策树，朴素贝叶斯
18. classifiers = [
19.     SVC(kernel="linear",C=0.025),
20.     DecisionTreeClassifier(random_state=0,max_depth=5),
21.     GaussianNB(priors=[0.5,0.5])
22. ]
23. #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
24. X,y = make_classification(n_features=2,n_redundant=0,n_informative=2,random_state=1,n_clusters_per_class=1)
25. rng = np.random.RandomState(2)#设置一个伪随机种子
26. X+=2*rng.uniform(size=X.shape)#对x变量上加随机扰动
27. 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
28. datasets=[make_moons(noise=0.1,random_state=0),
29.            make_circles(noise=0.1,factor=0.5,random_state=1),
30.            linearly_separable
31.           ]
32. #设置显示结果图的大小
33. figure = plt.figure(figsize=(27,9))
34. i=1
35. #对每个数据集做训练及测试
36. for ds_cnt,ds in enumerate(datasets):
37.     X,y=ds
38.     X=StandardScaler().fit_transform(X)#标准化数据
39.     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.4,random_state=42)#分割数据集
40.     x_min,x_max=X[:,0].min()-0.5,X[:,0].max()+0.5
```

```

41.     y_min,y_max=X[:,1].min()-0.5,X[:,1].max()+0.5
42.     h=0.02#设置网络的步长
43.     xx,yy=np.meshgrid(np.arange(x_min,x_max,h),
44.                         np.arange(y_min,y_max,h)
45.                         )
46.     #先展示输入数据集
47.     cm=ListedColormap(['red','blue'])#设置分割面颜色
48.     cm_bright=ListedColormap(['yellow','green'])#设置散点颜色为黄和绿
49.     ax=plt.subplot(len(datasets),len(classifiers)+1,i)#划分子面
50.     if ds_cnt==0:
51.         ax.set_title("Input data")
52.         #画训练集散点
53.         ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
54.                   edgecolors='k',marker='o',label='train set')
55.         #画测试集散点
56.         ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,alpha=0.6,
57.                   edgecolors='k',marker='x',label='test set')
58.         #画坐标轴
59.         ax.set_xlim(xx.min(),xx.max())
60.         ax.set_ylim(yy.min(),yy.max())
61.         ax.set_xticks(())
62.         ax.set_yticks(())
63.         plt.legend(loc=2)
64.         i+=1
65.         #分别对每个分类器做训练测试
66.         for name,clf in zip(names,classifiers):
67.             ax=plt.subplot(len(datasets),len(classifiers)+1,i)
68.             clf.fit(X_train,y_train)#训练集训练分类器
69.             score=clf.score(X_test,y_test)#测试集测试分类器
70.             if hasattr (clf,"decision_function"):
71.                 Z=clf.decision_function(np.c_[xx.ravel(),yy.ravel()])
72.             else:
73.                 Z=clf.predict_proba(np.c_[xx.ravel(),yy.ravel()])[:,1]
74.             Z=Z.reshape(xx.shape) #分类结果用等高线函数画出
75.             ax.contourf(xx,yy,Z,cmap=cm,alpha=.4)
76.             #画训练集点
77.             ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
78.                       edgecolors='k',marker='o',label='train set')
79.             #画测试集点
80.             ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,
81.                       edgecolors='k',marker='x',label='test set')
82.             #画坐标轴
83.             ax.set_xlim(xx.min(),xx.max())
84.             ax.set_ylim(yy.min(),yy.max())

```

```

85.         ax.set_xticks(())
86.         ax.set_yticks(())
87.         if ds_cnt==0:
88.             #画子图标题
89.             ax.set_title(name)
90.             #显示测试正确率
91.             ax.text(xx.max()-0.3,yy.min()+0.3,('%0.3f'% score).lstrip('0'),
92.                     size=20,horizontalalignment='right')
93.             plt.legend(loc=2)#左上角加图例
94.             i+=1
95. plt.tight_layout()
96. plt.show()

```

创新与拓展(2)源代码

```

1.     #导入包
2.     import numpy as np
3.     import matplotlib.pyplot as plt
4.     from matplotlib.colors import ListedColormap
5.     #随机划分训练集、测试集
6.     from sklearn.preprocessing import StandardScaler
7.     #标准化数据集
8.     #导入三种分别的数据集，都是具体数据函数生成
9.     from sklearn.datasets import make_moons,make_circles,make_classification
10.    from sklearn.model_selection import train_test_split
11.    #导入三分类器
12.    from sklearn.svm import SVC #支持向量机
13.    from sklearn.tree import DecisionTreeClassifier #决策树分类器
14.    from sklearn.naive_bayes import GaussianNB #先验为高斯分布的朴素贝叶斯分类器
15.    #设置显示结果的图的标题
16.    names = ["Linear SVM","poly SVM","rbf SVM"]
17.    #设置分类器，用到线性 SVM，决策树，朴素贝叶斯
18.    classifiers = [
19.        SVC(kernel="linear",C=0.025),
20.        SVC(kernel="poly",C=0.025),
21.        SVC(kernel="rbf",C=0.025)
22.    ]
23.    #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
24.    X,y = make_classification(n_features=2,n_redundant=0,n_informative=2,random_state=1,n_clusters_per_
        _class=1)
25.    rng = np.random.RandomState(2)#设置一个伪随机种子
26.    X+=2*rng.uniform(size=X.shape)#对 x 变量上加随机扰动
27.    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
28.    datasets=[make_moons(noise=0.1,random_state=0),

```

```

29.         make_circles(noise=0.1, factor=0.5, random_state=1),
30.         linearly_separable
31.     ]
32.     #设置显示结果图的大小
33.     figure = plt.figure(figsize=(27,9))
34.     i=1
35.     #对每个数据集做训练及测试
36.     for ds_cnt, ds in enumerate(datasets):
37.         X,y=ds
38.         X=StandardScaler().fit_transform(X)#标准化数据
39.         X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.4,random_state=42)#分割数据集
40.         x_min,x_max=X[:,0].min()-0.5,X[:,0].max()+0.5
41.         y_min,y_max=X[:,1].min()-0.5,X[:,1].max()+0.5
42.         h=.02#设置网络的步长
43.         xx,yy=np.meshgrid(np.arange(x_min,x_max,h),
44.                             np.arange(y_min,y_max,h)
45.                             )
46.         #先展示输入数据集
47.         cm=ListedColormap(['red','blue'])#设置分割面颜色
48.         cm_bright=ListedColormap(['yellow','green'])#设置散点颜色为黄和绿
49.         ax=plt.subplot(len(datasets),len(classifiers)+1,i)#划分子面
50.         if ds_cnt==0:
51.             ax.set_title("Input data")
52.             #画训练集散点
53.             ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
54.                         edgecolors='k',marker='o',label='train set')
55.             #画测试集散点
56.             ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,alpha=0.6,
57.                         edgecolors='k',marker='x',label='test set')
58.             #画坐标轴
59.             ax.set_xlim(xx.min(),xx.max())
60.             ax.set_ylim(yy.min(),yy.max())
61.             ax.set_xticks(())
62.             ax.set_yticks(())
63.             plt.legend(loc=2)
64.             i+=1
65.         #分别对每个分类器做训练测试
66.         for name,clf in zip(names,classifiers):
67.             ax=plt.subplot(len(datasets),len(classifiers)+1,i)
68.             clf.fit(X_train,y_train)#训练集训练分类器
69.             score=clf.score(X_test,y_test)#测试集测试分类器
70.             if hasattr (clf,"decision_function"):
71.                 Z=clf.decision_function(np.c_[xx.ravel(),yy.ravel()])
72.             else:

```



```

73.         Z=clf.predict_proba(np.c_[xx.ravel(),yy.ravel()])[:,1]
74.         Z=Z.reshape(xx.shape) #分类结果用等高线函数画出
75.         ax.contourf(xx,yy,Z,cmap=cm,alpha=.4)
76.         #画训练集点
77.         ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
78.                   edgecolors='k',marker='o',label='train set')
79.         #画测试集点
80.         ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,
81.                   edgecolors='k',marker='x',label='test set')
82.         #画坐标轴
83.         ax.set_xlim(xx.min(),xx.max())
84.         ax.set_ylim(yy.min(),yy.max())
85.         ax.set_xticks(())
86.         ax.set_yticks(())
87.         if ds_cnt==0:
88.             #画子图标题
89.             ax.set_title(name)
90.             #显示测试正确率
91.             ax.text(xx.max()-.3,yy.min()+.3,('%0.3f' % score).lstrip('0'),
92.                   size=20,horizontalalignment='right')
93.             plt.legend(loc=2)#左上角加图例
94.             i+=1
95. plt.tight_layout()
96. plt.show()

```

创新与拓展(3):

```

1.     #导入包
2.     import numpy as np
3.     import matplotlib.pyplot as plt
4.     from matplotlib.colors import ListedColormap
5.     #随机划分训练集、测试集
6.     from sklearn.preprocessing import StandardScaler
7.     #标准化数据集
8.     #导入三种分别的数据集，都是具体数据函数生成
9.     from sklearn.datasets import make_moons,make_circles,make_classification
10.    from sklearn.model_selection import train_test_split
11.    #导入三分类器
12.    from sklearn.svm import SVC #支持向量机
13.    from sklearn.tree import DecisionTreeClassifier #决策树分类器
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),
21.     DecisionTreeClassifier(random_state=0,max_depth=5),
22.     GaussianNB(priors=[0.45,0.55]),
23.     RandomForestClassifier()
24. ]
25. #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
26. X,y = make_classification(n_features=2,n_redundant=0,n_informative=2,random_state=1,n_clusters_per_
    _class=1)
27. rng = np.random.RandomState(2)#设置一个伪随机种子
28. X+=2*rng.uniform(size=X.shape)#对 x 变量上加随机扰动
29. 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
30. datasets=[make_moons(noise=0.1,random_state=0),
31.            make_circles(noise=0.1, factor=0.5, random_state=1),
32.            linearly_separable
33.           ]
34. #设置显示结果图的大小
35. figure = plt.figure(figsize=(27,9))
36. i=1
37. #对每个数据集做训练及测试
38. for ds_cnt,ds in enumerate(datasets):
39.     X,y=ds
40.     X=StandardScaler().fit_transform(X)#标准化数据
41.     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.4,random_state=42)#分割数据集
42.     x_min,x_max=X[:,0].min()-0.5,X[:,0].max()+0.5
43.     y_min,y_max=X[:,1].min()-0.5,X[:,1].max()+0.5
44.     h=.02#设置网络的步长
45.     xx,yy=np.meshgrid(np.arange(x_min,x_max,h),
46.                        np.arange(y_min,y_max,h)
47.                       )
48.     #先展示输入数据集
49.     cm=ListedColormap(['red','blue'])#设置分割面颜色
50.     cm_bright=ListedColormap(['yellow','green'])#设置散点颜色为黄和绿
51.     ax=plt.subplot(len(datasets),len(classifiers)+1,i)#划分子面
52.     if ds_cnt==0:
53.         ax.set_title("Input data")
54.         #画训练集散点
55.         ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
56.                   edgecolors='k',marker='o',label='train set')
57.         #画测试集散点
58.         ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,alpha=0.6,
59.                   edgecolors='k',marker='x',label='test set')
60.         #画坐标轴

```

```

61.     ax.set_xlim(xx.min(),xx.max())
62.     ax.set_ylim(yy.min(),yy.max())
63.     ax.set_xticks(())
64.     ax.set_yticks(())
65.     plt.legend(loc=2)
66.     i+=1
67.     #分别对每个分类器做训练测试
68.     for name,clf in zip(names,classifiers):
69.         ax=plt.subplot(len(datasets),len(classifiers)+1,i)
70.         clf.fit(X_train,y_train)#训练集训练分类器
71.         score=clf.score(X_test,y_test)#测试集测试分类器
72.         if hasattr (clf,"decision_function"):
73.             Z=clf.decision_function(np.c_[xx.ravel(),yy.ravel()])
74.         else:
75.             Z=clf.predict_proba(np.c_[xx.ravel(),yy.ravel()])[:,1]
76.             Z=Z.reshape(xx.shape) #分类结果用等高线函数画出
77.             ax.contourf(xx,yy,Z,cmap=cm,alpha=.4)
78.             #画训练集点
79.             ax.scatter(X_train[:,0],X_train[:,1],c=y_train,cmap=cm_bright,
80.                        edgecolors='k',marker='o',label='train set')
81.             #画测试集点
82.             ax.scatter(X_test[:,0],X_test[:,1],c=y_test,cmap=cm_bright,
83.                        edgecolors='k',marker='x',label='test set')
84.             #画坐标轴
85.             ax.set_xlim(xx.min(),xx.max())
86.             ax.set_ylim(yy.min(),yy.max())
87.             ax.set_xticks(())
88.             ax.set_yticks(())
89.             if ds_cnt==0:
90.                 #画子图标题
91.                 ax.set_title(name)
92.                 #显示测试正确率
93.                 ax.text(xx.max()-.3,yy.min()+.3,('%0.3f' % score).rstrip('0'),
94.                        size=20,horizontalalignment='right')
95.                 plt.legend(loc=2)#左上角加图例
96.                 i+=1
97.     plt.tight_layout()
98.     plt.show()

```