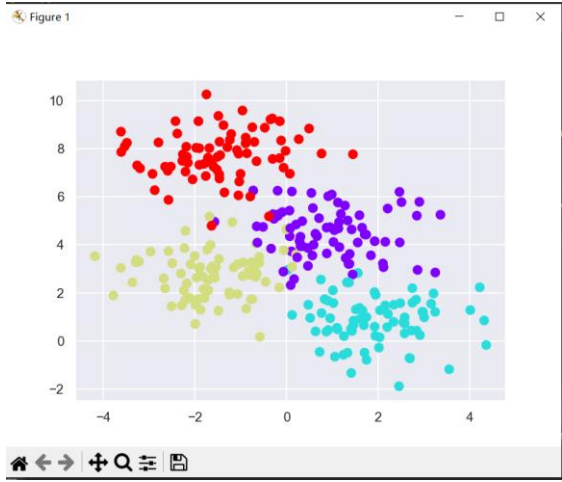


计算机学院实验报告

实验题目： 机器学习实验十一随机森林		学号： 202100130052
日期： 2023. 5. 23	班级： 21 智能	姓名： 刘欣月
Email： 202100130052@mail.sdu.edu.cn		
实验目的： 学习随机森林的非参数算法。		
实验语言介绍： python		
<p>实验步骤：</p> <p>1. 导入标准程序库：</p> <pre>%matplotlib inline import numpy as np import matplotlib.pyplot as plt import seaborn as sns; sns.set()</pre> <p>2. 随机森林的诱因：决策树</p> <p>二叉树分支方法可以非常有效地进行分类：在一棵结构合理的决策树中，每个问题基本上都可将种类可能性减半。</p> <p>决策树的难点在于如何设计每一步的问题。</p> <p>3. 构建决策树</p> <p>看看下面的二维数据，它一共有四种标签</p> <pre>from sklearn.datasets import make_blobs X, y = make_blobs(n_samples=300, centers=4, random_state=0, cluster_std=1.0) plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='rainbow');</pre> 		

在这组数据上构建的简单决策树不断将数据的一个特征或另一个特征按照某种判定条件进行分割。每分割一次，都将新区域内点的多数投票结果标签分配到该区域上。

如果想在 Scikit-Learn 中使用决策树拟合数据，可以用 `DecisionTreeClassifier` 评估器：

快速写一个辅助函数，对分类器的结果进行可视化：

```
def visualize_tree(estimator, X, y, boundaries=True,
                  xlim=None, ylim=None, ax=None):
    ax = ax or plt.gca()

    # Plot the training points
    ax.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap='viridis',
              clim=(y.min(), y.max()), zorder=3)

    ax.axis('tight')
    ax.axis('off')

    if xlim is None:
        xlim = ax.get_xlim()
    if ylim is None:
        ylim = ax.get_ylim()

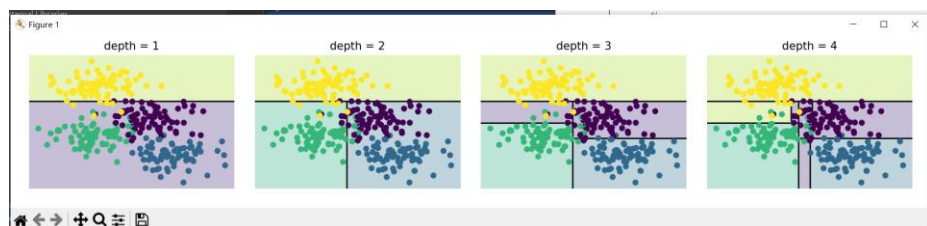
    # fit the estimator
    estimator.fit(X, y)

    xx, yy = np.meshgrid(np.linspace(*xlim, num=200),
                        np.linspace(*ylim, num=200))
    Z = estimator.predict(np.c_[xx.ravel(), yy.ravel()])

    # Put the result into a color plot
    n_classes = len(np.unique(y))
    Z = Z.reshape(xx.shape)
    contours = ax.contourf(xx, yy, Z, alpha=0.3,
                          levels=np.arange(n_classes + 1) - 0.5,
                          cmap='viridis', clim=(y.min(), y.max()),
                          zorder=1)

    ax.set(xlim=xlim, ylim=ylim)
```

下图展示了决策树对这组数据前四次分割的可视化结果：



如果想在 Scikit-Learn 中使用决策树拟合数据，可以用 `DecisionTreeClassifier` 评估器：

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier().fit(X, y)
```

快速写一个辅助函数，对分类器的结果进行可视化：

```
def visualize_classifier(model, X, y, ax=None, cmap='rainbow'):
    ax = ax or plt.gca()

    # Plot the training points
    ax.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=cmap,
               clim=(y.min(), y.max()), zorder=3)
    ax.axis('tight')
    ax.axis('off')
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

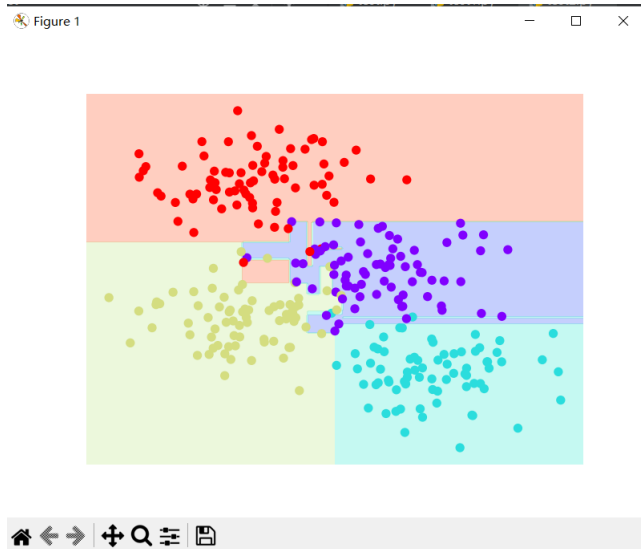
    # fit the estimator
    model.fit(X, y)
    xx, yy = np.meshgrid(np.linspace(*xlim, num=200),
                          np.linspace(*ylim, num=200))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

    # Create a color plot with the results
    n_classes = len(np.unique(y))
    contours = ax.contourf(xx, yy, Z, alpha=0.3,
                           levels=np.arange(n_classes + 1) - 0.5,
                           cmap=cmap, clim=(y.min(), y.max()),
                           zorder=1)

    ax.set(xlim=xlim, ylim=ylim)
```

现在就可以检查决策树分类的结果了

```
visualize_classifier(DecisionTreeClassifier(), X, y)
```

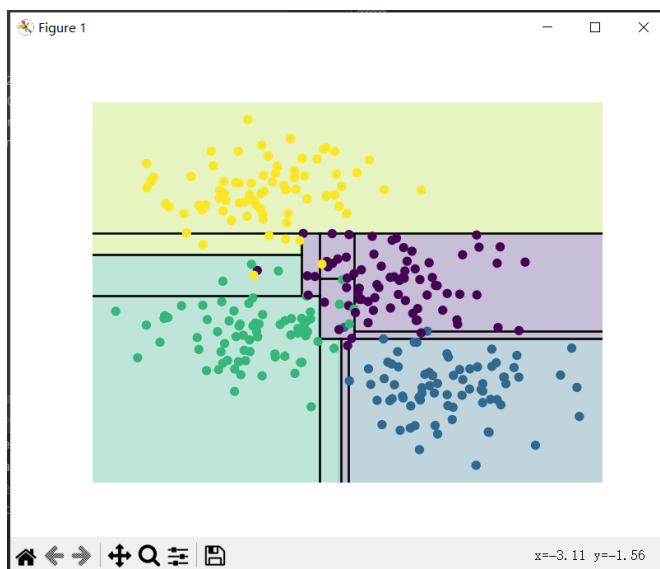


生成决策树创建过程的交互式可视化

helpers_05_08 is found in the online appendix

```
import helpers_05_08
```

```
plot_tree_interactive(X, y);
```



4. 决策树和过拟合

在图中我们训练了两棵不同的决策树，每棵树拟合一半数据。

```
model = DecisionTreeClassifier()
```

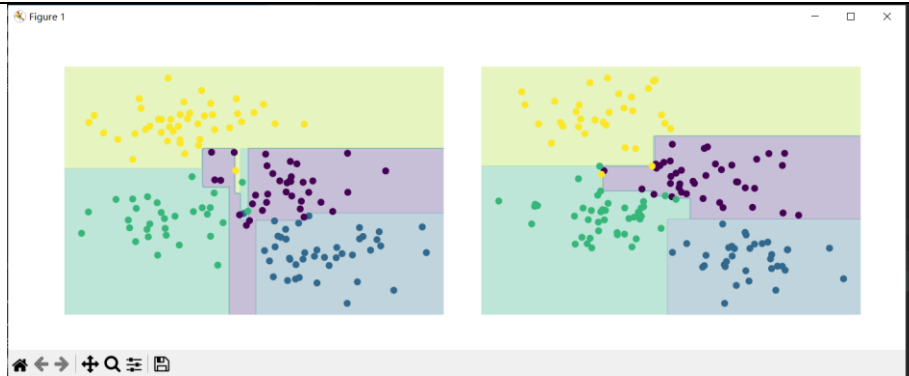
```
fig, ax = plt.subplots(1, 2, figsize=(16, 6))
```

```
fig.subplots_adjust(left=0.0625, right=0.95, wspace=0.1)
```

```
visualize_tree(model, X[:,2], y[:,2], boundaries=False, ax=ax[0])
```

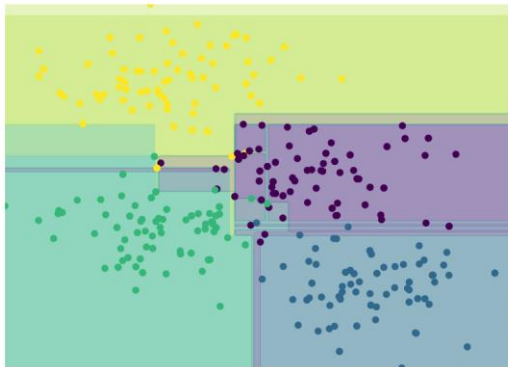
```
visualize_tree(model, X[1::2], y[1::2], boundaries=False, ax=ax[1])
```

结果如下所示：在一些区域，两棵树产生了一致的结果（例如4个角上）；而在另一些区域，两棵树的分类结果差异很大（例如两类接壤的区域）



就像用两棵决策树的信息改善分类结果一样，我们可以用更多决策树的信息来改善分类结果。

```
randomized_tree_interactive(X, y)
```



5. 评估器集成算法：随机森林

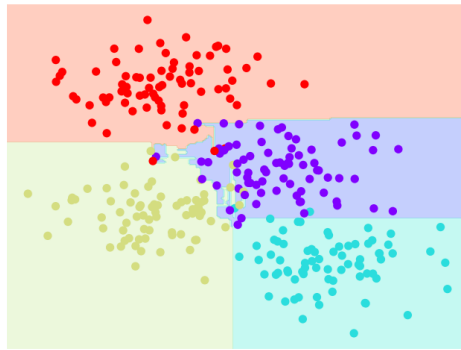
我们可以用Scikit-Learn 的BaggingClassifier 元评估器来实现这种装袋分类器

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier

tree = DecisionTreeClassifier()
bag = BaggingClassifier(tree, n_estimators=100, max_samples=0.8,
                        random_state=1)

bag.fit(X, y)
visualize_classifier(bag, X, y)
```

Figure 1



在Scikit-Learn 里对随机决策树集成算法的优化是通过 RandomForestClassifier 评估器实现的，它会自动进行随机化决策。你只要选择一组评估器，它们就可以非常快速地完成（如果需要可以并行计算）每棵树的拟合任务

```
from sklearn.ensemble import RandomForestClassifier
```

```
model = RandomForestClassifier(n_estimators=100, random_state=0)
```

```
visualize_classifier(model, X, y);
```

Figure 1

