

scikit-learn 实战之监督学习 (/courses/866)

一、实验介绍

1.1 实验内容

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

2.1 K 近邻

2.2 决策树和随机森林

三、常见监督学习方法对比

五、实验总结

六、课后习题

课程名称：scikit-learn 实战之监督学习

一、实验介绍

1.1 实验内容

监督学习（英语：Supervised learning）是机器学习中最为常见、应用最为广泛的分支之一。本次实验将带你了解监督学习中常见的分类方法，并学会使用 scikit-learn 来构建预测模型，用于解决实际问题。

最有效的
学习方式



1.2 实验知识点

- 了解常见的监督学习方法
- 使用多个方法进行分类预测比较

1.3 实验环境

- python3

1.4 适合人群

本课程难度为一般，属于初级级别课程，适合具有 Python 基础和线性代数基础，并对机器学习中分类问题感兴趣的用户。

二、其他监督学习方法介绍

scikit-learn 实战之监督学习 ([courses/866](#))

一、实验介绍

1.1 实验内容

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

2.1 K 近邻

2.2 决策树和随机森林

二、常见监督学习方法对比

五、实验总结

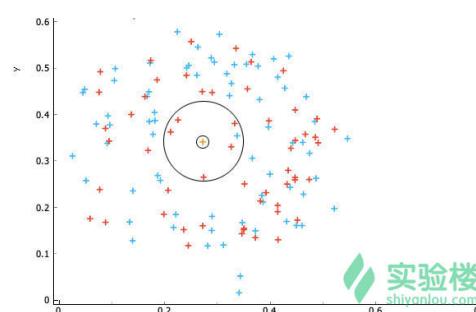
六、课后习题

除了上面两节介绍过的广义线性模型以及支持向量机，还有很多监督学习方法都非常流行。例如：K 近邻、决策树、随机森林、朴素贝叶斯等。

2.1 K 近邻

K 近邻是一种十分常用的监督学习算法。简单来讲，K 近邻就是假设一个给定的数据集，且数据的类别已经确定。这些数据的特征所构成的特征向量可以映射到对应的特征空间中。现在，假设一个输入实例，我们可以计算该输入和其他特征之间的距离，再通过多数表决的方式，来确定新输入实例的类别，最后完成分类。

其中，K 近邻中的“近邻”代表原有特征空间中与新输入实例距离最近的那些样本。而 K 代表距离最近的 K 个值。所以，对于 K 近邻而言。K 值的大小和距离的度量方式（欧式距离或者曼哈顿距离）是其构成的两个关键因素。



如上图所示，原数据集由红、蓝两类组成。现在，我们新输入一个橙色实例，图中小圆圈表示了该实例对应在特征空间的位置。现在，我们确定 $K = 6$ ，然后可以圈定出距离橙色实例最近的 6 个样本点。其中，红色样本为 5 个，蓝色 1 个。根据多数表决的规则，最终确定新

最有效的
学习方式



输入的橙色样本数据属于红色样本类别。
scikit-learn 实战之监督学习 (/courses/866)

2.2 决策树和随机森林

一、实验介绍

1.1 实验内容

决策树也是一种十分常见的监督学习方法。它是一种特殊的树形结构，一般由节点和有向边组成。其中，节点表示特征、属性或者一个类。而有向边包含有判断条件。

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

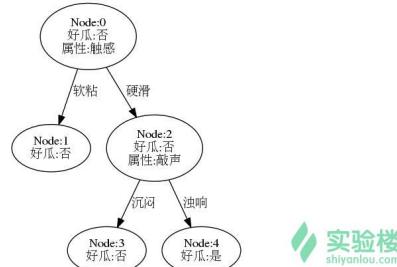
2.1 K 近邻

2.2 决策树和随机森林

三、常见监督学习方法对比

五、实验总结

六、课后习题



最有效的
学习方式



如图所示，决策树从根节点开始延伸，经过不同的判断条件后，到达不同的子节点。而上层子节点又可以作为父节点被进一步划分为下层子节点。一般情况下，我们从根节点输入数据，经过多次判断后，这些数据就会被分为不同的类别。这就构成了一颗简单的分类决策树。

当我们使用决策树分类时，对于已有训练集只建立一颗决策树。而随机森林的概念是，对于一个训练集随机建立多颗决策树。而建立这些决策树时，会采取一种叫 Bootstrap 的取样方式，即每一次从数据集中又放回的取出一部分数据，再用这部分数据去建立小决策树。对于随机森林而言，最终的分类结果由众多小决策树输出类别的众数确定。

由于随机森林的特点，有效地降低过拟合程度，具有较好的泛化误差。另外，训练速度也非常快，模型的表现往往都比较好，是十分受欢迎的一种机器学习方法。

监督学习包含的方法众多，在此就不再一一介

④ scikit-learn 实战之监督学习 ([/courses/866/](#)) 中一些常用监督学习方法的分类效果如何。

一、实验介绍

1.1 实验内容

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

2.1 K 近邻

2.2 决策树和随机森林

二、常见监督学习方法对比

五、实验总结

六、课后习题

二、常见监督学习方法对比

下面，我们通过同一个数据集来对常见的监督学习算法分类性能做一个比较。首先，你需要在终端中键入以下代码获取数据集。

☞ 示例代码：

```
!wget http://labfile.oss.aliyuncs.com/courses/866/class_data.csv
```

最有效的
学习方式



☞ 动手练习：

为了更方便可视化，这里选用了一个随机生成的二分类数据集。总共包含 300 条数据，类别为 0 和 1。

接下来，你可以通过 pandas 读取数据集并预览这些数据。

☞ 示例代码：

```
import pandas as pd # 加载 pandas 模块

data = pd.read_csv("class_data.csv",
header=0) # 读取 csv 文件，并将第一行设为表头

data.head() #输出数据预览
```

☞ 动手练习：

☞ 参考结果：

	X	Y	CLASS

⌚ scikit-learn 实战之监督学习 (/courses/866)

	X	Y	CLASS
0	1.747635	0.768889	1
1	0.842727	1.603604	1
2	1.721907	-1.237197	1
3	-3.040515	1.519587	0
4	1.969260	-2.687745	1

一、实验介绍

1.1 实验内容

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

2.1 K 近邻

2.2 决策树和随机森林

二、常见监督学习方法对比

五、实验总结

六、课后习题

通常情况下，可视化是直观认识陌生数据的很好方法。这里，我们通过 matplotlib 来可视化这些数据。只需要 3 行代码，就可以画出数据集的散点图。

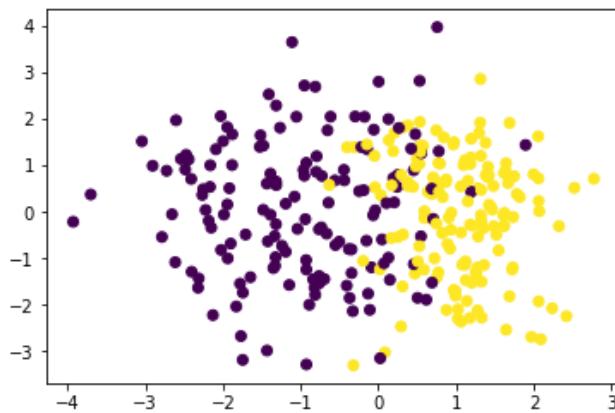
⌚ 示例代码：

```
from matplotlib import pyplot as plt
# 加载绘图模块
%matplotlib inline

plt.scatter(data["X"], data['Y'], c=
data['CLASS']) # 绘制散点图
plt.show() #显示图
```

⌚ 动手练习：

⌚ 参考结果：



上面，我们用 `c=data['CLASS']` 参数来控制散点的颜色。

最有效的
学习方式



第一步，加载本次实验需要的模块，以及
scikit-learn 实战之监督学习 (/courses/866)
scikit-learn 中常见的分类器。

☞ **示例代码：**

一、实验介绍

1.1 实验内容

1.2 实验知识 点

1.3 实验环境

1.4 适合人群

二、其他监督 学习方法介绍

2.1 K 近邻

2.2 决策树和 随机森林

二、常见监督 学习方法对比

五、实验总结

六、课后习题

最有效的
学习方式



④ scikit-learn 实战之监督学习 (course/866)

一、实验介绍

1.1 实验内容

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

2.1 K 近邻

2.2 决策树和随机森林

三、常见监督学习方法对比

五、实验总结

六、课后习题

```
import numpy as np # 导入数值计算模块
from sklearn.model_selection import train_test_split # 导入数据集切分模块
from sklearn.metrics import accuracy_score # 导入准确度评估模块
from matplotlib.colors import ListedColormap # 加载色彩模块

# 集成方法分类器
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier

# 高斯过程分类器
from sklearn.gaussian_process import GaussianProcessClassifier

# 广义线性分类器
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.linear_model import RidgeClassifier
from sklearn.linear_model import SGDClassifier

# K近邻分类器
from sklearn.neighbors import KNeighborsClassifier

#朴素贝叶斯分类器
from sklearn.naive_bayes import GaussianNB

# 神经网络分类器
from sklearn.neural_network import MLPClassifier

# 决策树分类器
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import ExtraTreeClassifier
```

最有效的
学习方式



scikit-learn 实战之监督学习 (courses/866) `svm import SVC
from sklearn.svm import LinearSVC`

一、实验介绍

1.1 实验内容

接下来，建立预测模型，采用预设参数即可。

1.2 实验知识点

由于方法较多，所有这里就不再单独新建模型，而是用列表形式管理。

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

2.1 K 近邻

2.2 决策树和随机森林

三、常见监督学习方法对比

五、实验总结

六、课后习题

支持向量机分类器

```
(courses/866) svm import SVC  
from sklearn.svm import LinearSVC
```

动手练习：

接下来，建立预测模型，采用预设参数即可。

由于方法较多，所有这里就不再单独新建模型，而是用列表形式管理。

示例代码：

```
# 建立模型  
model = [  
    AdaBoostClassifier(),  
    BaggingClassifier(),  
    ExtraTreesClassifier(),  
    GradientBoostingClassifier(),  
    RandomForestClassifier(),  
    GaussianProcessClassifier(),  
    PassiveAggressiveClassifier(),  
    RidgeClassifier(),  
    SGDClassifier(),  
    KNeighborsClassifier(),  
    GaussianNB(),  
    MLPClassifier(),  
    DecisionTreeClassifier(),  
    ExtraTreeClassifier(),  
    SVC(),  
    LinearSVC()  
]  
  
# 依次为模型命名  
classifier_Names = ['AdaBoost', 'Bag  
ging', 'ExtraTrees', 'GradientBoosti  
ng', 'RandomForest', 'GaussianProcess  
, 'PassiveAggressive', 'Ridge', 'SG  
D', 'KNeighbors', 'GaussianNB', 'MLP  
, 'DecisionTree', 'ExtraTree', 'SVC  
, 'LinearSVC']
```

最有效的
学习方式



动手练习：

然后，读取数据集并切分。70% 用于训练，另外 30% 用于测试。

☞ 示例代码： scikit-learn 实战之监督学习 (/courses/866)

一、实验介绍

1.1 实验内容

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

2.1 K 近邻

2.2 决策树和随机森林

二、常见监督学习方法对比

五、实验总结

六、课后习题

```
# 读取数据并切分  
data = pd.read_csv("class_data.csv",  
header=0)  
  
feature = data[['X', 'Y']] # 指定特征变量  
target = data['CLASS'] # 指定分类变量  
X_train, X_test, y_train, y_test = train_test_split(feature, target, test_size=.3) # 切分数据集
```

☞ 动手练习：

准备好数据之后，就可以开始模型训练和测试了。

☞ 示例代码：

```
# 迭代模型  
for name, clf in zip(classifier_Names, models):  
  
    clf.fit(X_train, y_train) # 训练模型  
    pre_labels = clf.predict(X_test)  
    # 模型预测  
    score = accuracy_score(y_test, pre_labels) # 计算预测准确度  
  
    print('%s: %.2f' % (name, score))  
    # 输出模型准确度
```

☞ 动手练习：

执行过程中出现一些警告无需担心，因为 scikit-learn 版本变动比较快。我们可以看到，这 16 个分类器最终的准确度均在 80% ~ 90% 之间，差距不是很大。对于这种现象，主要有两个原因。首先，本次使用的是一个非常规范整洁的线性分类数据集。其次，所有的分类器均采用了默认参数。

最有效的
学习方式



④ scikit-learn 实战之监督学习 (/courses/866)
接下来，我们通过可视化的方法将每一个模型
在分类时的决策边界展示出来，这样能更加直
观的感受到机器学习模型在执行分类预测时发
生的变化。

一、实验介绍

1.1 实验内容

继续编写代码。书写代码时，一定要注意格式
缩进，以免运行时报错。

1.2 实验知识 点

☞ **示例代码：**

1.3 实验环境

1.4 适合人群

二、其他监督 学习方法介绍

2.1 K 近邻

最有效的
学习方式



2.2 决策树和 随机森林



二、常见监督 学习方法对比

五、实验总结

六、课后习题

④ scikit-learn 实战之监督学习 (#绘制数据集)

```
i = 1 # 为绘制子图设置的初始编号参数
cm = plt.cm.Reds # 为绘制热力图选择的样式
cm_color = ListedColormap(['red', 'yellow']) # 为绘制训练集和测试集选择的样式
```

一、实验介绍

1.1 实验内容

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

2.1 K 近邻

2.2 决策树和随机森林

三、常见监督学习方法对比

五、实验总结

六、课后习题

```
# 栅格化
x_min, x_max = data['X'].min() - .5,
data['X'].max() + .5
y_min, y_max = data['Y'].min() - .5,
data['Y'].max() + .5
```

```
xx, yy = np.meshgrid(np.arange(x_min,
, x_max, .1),
np.arange(y_min,
, y_max, .1))
```

```
# 模型迭代
plt.figure(figsize=(20,10))
```

```
for name, clf in zip(classifier_Names, models):
    ax = plt.subplot(4, 4, i) # 绘制 4x4 子图
```

```
clf.fit(X_train, y_train) # 模型训练
```

```
pre_labels = clf.predict(X_test)
```

```
# 模型测试
score = accuracy_score(y_test, pre_labels) # 模型准确度
```

```
# 决策边界判断
if hasattr(clf, "decision_function"):
```

```
    Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
    print("decision_function: ", clf)
else:
    Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
    print("predict_proba: ", clf)
```

```
# 绘制决策边界热力图
```

```
Z = Z.reshape(xx.shape)
ax.contourf(xx, yy, Z, cmap=cm, alpha=.6)
```

最有效的
学习方式



⌚ scikit-learn 实战之监督学习 (/courses/866)

一、实验介绍

1.1 实验内容

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

2.1 K 近邻

2.2 决策树和随机森林

二、常见监督学习方法对比

五、实验总结

六、课后习题

```
#绘制训练集和测试集
ax.scatter(X_train['X'], X_train['Y'], c=y_train, cmap=cm_color)
ax.scatter(X_test['X'], X_test['Y'], c=y_test, cmap=cm_color, edgecolors='black')
```

#图形样式设定

```
ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())
ax.set_title('%s | %.2f' % (name, score))
```

i += 1

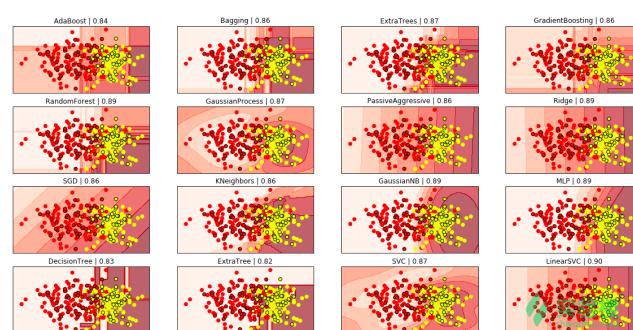
plt.show() #显示图

最有效的
学习方式



👉 动手练习：

按照上面的步骤执行，你就能看到如下图所示的对比图了。



上面将决策边界绘制出来，并用热力图显示。其中，颜色越深表示偏向于黄色散点分类的概率越高，而颜色越浅，则表示偏向红色散点的概率越高。

为了进一步探索各分类器对于不同特征分布的数据集的适用情况。接下来，我们将原有数据集做一些变换。

在前几节的课程中，我们都知道了 `sklearn.datasets` 这个模块可以导入一些

预设的数据集。其实，不仅如此，这个模块开
scikit-learn 实战之监督学习(/courses/866)提供了~~一些~~数据集的生成方法。比如：

一、实验介绍

1.1 实验内容

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

2.1 K 近邻

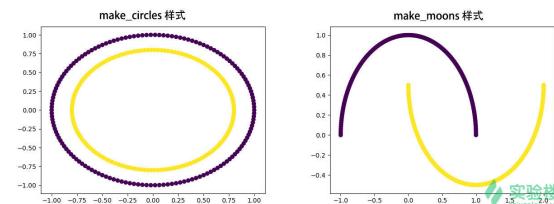
2.2 决策树和随机森林

二、常见监督学习方法对比

五、实验总结

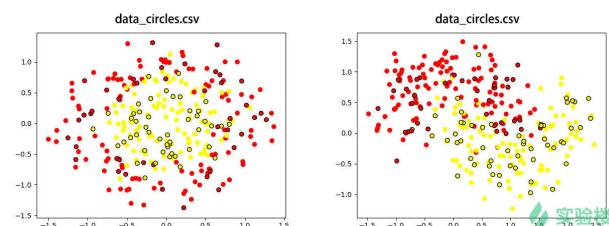
六、课后习题

- `sklearn.datasets.make_circles` 方法可以生成大圆环包小圆环样式的数据集。
- `sklearn.datasets.make_moons` 方法可以生成两个交织间隔圆环样式的数据集。
- +



本次试验中，已经为拟提供了两个生成好的数据集，并添加了噪声。拟可以先通过以下代码获取。当然你可以自行尝试通过上面的两个方法生成数据。

```
# 获取 circles 数据集
!wget http://labfile.oss.aliyuncs.co
m/courses/866/data_circles.csv
# 获取 moons 数据集
!wget http://labfile.oss.aliyuncs.co
m/courses/866/data_moons.csv
```



这两个数据集的结构和一开始的 `class_data.csv` 数据集一致，也就是说，你只需要更改导入数据的那一行代码即可重复上面的流程。

下面是这两个新数据集的预测结果。现在，线

最有效的
学习方式



性分类器和非线性分类器的差别就很明显了。 scikit-learn 实战之监督学习 (/courses/866)

一、实验介绍

1.1 实验内容

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

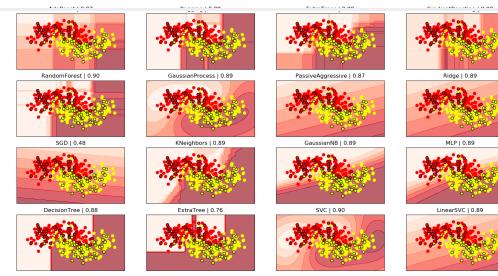
2.1 K 近邻

2.2 决策树和随机森林

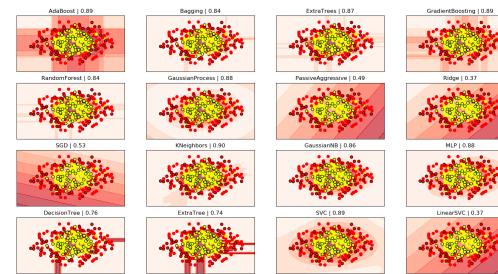
二、常见监督学习方法对比

五、实验总结

六、课后习题



实验楼



实验楼

最有效的
学习方式



五、实验总结

本次实验对比了 scikit-learn 中常见的监督学习方法，我们可以看出不同方法之间差别。另外，对于不同空间分布的数据集，模型的适用性也不一样。大多数非线性分类模型的表现和适用性都较好。

六、课后习题

1. 自己尝试通过

```
sklearn.datasets.make_multilabel_classification
```

方法生成一个多类别的数据集，并通过常见的非线性分类器完成分类。

© 本课程内容，由作者授权实验楼发布，未经允许，禁止转载、下载及非法传播。

*本课程内容，由作者授权实验楼发布，未经允许，禁止转载、下载及非法传播。

④ scikit-learn 实战之监督学习模型概览 /866/3158/document)

一、实验介绍

1.1 实验内容

1.2 实验知识点

1.3 实验环境

1.4 适合人群

二、其他监督学习方法介绍

2.1 K 近邻

2.2 决策树和随机森林

二、常见监督学习方法对比

五、实验总结

六、课后习题

最有效的
学习方式

