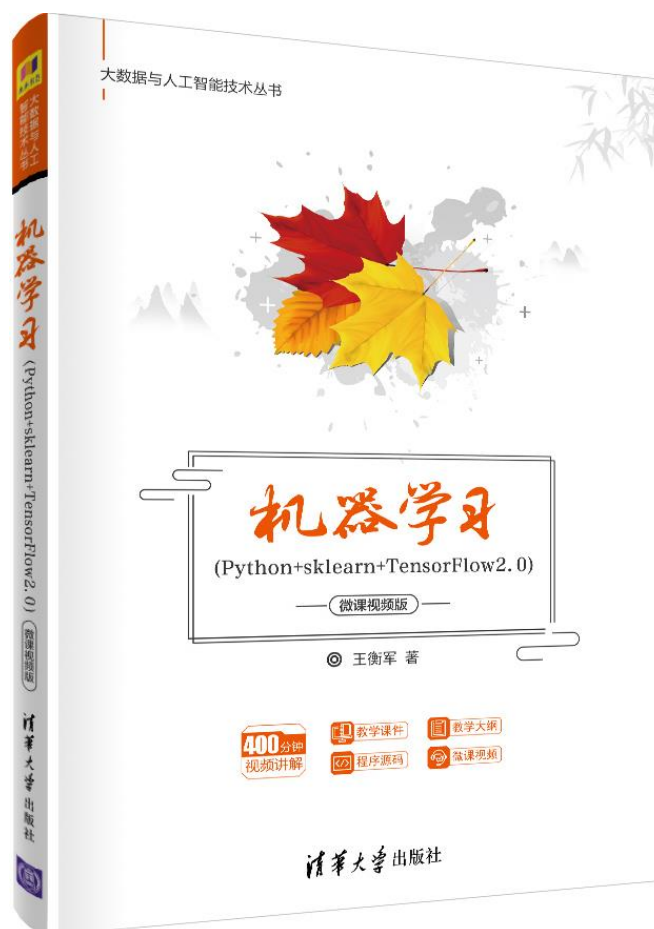


《机器学习》实验参考



实验项目已经全部完成。

各位老师，如果觉得一个实验过大，建议根据教学进度将实验拆开，每一个小的步骤设计成一个实验。

如实验项目 4，可以：

在讲第三章回归时，做第 1 步（样本数据分析和处理）和第 2 步（回归算法建模及分析）实验；

在讲第五章特征工程时，做第 3 步（超参数调优）和第 4 步（特征选择）实验；

在讲第七章神经网络时，做第 5 步（神经网络模型）实验。

这样一件事前后连贯起来做，实验效果可能会更好。

说明：应多位老师的要求，编写了此实验指导书及其参考书，供实践教学环节暂时使用。

本实验参考书内容来自于本人正在编写的新书《机器学习与深度学习》(暂定名)，编写该书的目的之一作为原书的配套用书，因此，它不再以理论推导为主，而是以示例为主要手段来分析、验证各算法，更加适合初学者入门使用。当然，该书内容相对完整，也可单独使用，对数学知识较薄弱的读者或者注重工程应用的读者更加有效。

因应机器学习领域发展现状，该书大量增加了深度学习的内容，包括强化学习和对抗样本等内容。其示例增加了 MindSpore 的版本，MindSpore 是华为公司于 2020 年开源的深度学习框架，目前还在快速发展中，未来可见在我国会有广泛的应用。该书没有直接聚焦于扩展库和深度学习框架本身，而是以基本理论知识为主线来逐步展开，从概念入手逐步讨论算法思想，着重考虑了知识的关联性，最后落实到扩展库和深度学习框架的具体应用上。

在算法分析之后，该书试图从大学生学科竞赛、研究生学术研究和工业界实际应用三个角度提供一些完整的实际案例供读者参考。

实验项目1 环境搭建

1.1.Anaconda 安装

Python 的应用范围非常广泛，包括且不限于科学计算、人工智能、大数据、云计算、网站开发、游戏开发等领域。Python 的强大功能来源于它的数量庞大、功能完善的第三方扩展库（也称为包或者模块，`package`），本书只涉及其中与机器学习相关的扩展库。

Python 的可应用于不同领域业务开发的第三方扩展库来自于不同的机构。它们不仅数量庞大，而且相互之间存在复杂的依赖关系，维护管理起来很麻烦。Anaconda 是一个可对 Python 及其常用扩展库进行下载、安装和自动管理的软件。Anaconda 支持 Windows、Linux 和 macOS 等操作系统。考虑到大部分初学者都使用 Windows 操作系统，因此，本书以 Windows 下的安装步骤为例来介绍 Anaconda 的安装。

1.1.1. 几个概念

对于初学者，在使用 Anaconda 之前，有几个重要的概念需要了解。

1.图形化管理与命令行管理。Anaconda 对包、环境等的管理可以通过图形化界面和命令行界面来进行，分别称为 Navigator（图 1-1）和 Conda（图 1-5）。在管理效果上，两者没有区别。但是，Conda 的操作方式在 Windows、Linux 和 macOS 等操作系统上保持一致，且响应一般要快一些，因此受到大部分使用者的欢迎。本书的相关操作主要用命令行的方式进行介绍。

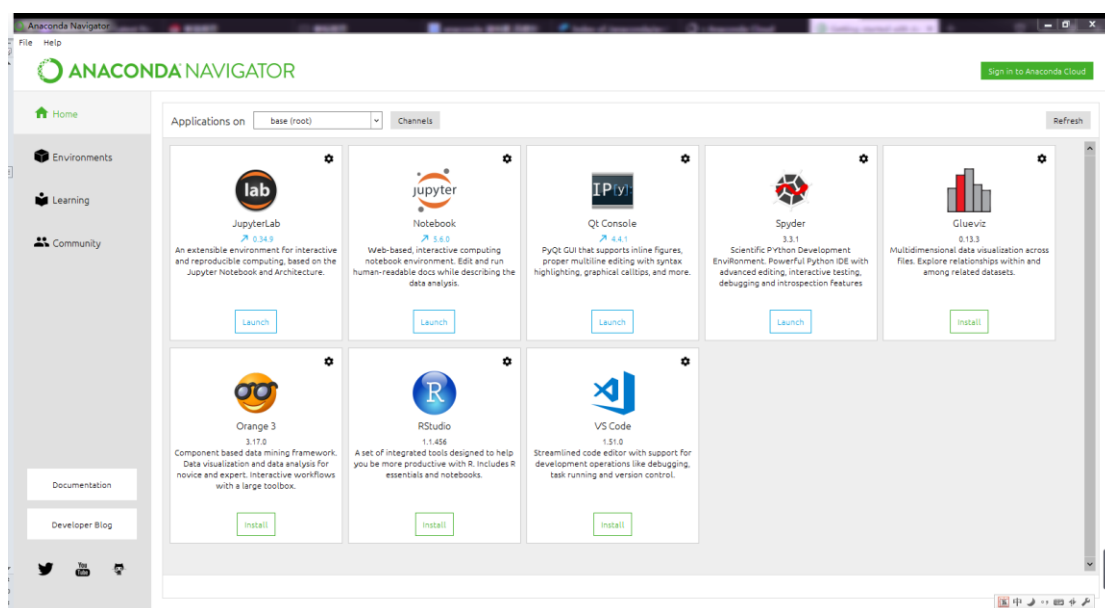


图 1-1 图形化管理界面

2.环境。如前文所述，Python 的扩展库之间存在复杂的依赖关系，一个最终使用的包可能依赖于多个其他包才能正常运行，而其他包又可能需要依赖另外的包。Anaconda 中的环境可用来隔离不同应用开发的包依赖关系。比如，当需要同时开发机器学习应用和网络应用时，可以分别放在两个不同的环境中，以免相互影响而产生意想不到的干扰。环境还可以用来区别不同的版本，比如有的程序要求用 Python3 来开发，而有的程序要求用 Python2 来开发。总之，在 Anaconda 上可以建立多个用于不同开发目的的环境，它们互不干扰。此外，建好的环境还可以整体迁移到其他计算机上，这对于在一些不能联接互联网的计算机上安装开发环境十分有用。

3.下载源。Anaconda 可以对常用扩展库进行自动下载，但它的官方服务器在国外，大部分用户的下载速度很慢。国内清华大学、中国科技大学等机构提供了镜像服务，这些下载源的下载速度相对快的多。实际使用时，下载源一般要更换为国内的镜像服务器。

1.1.2. Anaconda 安装

可到 Anaconda 官方网站¹下载安装包安装。如果官方网站下载速度不稳定，可到清华大学开源软件镜像站²，选择合适版本的安装软件下载。本书选择 Anaconda3-2019.10-Windows-x86_64.exe 文件（64 位 Windows 操作系统版本）。下载后进行安装，初始界面如图 1-2 所示。

¹ <https://www.anaconda.com/products/individual>

² <https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/>

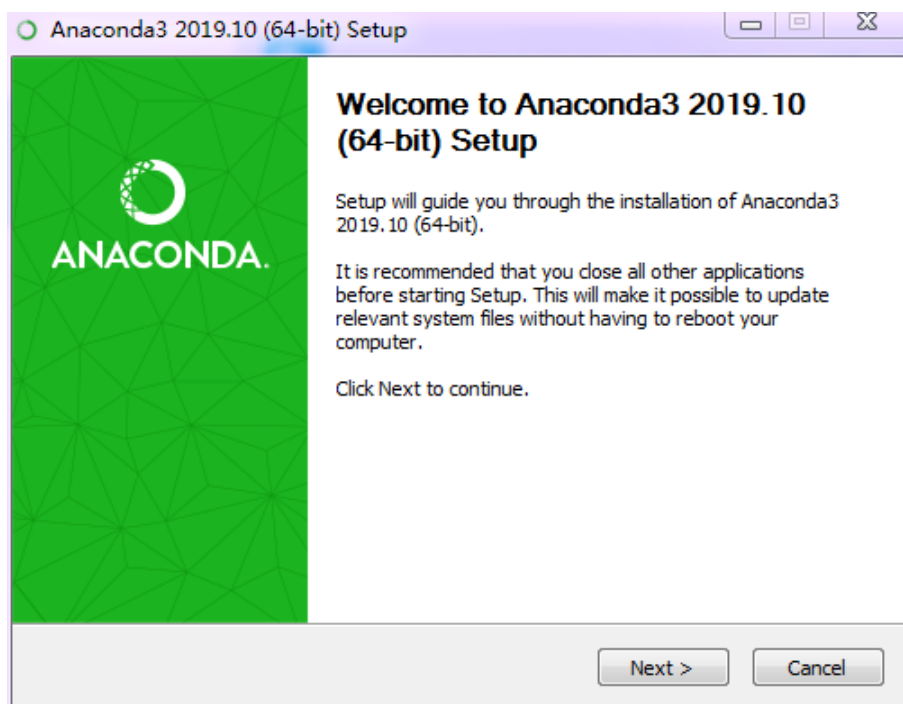


图 1-2 Anaconda 初始安装界面

连续选择“下一步”和“同意”后，来到安装目录界面，界面如图 1-3 所示。本书的实验环境安装到“E:\Anaconda3”文件夹，读者可根据需要确定合适安装目录。

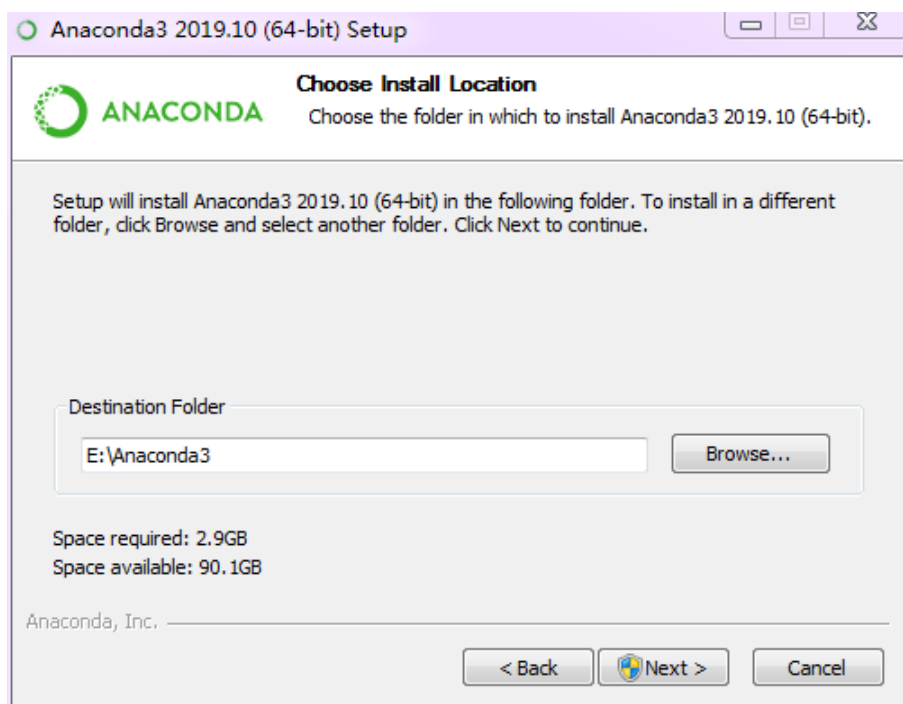


图 1-3 Anaconda 安装目录界面

安装完成后，在“开始”菜单里会出现 Anaconda3 的程序组，如图 1-4 所示。

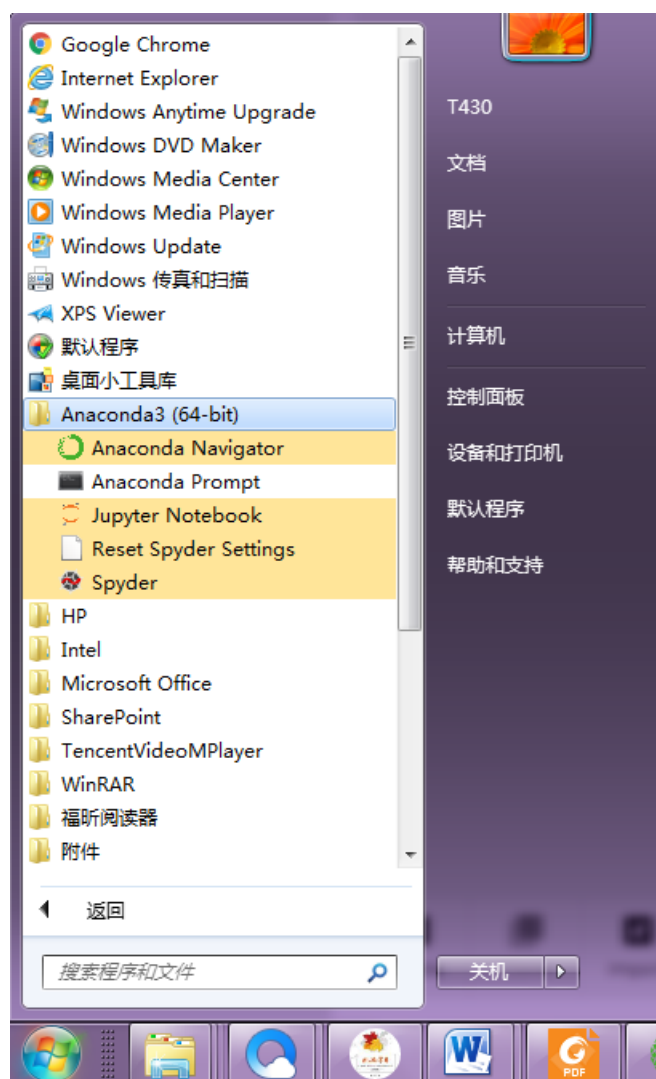
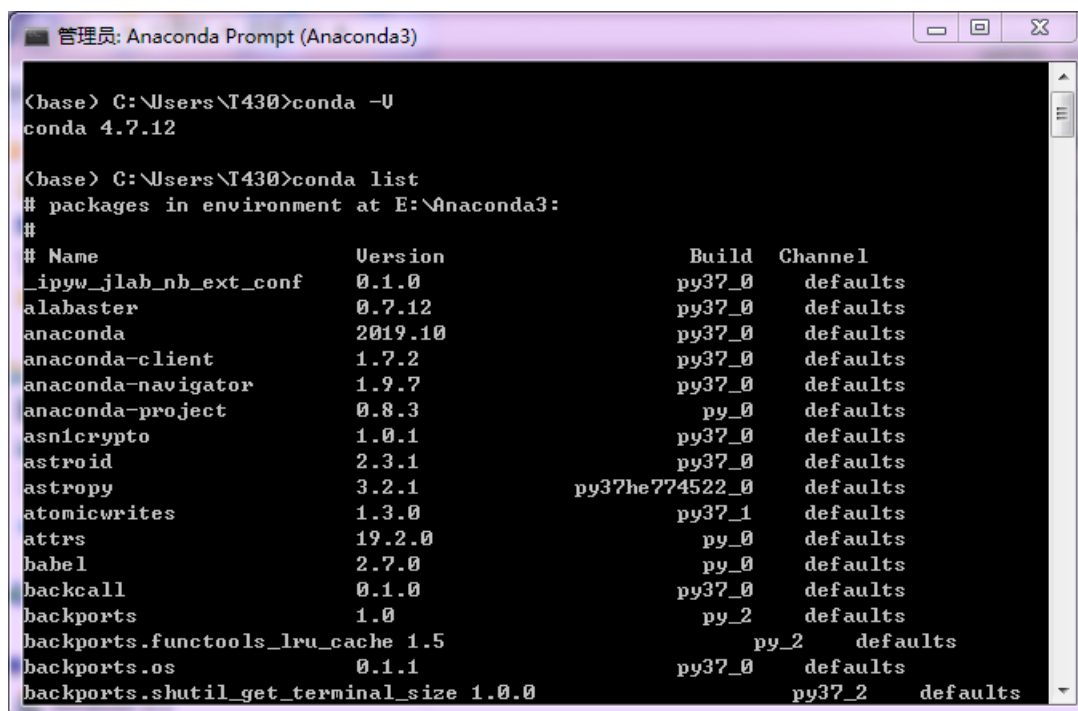


图 1-4 Anaconda3 程序组

在 Anaconda3 程序组里选择点击“Anaconda Navigator”，启动 Navigator 图形化管理界面如图 1-1 所示。

Conda 命令行管理界面通过选择点击“Anaconda Prompt”启动，输入“conda -V”和“conda list”命令，分别查看 Anaconda 的版本号和已经默认安装的包及其版本号，如图 1-5 所示。

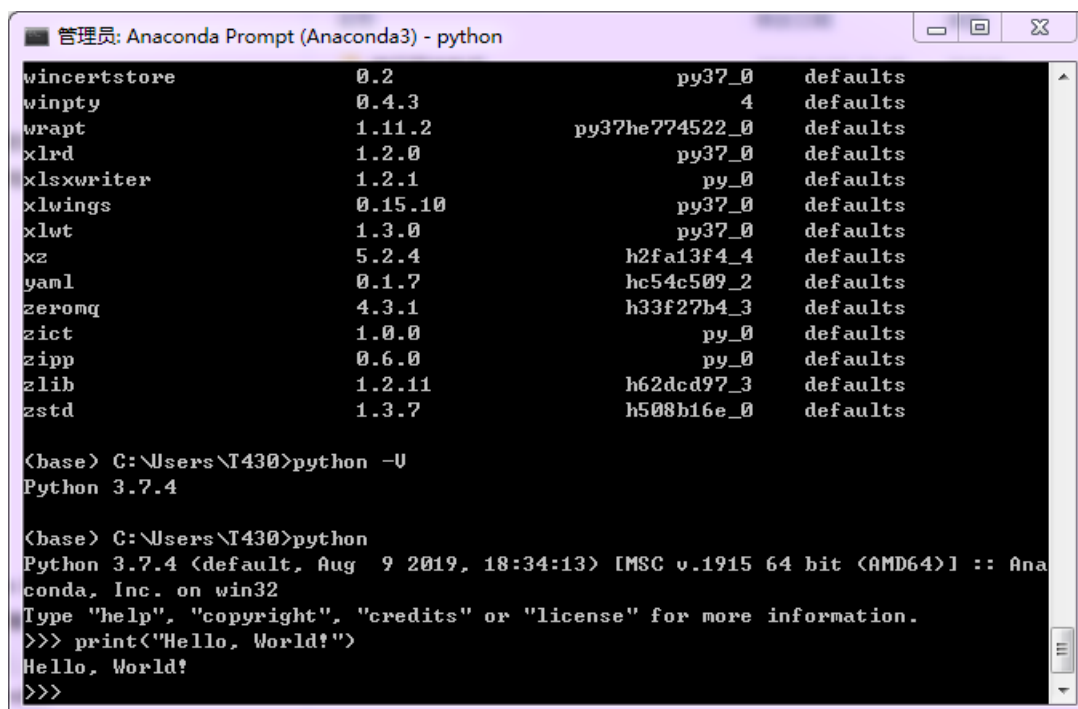


```
(base) C:\Users\T430>conda -U
conda 4.7.12

(base) C:\Users\T430>conda list
# packages in environment at E:\Anaconda3:
#
# Name                                Version                                Build                                Channel
_ipyw_jlab_nb_ext_conf                0.1.0                                py37_0                             defaults
alabaster                              0.7.12                               py37_0                             defaults
anaconda                              2019.10                              py37_0                             defaults
anaconda-client                       1.7.2                                py37_0                             defaults
anaconda-navigator                    1.9.7                                py37_0                             defaults
anaconda-project                      0.8.3                                py_0                                defaults
asn1crypto                            1.0.1                                py37_0                             defaults
astroid                               2.3.1                                py37_0                             defaults
astropy                               3.2.1                                py37he774522_0                     defaults
atomicwrites                          1.3.0                                py37_1                             defaults
attrs                                 19.2.0                               py_0                                defaults
babel                                 2.7.0                                py_0                                defaults
backcall                              0.1.0                                py37_0                             defaults
backports                             1.0                                   py_2                                defaults
backports.functools_lru_cache         1.5                                  py_2                                defaults
backports.os                          0.1.1                                py37_0                             defaults
backports.shutil_get_terminal_size    1.0.0                                py37_2                             defaults
```

图 1-5 命令行管理界面

输入“python -V”可以查看该版本的 Anaconda 默认安装的 python 的版本为 3.7.0。输入“python”，启动交互式编程模式，在第一行指出了 python 的版本。在“>>>”后输入“print(“Hello, World!”)”后回车，将在下一行输出“Hello, World!”。print 语句的作用是输出一段指定的字符串。以上操作见图 1-6。



```
winertstore 0.2 py37_0 defaults
winpty 0.4.3 4 defaults
wrap 1.11.2 py37he774522_0 defaults
xlrd 1.2.0 py37_0 defaults
xlsxwriter 1.2.1 py_0 defaults
xlwings 0.15.10 py37_0 defaults
xlwt 1.3.0 py37_0 defaults
xz 5.2.4 h2fa13f4_4 defaults
yaml 0.1.7 hc54c509_2 defaults
zeromq 4.3.1 h33f27b4_3 defaults
zict 1.0.0 py_0 defaults
zipp 0.6.0 py_0 defaults
zlib 1.2.11 h62dc97_3 defaults
zstd 1.3.7 h508b16e_0 defaults

(base) C:\Users\T430>python -V
Python 3.7.4

(base) C:\Users\T430>python
Python 3.7.4 (default, Aug 9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
>>>
```

图 1-6 命令行下查看 python 版本并输出“Hello, World!”

1.2. 开发环境

本书采用 Anaconda 自带的 Spyder 和 Jupyter Notebook 作为开发环境。Spyder 是图形化集成开发环境，在 Anaconda 程序组里选择点击“Spyder”，或者在 Conda 命令行管理界面里输入“spyder”启动 Spyder，如图 1-7 所示。启动之后，默认会打开一个名为：tmp.py 的临时代码文件。在其中新的一行输入“print(“Hello, World!”)”，点击工具栏中的绿色右三角运行（初次运行时会出现如图 1-8 所示的运行配置窗口，一般选择默认值即可），将在右下的输出窗口中输出“Hello, World!”。

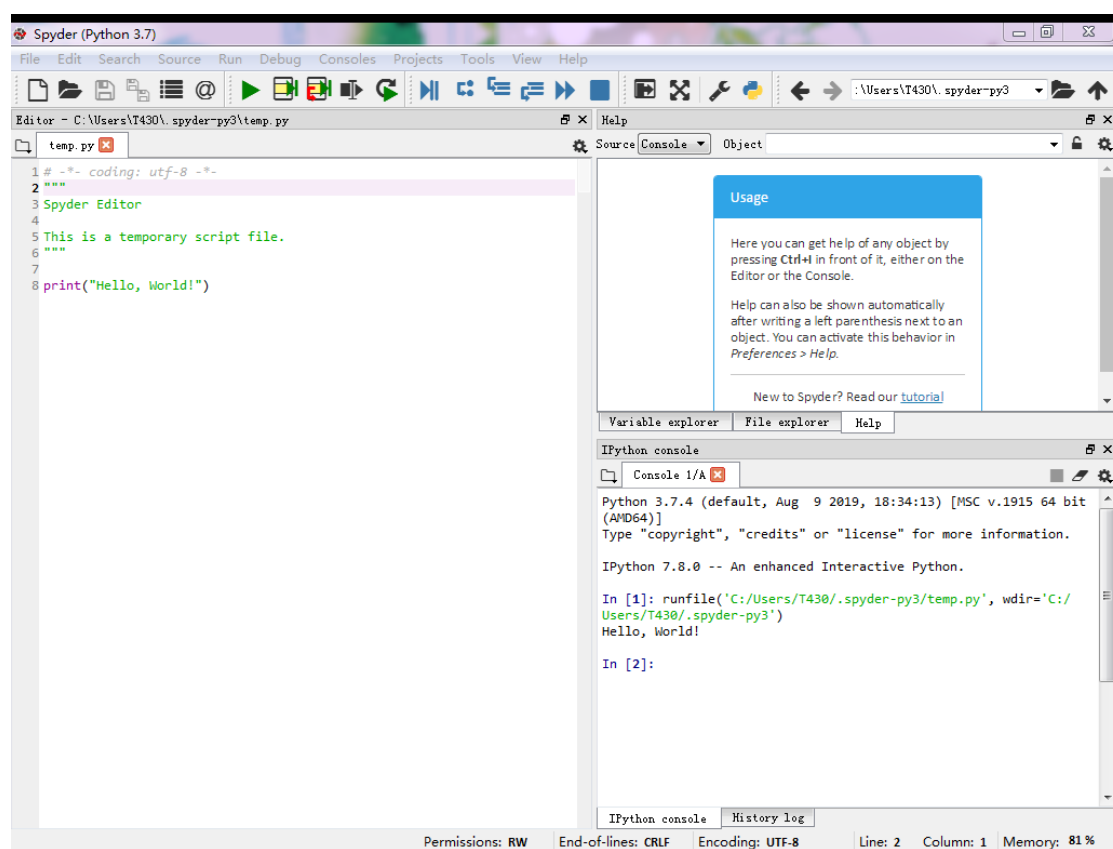


图 1-7 Spyder 集成开发环境并输出“Hello, World!”

在菜单栏 File 项里选择“Save as...”命令，可将文件改名另存到任意文件夹里。

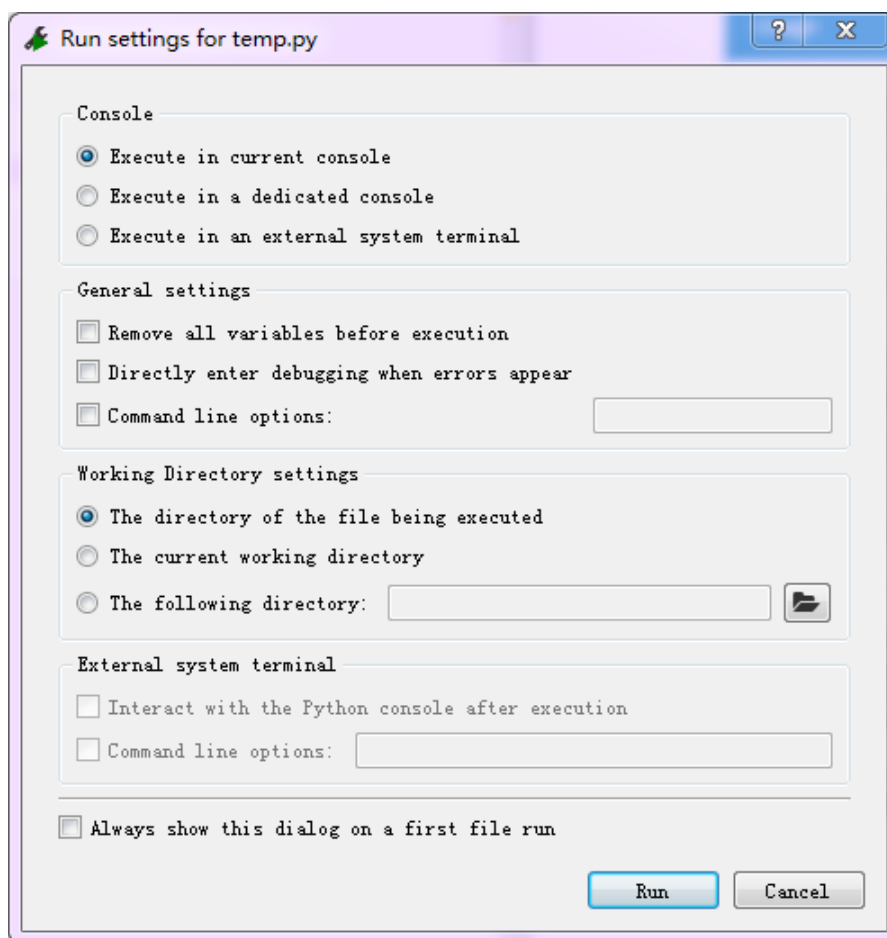
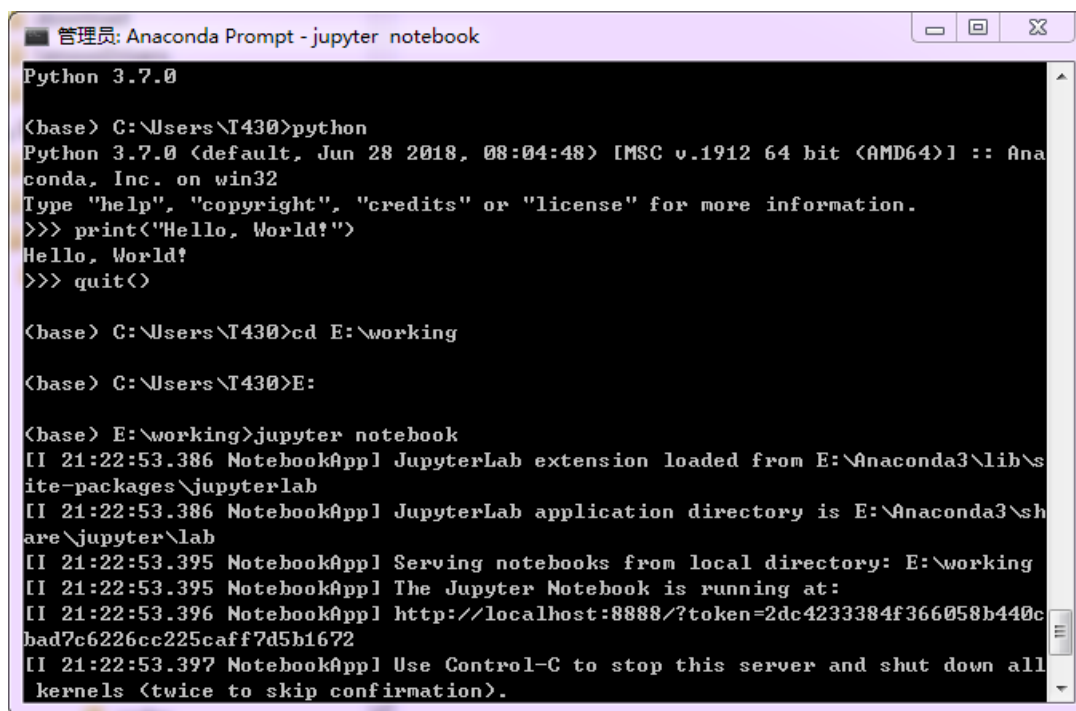


图 1-8 Spyder 环境中初次运行时环境配置界面

Jupyter Notebook（此前被称为 IPython notebook）是一个交互式笔记本，支持运行 40 多种编程语言。它是一个 Web 应用程序，可将代码、文本说明、数学方程、图表等集合在一个文档里，表达能力非常强，便于交流。现在很多资料都采用 Jupyter 格式，成为同行交流的重要工具。



```
管理员: Anaconda Prompt - jupyter notebook
Python 3.7.0

(base) C:\Users\T430>python
Python 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, World!")
Hello, World!
>>> quit()

(base) C:\Users\T430>cd E:\working

(base) C:\Users\T430>E:

(base) E:\working>jupyter notebook
[I 21:22:53.386 NotebookApp] JupyterLab extension loaded from E:\Anaconda3\lib\site-packages\jupyterlab
[I 21:22:53.386 NotebookApp] JupyterLab application directory is E:\Anaconda3\share\jupyter\lab
[I 21:22:53.395 NotebookApp] Serving notebooks from local directory: E:\working
[I 21:22:53.395 NotebookApp] The Jupyter Notebook is running at:
[I 21:22:53.396 NotebookApp] http://localhost:8888/?token=2dc4233384f366058b440cbad7c6226cc225caff7d5b1672
[I 21:22:53.397 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
```

图 1-9 命令行启动 Jupyter Notebook

在 python 交互式编程界面中输入“quit()”退出交互式编程模式。在 Conda 命令行管理界面中，进入到工作目录（本书为 E:\working），输入“jupyter notebook”命令，以上输入见图 1-9，在浏览器中启动 Jupyter Notebook，工作目录初始为空目录，点击右上角的 new 按钮，选择其中的“Python3”建立一个新的代码文件，以上过程见图 1-10。

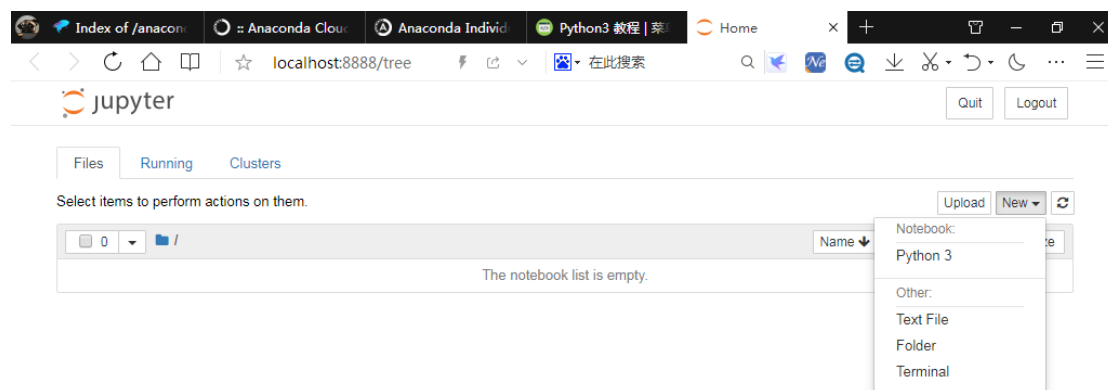


图 1-10 在浏览器中打开 Jupyter Notebook 并新建一个代码文件

在新打开的窗口的第一行输入“print(“Hello, World!”)”，然后点击工具栏中的 Run 按钮或者输入 Shift+Enter 快捷键使程序运行，会在该行下面输出“Hello, World!”，如图 1-11 所示。点击 File 菜单，选择其中的“Rename...”项，可以将保存在工作目录下的文件更名为“hello.ipynb”。可见在 Jupyter 方式里，也可以实行交互行编程，还可以保存运行过程供以

后再次使用。

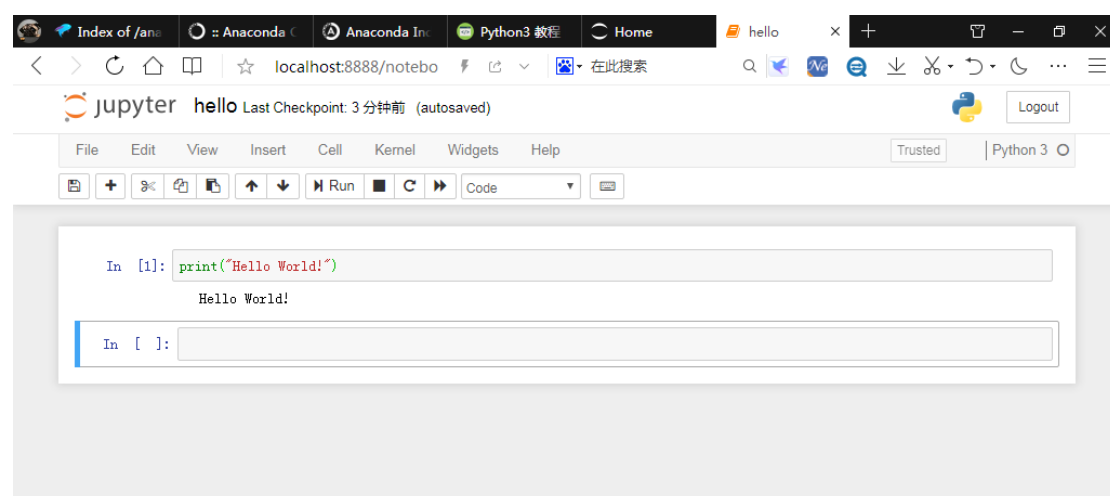


图 1-11 Jupyter 编程环境中的“Hello, World!”程序

本小节和上小节演示了 `python` 的命令行交互式、`spyder` 集成开发环境和 `Jupyter Notebook` 开发环境初始编程。

本书的示例主要采用 `spyder` 集成开发环境和 `Jupyter Notebook` 开发环境来完成。要注意的是，`Jupyter Notebook` 文件中可以嵌入详细的说明，便于交流，而 `spyder` 集成开发环境可以单步跟踪代码运行情况，观察数据的变化情况，有利于理解算法的运行过程。当然两个环境并不矛盾，在一个环境中写的代码很容易移植到另一个环境中运行，读者可以根据需要应用它们。

关于编程环境的问题，读者可以在需要的时候上网查询更复杂的操作，不难掌握。

下面来安装本书要使用的 `sklearn` 机器学习扩展库、`TensorFlow2` 深度学习框架。本书面向初学者，示例一般基于最基本的 `CPU` 平台实现，虽然运行速度较慢，但不影响对原理的理解。因此，这里只介绍安装支持 `CPU` 平台的版本。如果不能一次成功，可到网上查阅解决办法。

1. 先将下载源更换为清华大学镜像站。

在 `Conda` 命令行管理界面中，输入如下四条命令（可到网上搜索相关内容，直接拷贝到 `Conda` 命令行管理界面中执行即可，不必逐字输入）：

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/main/
```

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free/
```

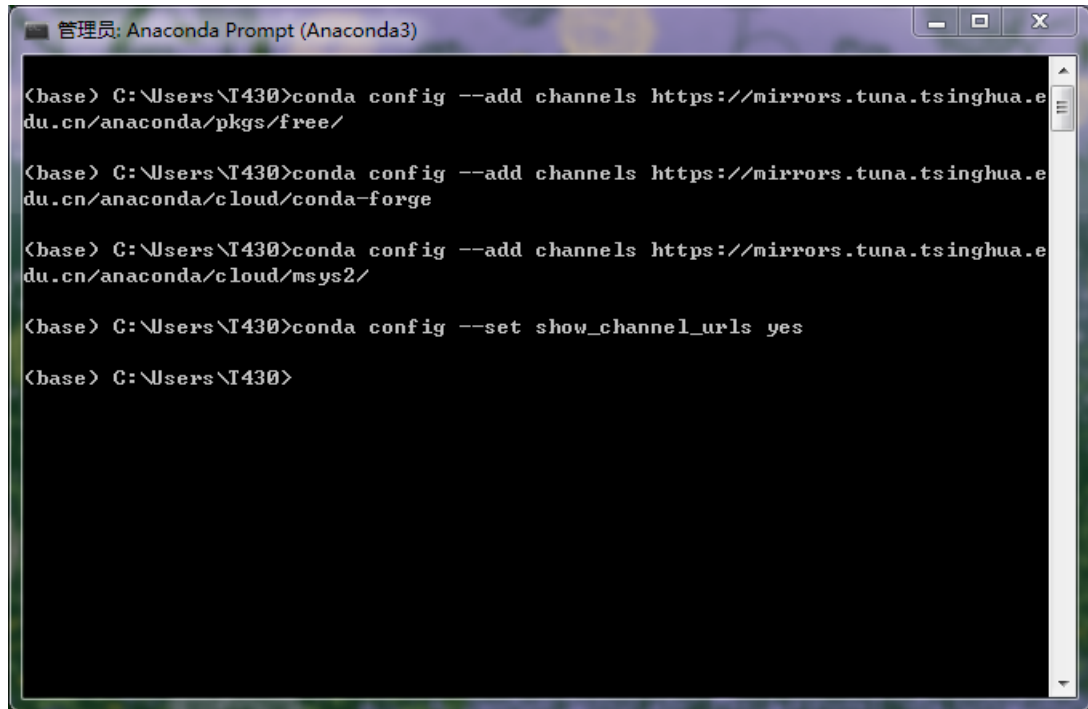
```
conda config --add channels
```

```
https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
```

```
conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/msys2/
```

```
conda config --set show_channel_urls yes
```

成功执行后，即将默认下载源更换为清华大学的镜像站，如图 1-12 所示。最后一条命令是设置提示通道地址。



```
管理员: Anaconda Prompt (Anaconda3)

(base) C:\Users\T430>conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkg/free/

(base) C:\Users\T430>conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge

(base) C:\Users\T430>conda config --add channels https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/msys2/

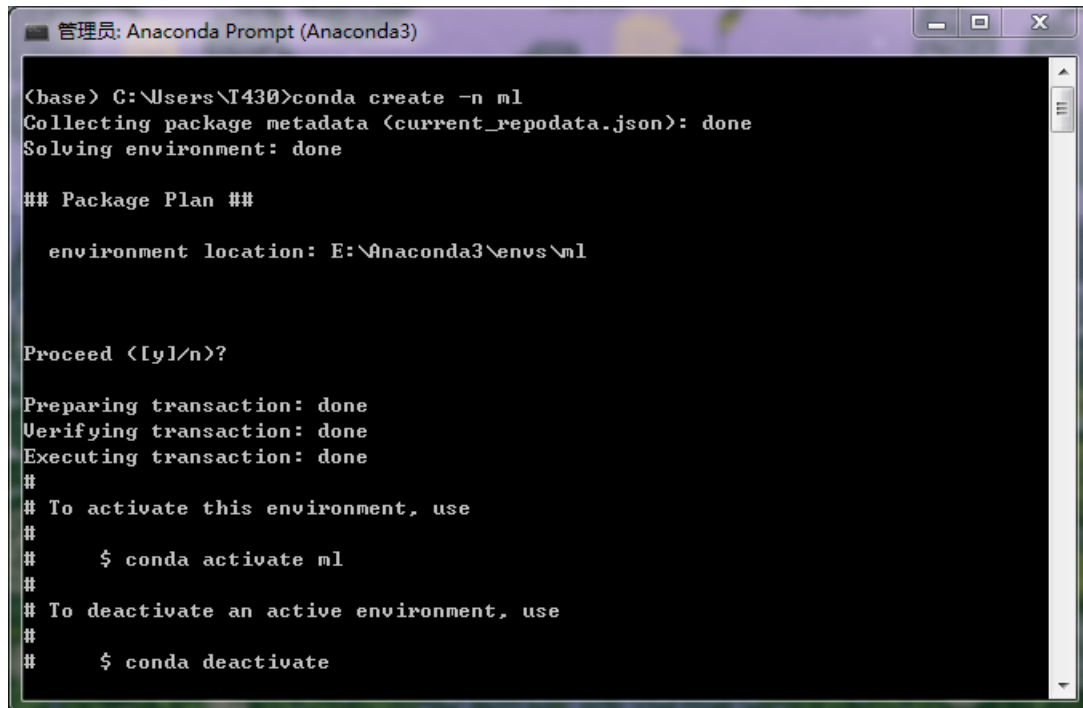
(base) C:\Users\T430>conda config --set show_channel_urls yes

(base) C:\Users\T430>
```

图 1-12 Conda 环境下更换下载源

2.新建一个专用于机器学习的名为 ml 的环境。

在 conda 命令行管理界面中输入并执行“conda create -n ml”命令后，即新建一个名为 ml 的环境，如图 1-13 所示。



```
(base) C:\Users\T430>conda create -n ml
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: E:\Anaconda3\envs\ml

Proceed [y]/n)?
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate ml
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

图 1-13 Conda 环境下新建环境

3. 安装扩展库

新建环境是空环境，需要安装各种库，anaconda 会自动安装各种依赖库。安装库的命令为：conda install ***。

输入并执行“activate ml”命令切换到新建的 ml 环境中。

输入并执行以下命令以安装本书示例所使用的库：

```
conda install python=3.7.5
```

```
conda install scikit-learn=0.23.2
```

```
conda install tensorflow=2.0.0
```

在新的环境中，还需要安装各种辅助的库，如编程环境 spyder 和 jupyter 等。

1.3. Python 初步应用示例-迭代法

迭代法（iteration）与其说是一种算法，更是一种思想，它不像传统数学解析方法那样一步到位得到精确解，而是步步为营，逐次推进，逐步接近。迭代法是现代计算机求解问题的一种基本形式。迭代法又称辗转法或逐次逼近法。

迭代法的核心是建立迭代关系式。迭代关系式指明了前进的方式，只有正确的迭代关系式才能取得正确解。

来看一个示例。假设在空池塘中放入一颗水藻，该类水藻会每周长出三颗新的水藻，问十周后，池塘中有多少颗水藻？

该问题可以用数学方法来直接计算。这里来看看如何用迭代法求解。

第 1 周的水藻数量：1；

第 2 周的水藻数量： $1 + 1 \times 3$ ；

第 3 周的水藻数量： $1 + 1 \times 3 + (1 + 1 \times 3) \times 3$

...

可以归纳出从当前周水藻数量到下一周水藻数量的迭代关系式。设上周水藻数量为 x ，从上周到本周水藻将增加的数量为 y ，本周的水藻数量为 x' ，那么在一次迭代中：

$$\begin{aligned} y &\leftarrow 3x \\ x' &\leftarrow x + y \end{aligned} \quad 1-1$$

迭代开始时，水藻的数量为 1，为迭代法的初始条件。

迭代次数为 9（不包括第一周），为迭代过程的控制条件。

该示例实现的代码见代码 1-1，迭代过程共循环 9 次，用 while 语句来循环实现迭代，一般用一轮循环来实现一次迭代。读者可以自己尝试改用 for 语句方式来实现迭代。

代码 1-1 迭代法应用示例（迭代法.ipynb）

```
1 x = 1 # 初始条件：第一周水藻数量
2 times = 1 # 迭代次数
3 while times < 10: # 迭代过程
4     y = 3 * x
5     x = x + y
6     times += 1
7     print("第%d周的水藻数量:%d" % (times, x))
8 >>> 第 2 周的水藻数量:4
9 >>> 第 3 周的水藻数量:16
10 >>> 第 4 周的水藻数量:64
11 >>> 第 5 周的水藻数量:256
12 >>> 第 6 周的水藻数量:1024
13 >>> 第 7 周的水藻数量:4096
14 >>> 第 8 周的水藻数量:16384
15 >>> 第 9 周的水藻数量:65536
16 >>> 第 10 周的水藻数量:262144
```

迭代法是求解机器学习问题的基本方法，有着广泛的应用，比如机器学习领域大名鼎鼎的梯度下降法就是一种以梯度来建立迭代关系式的迭代法。本小节先用解方程的例子来说明

它在数值计算领域的应用，为后续讨论梯度下降法打下基础。

用迭代法求下列方程的解：

$$x^3 + \frac{e^x}{2} + 5x - 6 = 0 \quad 1-2$$

该方程很难用解析的方法直接求解。

应用迭代法来求解，如何建立迭代关系式呢？

在迭代法求解中，每次迭代都得到一个新的 x 值，将每次迭代得到的 x 值依序排列就可得到数列 $\{x_k\}$ 。设 x_0 为初值。在用迭代法求解方程时有个常用的迭代关系式建立方法：先将方程 $f(x) = 0$ 变换为 $x = \varphi(x)$ ，然后建立起迭代关系式：

$$x_{k+1} = \varphi(x_k) \quad 1-3$$

如果 $\{x_k\}$ 收敛于 x^* ，那么 x^* 就是方程的根，因为：

$$x^* = \lim_{k \rightarrow \infty} x_{k+1} = \lim_{k \rightarrow \infty} \varphi(x_k) = \varphi\left(\lim_{k \rightarrow \infty} x_k\right) = \varphi(x^*) \quad 1-4$$

即，当 $x = x^*$ 时，有 $f(x) = x - \varphi(x) = 0$ 。

按式 1-3 建立上例的迭代关系式为：

$$x = \frac{\left(6 - x^3 - \frac{e^x}{2}\right)}{5} \quad 1-5$$

迭代的结束条件是实际应用时需要考虑的问题，在该例中没有明确的结束条件。在无法预估时，可采用控制总的迭代次数的办法。也可以根据数列 $\{x_k\}$ 的变化情况来判断，如将 $|x_{k+1} - x_k|$ 的值小于某个阈值作为结束的标准。还可以将两种办法结合使用。

用迭代法求解方程的示例代码如代码 1-2 所示。该示例采用控制总的迭代次数作为结束的条件。这里将初始值设为 0，读者可以设为其他值来观察一下迭代过程，要注意的是，不同的初始值可能会导致数列 $\{x_k\}$ 不收敛。

代码 1-2 迭代法求解方程示例 1（迭代法.ipynb）

```
1 import math
2 x = 0
3 for i in range(100):
4     x = (6 - x**3 - (math.e**x)/2.0)/5.0
5     print(str(i)+"-"+str(x))
```

运行结果显示从 28 次迭代开始，收敛于 0.84592。

第 1 行导入了 math 库，并在第 4 行使用了它的指数函数。

math 库包含丰富的数学函数，如果内置函数库不够用时，可以到 math 库中去找合适的

数学函数。

代码 1-3 给出了更常用的采用阈值来控制迭代结束的示例，如果相邻两次迭代 x_k 的差值小于指定的 delta，则通过 break 语句退出迭代。

代码 1-3 迭代法求解方程示例 2（迭代法.ipynb）

```
1 x = 0 # 初始条件
2 delta = 0.00001 # 控制退出条件
3 times = 0 # 用来显示迭代次数
4 while True: # 条件为 True，如果没有别的退出手段，while 循环将会无限进行下去
5     x_old = x
6     x = (6 - x**3 - (math.e**x)/2.0)/5.0
7     print(str(times)+":"+str(x))
8     times += 1
9     if abs(x - x_old) < delta:
10         break # 如果符合退出条件，则直接退出循环
```

运行结果显示在第 28 次循环退出迭代。

第 8 行使用了赋值运算符。

实验项目2 PCA 降维

2.1.PCA 降维过程印证

顾名思义，主成分分析是找出主要成分来代替原来数据。用二维平面上的例子来简要说明其过程，如图 2-1 所示。在二维平面上有 x_1, x_2, x_3, x_4 四个点，坐标分别是 $(4, 2)$ 、 $(0, 2)$ 、 $(-2, 0)$ 和 $(-2, -4)$ ，它们满足所谓中心化要求，即 $\sum_{i=1}^4 x_i = 0$ 。对于不满足中心化要求的点，可通过减所有点的均值来满足该要求。

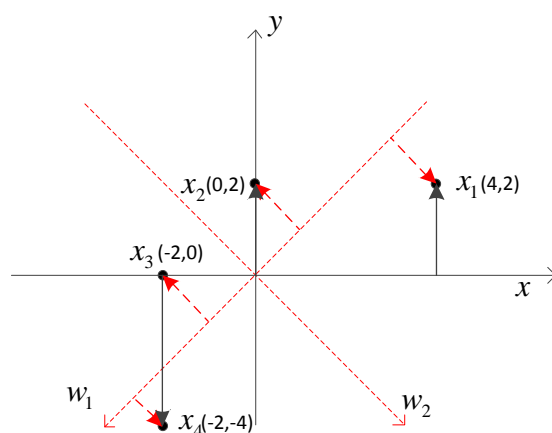


图 2-1 二维平面上的主成分降维示例

现在要将这四个点从二维降为一维，怎么降呢？一个很自然的想法是直接去掉每个点的一个坐标。比如，去掉 y 轴上的坐标，只保留 x 轴上的坐标。这实际上是用各点在 x 轴上的投影来代替原来的点，带来的误差为它们在 y 轴上的投影向量，如图中从 x 轴指向各点的带箭头实线所示，其中 x_3 点在 x 轴上，因此没有误差向量。

降维必定会带来误差，如何使总体误差最小是降维算法追求的目标。用所有误差向量的模的平方之和作为损失函数来衡量降维带来的误差（类似于误差平方和损失函数 SSE）。

试着同步旋转 x 和 y 轴，使得去掉 y 轴上的坐标带来的损失函数最小。比如 x 和 y 坐标轴保持正交旋转到图中的 w_1 和 w_2 坐标轴，降维的结果是只保留各点在 w_1 上的投影，放弃在 w_2 上的投影，所带来的误差向量为图中带箭头的虚线所示。

在此例中，是从二维降到一维，即用点到线的投影来代替平面上的点。如果在三维立体空间中，可将空间中的点投影到一个平面上或者一条线上。进一步推广，可以将多维空间中的点投影到一个低维的超平面上。

在 sklearn 扩展库的 decomposition 模块中实现了 PCA 算法。先用它来印证上述分析过

程，如代码 2-1 所示。

代码 2-1 二维平面上的主成分降维示例（二维平面上的主成分降维示例.ipynb）

```
1 x = [[4,2], [0,2], [-2,0], [-2,-4]] # 平面上四个点的坐标
2
3 from sklearn.decomposition import PCA
4
5 pca = PCA(n_components=2) # 只旋转不降维
6 pca.fit(x)
7 print("新的轴向量: ")
8 print(pca.components_)
9 print("各维度投影方差占比分布: ")
10 print(pca.explained_variance_ratio_)
11 print("各点在新轴上的投影: ")
12 print(pca.transform(x))
13 >>>
14 新的轴向量:
15 [[-0.70710678 -0.70710678]
16  [ 0.70710678 -0.70710678]]
17 各维度投影方差占比分布:
18 [0.83333333 0.16666667]
19 各点在新轴上的投影:
20 [[-4.24264069  1.41421356]
21  [-1.41421356 -1.41421356]
22  [ 1.41421356 -1.41421356]
23  [ 4.24264069  1.41421356]]
24
25 pca = PCA(n_components=1) # 降到一维
26 pca.fit(x)
27 print("新的轴向量: ")
28 print(pca.components_)
29 print("各维度投影方差占比分布: ")
30 print(pca.explained_variance_ratio_)
31 print("各点在新轴上的投影: ")
32 print(pca.transform(x))
33 >>>
34 新的轴向量:
35 [[-0.70710678 -0.70710678]]
36 各维度投影方差占比分布:
37 [0.83333333]
38 各点在新轴上的投影:
39 [[-4.24264069]
40  [-1.41421356]
41  [ 1.41421356]
```

第 5 行实例化 PCA 类，当把参数 `n_components` 设为与原特征数相同时，它只旋转不降维。第 6 行是用四个点的坐标来训练算法。第 8 行通过属性 `components_` 输出旋转后的轴向量，如第 15、16 行所示，第一个轴向量可以写成分数形式 $(-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$ ，可见它的方向是 x，y 轴中间 45 度。第 12 行用 `transform` 方法对四个点计算新的投影，如第 20 行到第 23 行所示。读者可以用几何知识验证一下。

第 10 行是通过属性 `explained_variance_ratio_` 观察各维度投影方差的占比分布，可以理解为各维度的成分比例，它是按从大到小排列输出，属性 `components_` 输出的新轴向量排列顺序要与它对应。输出如第 18 行所示。

第 25 行将参数 `n_components` 设为 1，即降到一维。对比第 14 行的输出和第 34 行的输出，可以看到，它是将第二个轴上的新投影直接丢弃了，保留了主要成分。

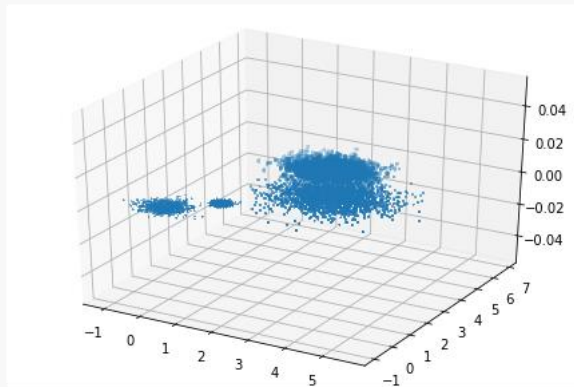
2.2.PCA 高维数据降维应用

下面来看一个可视化降维高维特征的示例。为了方便对比观察，不采用过高维的数据，而只用三维的数据示例，具体流程是先生成三维空间中分布的点，然后降到二维，观察并分析其过程。

在三维空间中生成四个簇，并察看它们的分布，见代码 2-2。

代码 2-2 在三维空间中产生簇（高维数据可视化降维示例.ipynb）

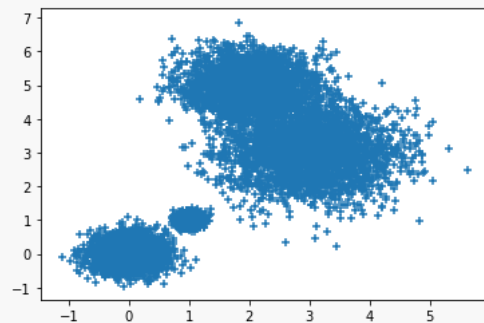
```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from mpl_toolkits.mplot3d import Axes3D
4 from sklearn.datasets import make_blobs
5
6 ##### 在三维空间中产生四个簇，共 1000 个样本
7 X, _ = make_blobs(n_samples=10000, n_features=3, centers=[[0,0,0],
8 [1,1,0.5], [3,3,3], [2,5,10]], cluster_std=[0.3, 0.1, 0.7, 0.5])
9 fig = plt.figure()
10 ax = Axes3D(fig)
11 plt.scatter(X[:, 0], X[:, 1], X[:, 2], marker='+')
```



```

11 >>>
12 ##### 看一下它们在三个面上的投影
13 plt.scatter(X[:, 0], X[:, 1], marker='+')
14 plt.show()

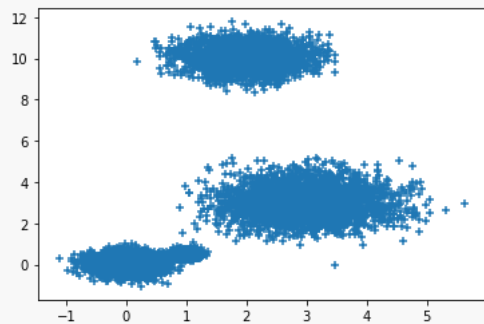
```



```

15 >>>
16 plt.scatter(X[:, 0], X[:, 2], marker='+')
17 plt.show()

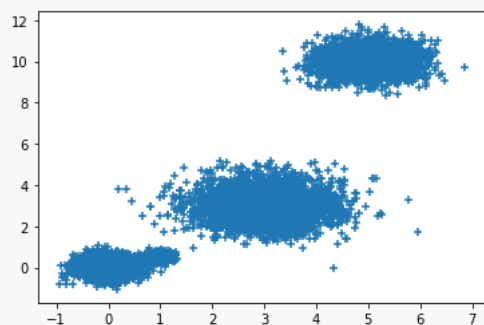
```



```

18 >>>
19 plt.scatter(X[:, 1], X[:, 2], marker='+')
20 plt.show()

```



```

21 >>>

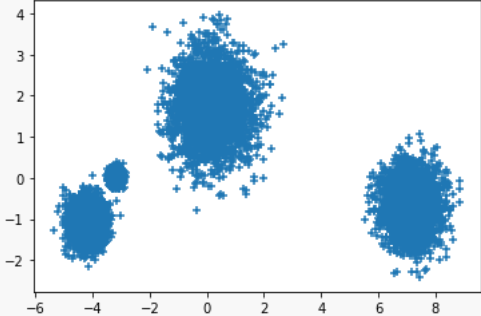
```

sklearn 扩展库的 datasets 模块中的 Make_blobs() 函数产生各向同性的高斯分布（即常说的正态分布）的样本数据。本例中，用它在三维空间中以指定标准差在指定的中心产生了四个簇，如第 7 行所示。第 8 行到第 10 行画出三维的分布图。然后分别看一下他们在三个面上的投影，可见每个面上的投影都有两个簇重叠的情况，不好区分开。

用 PCA 对它们进行降维，共进行了三次，见代码 2-3。第一次降到一个二维的平面上，可见可以较好地分开为四个簇。第二次和第三次通过指定 n_components 为一个小数来要求降维后保留的主成分占比。第二次要求保留 90%（第 11 行），此时，降到一维就可以达到要求了。第三次要求保留 99%（第 18 行），此时，不能降低维数，否则就达不到该要求。

代码 2-3 PCA 降维三维空间中的点（高维数据可视化降维示例.ipynb）

```
1  pca = PCA(n_components=2)
2  pca.fit(X)
3  print(pca.explained_variance_ratio_)
4  >>> [0.92755398 0.06230942]
5  X_new = pca.transform(X)
6  plt.scatter(X_new[:, 0], X_new[:, 1], marker='+')
7  plt.show()
8  >>>
```



```
9
10
11  pca = PCA(n_components=0.9) # 指定保留的主成分占比
12  pca.fit(X)
13  print(pca.explained_variance_ratio_)
14  print("降维后的特征数: " + str(pca.n_components_))
15  >>> [0.92755398]
16  降维后的特征数: 1
17
18  pca = PCA(n_components=0.99) # 指定保留的主成分占比
19  pca.fit(X)
20  print(pca.explained_variance_ratio_)
21  print("降维后的特征数: " + str(pca.n_components_))
22  >>> [0.92755398 0.06230942 0.0101366 ]
23  降维后的特征数: 3
```

该示例中，通过保留主要成分，将数据降至二维，可以直观地观察到数据的分布情况。在进行聚类 and 分类时，如果能提前观察到样本在空间的大概分布，对于选择合适的算法是很重要的。

实验项目3 聚类算法应用及比较

下面给出一个从多方面综合分析划分聚类、密度聚类和模型聚类，以及聚类算法内部评价指标的示例。该示例先生成三种二维平面上的实验数据和一种高维空间中的实验数据，然后分别用 kmeans、DBSCAN 和 GaussianMixture 三种算法对它们进行聚类，并计算 SC、DBI、CH 和 ZQ 四个指标。该示例展示了实验样本点的分布与聚类算法适用性、评价指标值有效性的关系。示例的实现代码见附属资源中的文件“聚类算法综合比较示例.ipynb”。

三种二维平面上的实验样本分布如图 3-1 所示，他们分别是圆环、高斯分布和月牙形状

的，由 datasets 模块中相应的函数产生。

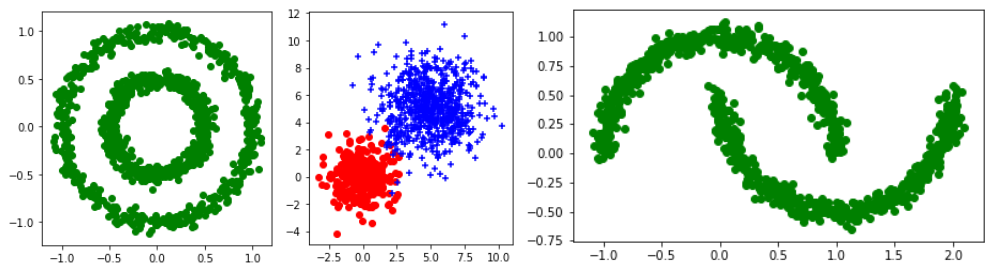


图 3-1 二维平面上的三种实验样本分布

高维空间中的实验样本通过 PCA 降维后，在二维平面上的分布如图 3-2 所示。它是由 datasets 模块中的 make_gaussian_quantiles () 函数在四维空间中以原点 (0, 0, 0, 0) 为中心，按高斯分布随机产生的，由内向外分为 9 层的类球状分布。随后去掉第 1-6 层和第 8 层，只保留内核的第 0 层和外面的第 7 层。可以将此数据想像成一个带核的空心四维类球体。

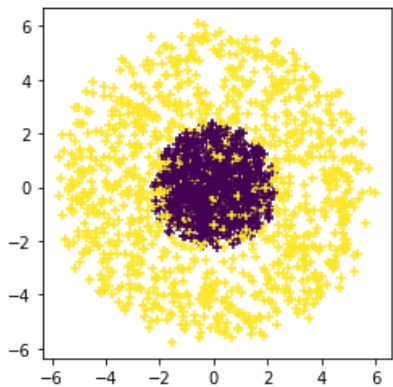
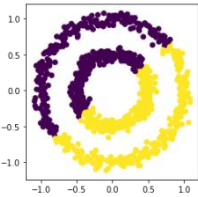
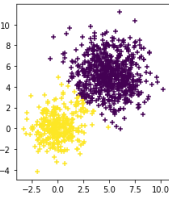
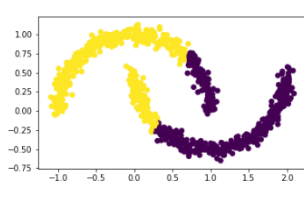
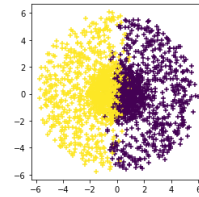
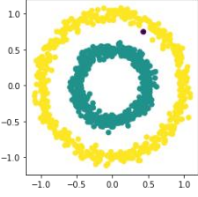
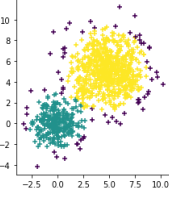
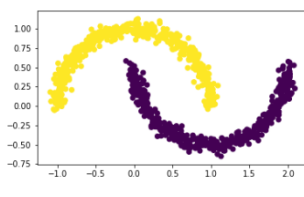
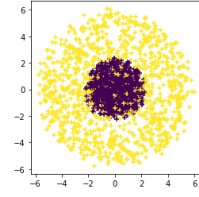
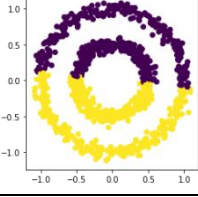
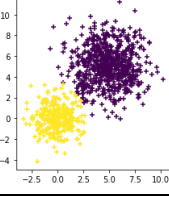
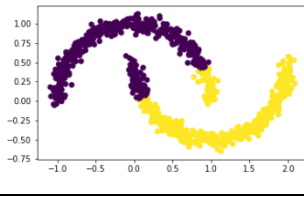
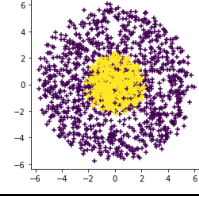


图 3-2 高维空间中实验样本降维后的分布

算法运行后的聚类效果及各评价指标值如表 3-1 所示。

表 3-1 聚类综合示例结果

	圆环	高斯分布	月牙	四维类球体
kmeans				
	SC:0.35418150800 DBI:1.1828798822 CH:576.234365353 ZQ:1.16871702321	SC:0.595507603 DBI:0.54554156 CH:2088.867795 ZQ:0.789103847	SC:0.4894026970788733 DBI:0.7797138414306899 CH:1487.6573120666626 ZQ:1.4870808283885213	SC:0.14673718069 DBI:2.0836980081 CH:411.728662603 ZQ:2.59313957972
DBSCAN				
	SC:-0.0688868183 DBI:150.72658703 CH:0.60426715731 ZQ:0.07193600153	SC:0.497113248 DBI:3.39191627 CH:800.1776282 ZQ:1.228896011	SC:0.33345414657834466 DBI:1.1539812259101607 CH:663.1677674098607 ZQ:0.06232429416022914	SC:0.16606150306 DBI:98.110576302 CH:0.18240617409 ZQ:0.21742577037
Gaussian Mixture				
	SC:0.35177000142 DBI:1.1896529582 CH:569.108807414 ZQ:1.24921647476	SC:0.596057416 DBI:0.52269590 CH:2020.497457 ZQ:1.032134517	SC:0.4680579424239563 DBI:0.823742120847451 CH:1282.7601223372637 ZQ:2.5003513365473453	SC:0.16606150306 DBI:98.110576302 CH:0.18240617409 ZQ:0.21742577037

DBSCAN 算法对非凸簇（四维类球体也是非凸簇）有较好的聚类效果。GaussianMixture 算法对高斯分布的簇有较好的聚类效果，四维类球体样本集也是按高斯分布产生的，因此，它可以很好地学习到模型参数。高斯分布的样本集在实际工程中比较常见。

预先探索样本集在空间中的分布对于选择合适的聚类算法很重要。除了通过降维来直观观察样本集在空间中的分布外，聚类内部评价指标也可以帮助分析。比如在面对大数据量的聚类任务时，可以先随机抽取或者划分网格抽取小部分样本进行试分簇，如果发现运行 DBSCAN 算法后的 ZQ 指标改善较多，而其他指标变差，则样本集可能是非凸的分布。

实验项目4 房价回归

该实验是依据房屋的属性信息，包括房屋的卧室数量、卫生间数量、房屋的大小、房屋地下室的大小、房屋的外观、房屋的评分、房屋的修建时间、房屋的翻修时间、房屋的位置信息等，对房屋的价格进行预测。

1. 初步数据分析

从 Kaggle 官网下载数据后，用 pandas 进行初步分析，发现数据完整，没有缺失和重复的现象，相关代码见代码 4-1。

代码 4-1 房价回归的初步数据分析（Kaggle 房价预测.ipynb）

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 sns.set()
5 raw_data = pd.read_csv('kc_house_data.csv')
6 raw_data
7 >>>
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	
...
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131	3.0	0	0	...	8	1530	0	
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813	2.0	0	0	...	8	2310	0	
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350	2.0	0	0	...	7	1020	0	
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388	2.0	0	0	...	8	1600	0	
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076	2.0	0	0	...	7	1020	0	

```
8 21613 rows x 21 columns
9
10 raw_data.info()
11 >>>
12 <class 'pandas.core.frame.DataFrame'>
13 RangeIndex: 21613 entries, 0 to 21612
14 Data columns (total 21 columns):
15 #   Column          Non-Null Count  Dtype
16 ---  ---
17 0    id              21613 non-null  int64
18 1    date            21613 non-null  object
19 2    price           21613 non-null  float64
20 3    bedrooms        21613 non-null  int64
21 4    bathrooms       21613 non-null  float64
22 5    sqft_living     21613 non-null  int64
```

```

23     6   sqft_lot      21613 non-null  int64
24     7   floors       21613 non-null  float64
25     8   waterfront   21613 non-null  int64
26     9   view         21613 non-null  int64
27    10   condition    21613 non-null  int64
28    11   grade        21613 non-null  int64
29    12   sqft_above   21613 non-null  int64
30    13   sqft_basement 21613 non-null  int64
31    14   yr_built     21613 non-null  int64
32    15   yr_renovated  21613 non-null  int64
33    16   zipcode      21613 non-null  int64
34    17   lat          21613 non-null  float64
35    18   long         21613 non-null  float64
36    19   sqft_living15 21613 non-null  int64
37    20   sqft_lot15   21613 non-null  int64
38   dtypes: float64(5), int64(15), object(1)
39   memory usage: 3.5+ MB
40
41   raw_data.duplicated().sum()
42   >>> 0

```

2. 划分训练集和验证集，并标准化

因为 id 特征为顺序号，可去掉。此外，date 特征是交易时间，应与交易价格关系不大，也去掉（读者可思考如何分析二者之间的相关性）。

用 sklearn.model_selection 中的 train_test_split（）划分训练集和验证集。

用 sklearn.preprocessing 中的 StandardScaler（）对特征进行标准化。

相关代码见代码 4-2。

代码 4-2 房价回归的划分训练集和验证集并标准化（Kaggle 房价预测.ipynb）

```

1   X = raw_data.drop(['id', 'date', 'price'], axis=1)
2   y = raw_data['price']
3
4   from sklearn.model_selection import train_test_split
5   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
6   random_state=1026)
7
8   from sklearn.preprocessing import StandardScaler
9   sc = StandardScaler()
10  sc.fit(X_train)
11  X_train= sc.transform(X_train)
12  X_test = sc.transform(X_test)

```

3. 初步建立模型

选择 K 近邻回归、决策树回归、随机森林回归和梯度提升树回归等多个模型进行初步实验，用其中得分最高的梯度提升树模型进行下一步的超参数调优。

用平均绝对误差、均方误差和 R 方值³作为评价指标。

代码见代码 4-3。

代码 4-3 房价回归的初步建立模型（Kaggle 房价预测.ipynb）

```
1 import time
2 from sklearn.metrics import mean_absolute_error, mean_squared_error,
  r2_score
3
4 from sklearn.neighbors import KNeighborsRegressor
5 model = KNeighborsRegressor(n_neighbors=10)
6 time_start=time.time()
7 model.fit(X_train, y_train)
8 print('K 近邻回归模型训练用时: ', time.time()-time_start)
9 y_pred=model.predict(X_test)
10 print ('K 近邻回归模型在验证集上的平均绝对误差、均方误差和 R 方值分别为: ',
11        mean_absolute_error(y_test,y_pred),
12        mean_squared_error(y_test,y_pred), r2_score(y_test,y_pred))
13 >>>K 近邻回归模型训练用时:  0.37302160263061523
14      K 近邻回归模型在验证集上的平均绝对误差、均方误差和 R 方值分别为:
15      91041.56878470081 32515907041.252583 0.7719926355916296
16
17 from sklearn.tree import DecisionTreeRegressor
18 model = DecisionTreeRegressor()
19 time_start=time.time()
20 model.fit(X_train, y_train)
21 print('决策树回归模型训练用时: ', time.time()-time_start)
22 y_pred=model.predict(X_test)
23 print ('决策树回归模型在验证集上的平均绝对误差、均方误差和 R 方值分别为: ',
24        mean_absolute_error(y_test,y_pred),
25        mean_squared_error(y_test,y_pred), r2_score(y_test,y_pred))
26 >>>决策树回归模型训练用时:  0.26601529121398926
27      决策树回归模型在验证集上的平均绝对误差、均方误差和 R 方值分别为:
28      99859.92396668723 31894473839.09855 0.7763502365159782
29
30 from sklearn.ensemble import RandomForestRegressor
31 model = RandomForestRegressor(n_estimators=500)
```

```

28     time_start=time.time()
29     model.fit(X_train, y_train)
30     print('随机森林回归模型训练用时: ', time.time()-time_start)
31     y_pred=model.predict(X_test)
32     print('随机森林回归模型在验证集上的平均绝对误差、均方误差和 R 方值分别为: ',
33           mean_absolute_error(y_test,y_pred),
34           mean_squared_error(y_test,y_pred), r2_score(y_test,y_pred))
35     >>>随机森林回归模型训练用时:  83.69578719139099
36         随机森林回归模型在验证集上的平均绝对误差、均方误差和 R 方值分别为:
37         70227.1909673897 17132073455.97604 0.8798668322371602
38
39     from sklearn.ensemble import GradientBoostingRegressor
40     model = GradientBoostingRegressor(n_estimators=500)
41     time_start=time.time()
42     model.fit(X_train, y_train)
43     print('梯度提升树回归模型训练用时: ', time.time()-time_start)
44     y_pred=model.predict(X_test)
45     print('梯度提升树回归模型在验证集上的平均绝对误差、均方误差和 R 方值分别为: ',
46           mean_absolute_error(y_test,y_pred),
47           mean_squared_error(y_test,y_pred), r2_score(y_test,y_pred))
48     >>> 梯度提升树回归模型训练用时:  20.54817533493042
49         梯度提升树回归模型在验证集上的平均绝对误差、均方误差和 R 方值分别为:
50         69371.1884622335 14950726186.854908 0.8951628300102105

```

4. 超参数调优

对梯度提升树的 `loss`、`learning_rate` 和 `min_samples_leaf` 三个超参数分别设置搜索值，进行网格搜索，代码见代码 4-4。

代码 4-4 房价回归的网格搜索超参数调优（Kaggle 房价预测.ipynb）

```

1     from sklearn.model_selection import GridSearchCV
2
3     model_slect = GradientBoostingRegressor()
4     parameters = {'loss': ['ls','lad','huber','quantile'],
5                   'learning_rate':[0.1, 0.2], 'min_samples_leaf': [1,2,3,4]}
6     time_start=time.time()
7     model_gs = GridSearchCV(estimator=model_slect, param_grid=parameters,
8                             verbose=3)
9     model_gs.fit(X,y)
10    print('网络搜索用时: ', time.time()-time_start)
11    print('最高得分:', model_gs.best_score_)
12    print('最好参数:', model_gs.best_params_)
13    >>> 网络搜索用时:  850.472644329071
14        最高得分: 0.8771974305698761

```

```

13     最好参数: {'learning_rate': 0.2, 'loss': 'ls', 'min_samples_leaf': 3}
14
15     model = GradientBoostingRegressor(n_estimators=500,
16     learning_rate=0.2, loss='ls', min_samples_leaf=4)
17     time_start=time.time()
18     model.fit(X_train, y_train)
19     print('调优后梯度提升树回归模型训练用时: ', time.time()-time_start)
20     y_pred=model.predict(X_test)
21     print ('调优后梯度提升树回归模型在验证集上的平均绝对误差、均方误差和 R 方值分别为:
22     ',
23     mean_absolute_error(y_test,y_pred),
24     mean_squared_error(y_test,y_pred), r2_score(y_test,y_pred))
25 >>> 调优后梯度提升树回归模型训练用时: 20.94319772720337
26     调优后梯度提升树回归模型在验证集上的平均绝对误差、均方误差和 R 方值分别为:
27     67721.28374524636 14424980969.361366 0.8988494496465295

```

可见调优后的模型有了一定的提升。

5. 特征选择

如果因为某种原因要去掉一些特征，如何来分析呢？下面讨论一些常用方法。

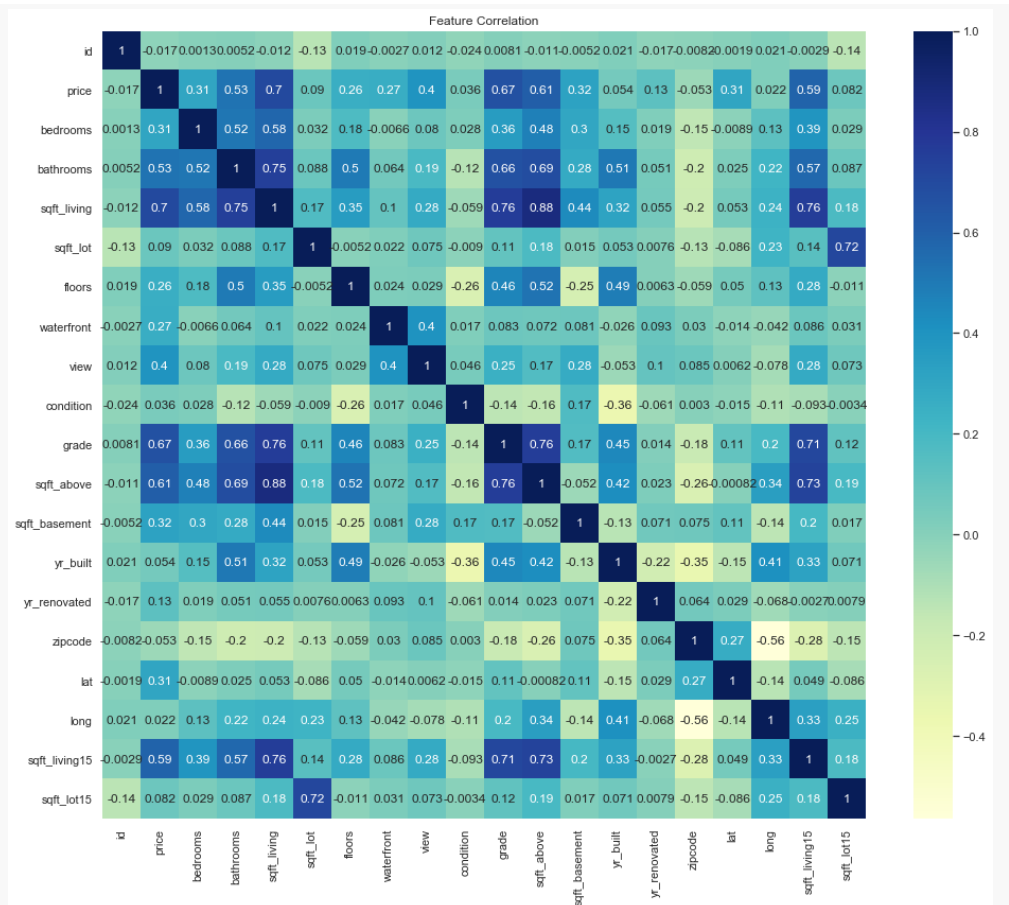
用 dataframe 的 corr() 函数和 plot() 函数可以分析特征之间以及特征和标签值之间的相关系数，见代码 4-5。

代码 4-5 房价回归的特征相关分析（Kaggle 房价预测.ipynb）

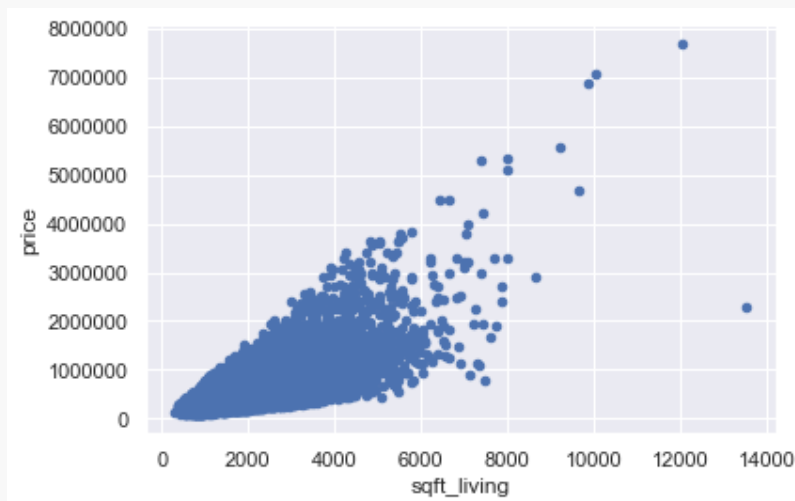
```

1     plt.figure(figsize=(14,12))
2     sns.heatmap(raw_data.corr(), annot=True, cmap="YlGnBu")
3     plt.title('Feature Correlation')
4     plt.tight_layout()
5     plt.show()
6     >>>

```



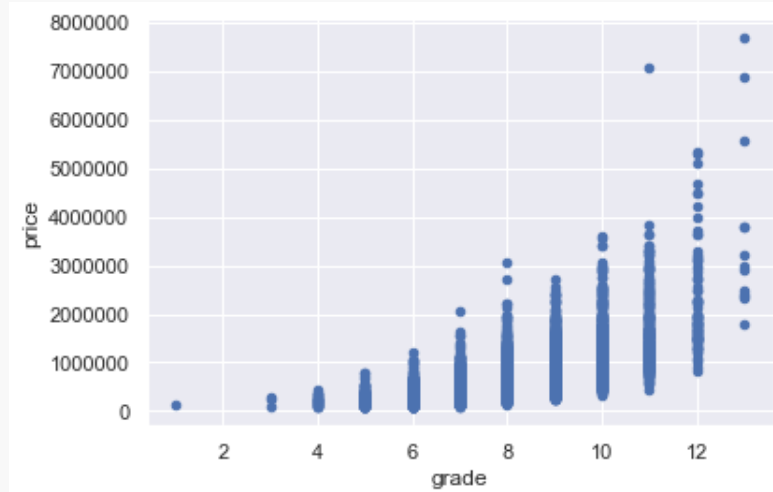
```
raw_data.plot(kind='scatter', x='sqft_living', y='price')
```



```
>>>
```

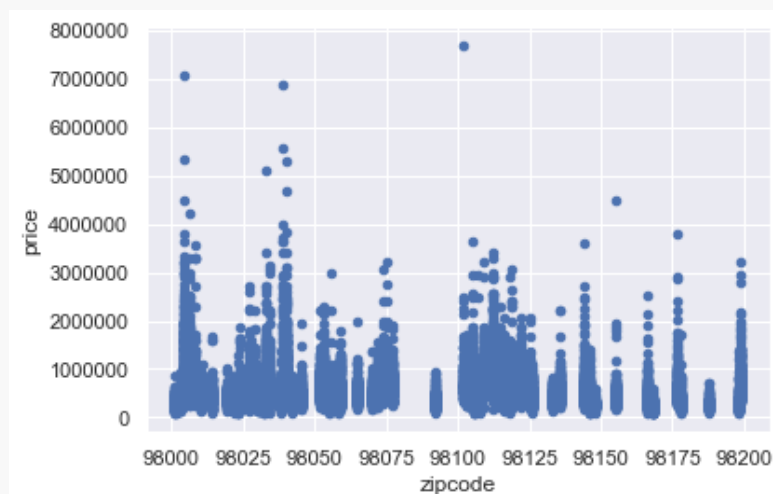
```
raw_data.plot(kind='scatter', x='grade', y='price')
```

13 >>>



14

15 raw_data.plot(kind='scatter', x='zipcode', y='price')



16 >>>

第 6 行的输出的图中，两个特征或特征与标签交叉的方块的颜色越深表示相关系数越大。可见 sqft_living 和 grade 特征与 price 标签相关性最大，zipcode 特征与 price 标签相关性最小，分别用散点图画出他们的分布，如第 10 行、第 13 行和第 16 行所示。由图可直接观察出前两个特征的取值趋势与价格的变化相近。

下面用模型来尝试去特征，为了使读者更容易理解其思想，采用轮流去掉特征进行模型训练的方法，而不是采用已经封装好的 sklearn.feature_selection.RFECV() 函数，代码见。

代码 4-6 房价回归的模型尝试选择特征（Kaggle 房价预测.ipynb）

```
1 columns = (X.columns).tolist()
2
3 for column in columns:
4     X1 = X.drop([column], axis=1)
5     #print(X1.columns)
```

```

6     X1_train, X1_test, y_train, y_test = train_test_split(X1, y,
test_size=0.3, random_state=1026)
7     sc.fit(X1_train)
8     X1_train= sc.transform(X1_train)
9     X1_test = sc.transform(X1_test)
10    time_start=time.time()
11    model.fit(X1_train, y_train)
12    print('去掉', column, '特征后的训练用时: ', time.time()-time_start)
13    y_pred=model.predict(X1_test)
14    print ('平均绝对误差、均方误差和 R 方值分别为: ',
15           mean_absolute_error(y_test,y_pred),
mean_squared_error(y_test,y_pred), r2_score(y_test,y_pred), "\n")
16    >>> 去掉 bedrooms 特征后的训练用时:  20.881194353103638
17    平均绝对误差、均方误差和 R 方值分别为:  67857.9093675719  14464470845.953144
0.8985725395584585
18    ...
19    去掉 sqft_living15 特征后的训练用时:  19.423110961914062
20    平均绝对误差、均方误差和 R 方值分别为:  68798.94891719022  15014902942.150944
0.8947128111067622
21
22    去掉 sqft_lot15 特征后的训练用时:  17.51600170135498
23    平均绝对误差、均方误差和 R 方值分别为:  67282.29665020402  14370580591.668686
0.8992309148390826

```

用 R 方值来衡量，第一个应去掉的特征是：sqft_basement，因为去掉该特征后，模型在验证集上取得最大 R 方值：0.903464213147279。

6. 神经网络模型

最后，尝试用全连接层神经网络来建模该回归问题，在 TensorFlow2 框架下实现，代码见代码 4-7。

代码 4-7 房价回归的神经网络模型（Kaggle 房价预测.ipynb）

```

1     import tensorflow as tf
2     tf_model = tf.keras.Sequential([
3         tf.keras.layers.Dense(50, activation='relu', input_shape=(18,),
kernel_initializer='random_uniform', bias_initializer='zeros'),
4         tf.keras.layers.Dense(50, activation='relu',
kernel_initializer='random_uniform', bias_initializer='zeros'),
5         tf.keras.layers.Dense(1, activation='relu',
kernel_initializer='random_uniform', bias_initializer='zeros')
6     ])
7
8     batch_size = 50 # 每批训练样本数（批梯度下降法）

```



```

9     tf_epoch = 2000
10    tf_model.compile(optimizer='adam', loss='mean_squared_error')
11    tf_model.fit(X_train.tolist(), y_train.tolist(), batch_size=batch_size,
12                epochs=tf_epoch, verbose=1)
13    tf_model.summary()
14    >>> ...
15    Epoch 1999/2000
16    15129/15129 [=====] - 4s 289us/sample - loss:
17    12648700270.9519s - loss: - ETA: 0s - loss: 12
18    Epoch 2000/2000
19    15129/15129 [=====] - 5s 303us/sample - loss:
20    12650739872.2771
21    Model: "sequential_1"
22
23    _____
24    Layer (type)                Output Shape                Param #
25    _____
26    dense_3 (Dense)              (None, 50)                  950
27    _____
28    dense_4 (Dense)              (None, 50)                  2550
29    _____
30    dense_5 (Dense)              (None, 1)                   51
31    _____
32
33    Total params: 3,551
34    Trainable params: 3,551
35    Non-trainable params: 0
36
37    _____
38
39    y_pred = tf_model.predict(X_test.tolist())
40    print ('平均绝对误差、均方误差和 R 方值分别为: ',
41          mean_absolute_error(y_test,y_pred),
42          mean_squared_error(y_test,y_pred), r2_score(y_test,y_pred))
43    >>>平均绝对误差、均方误差和 R 方值分别为: 73302.36958040948 16589982477.299114
44    0.8836680712787428

```

采用了 4 层全连接层神经网络，中间两个隐层的节点数为 50，采用 relu 激活函数、MSE 损失函数和 adam 优化算法。

以每批 50 个样本进行训练 2000 轮，得到的模型在验证集上的得分情况如第 36 行输出所示。可见，该结构的神经网络模型效果不如梯度提升树。该模型在训练时，耗费了相对很长的时间。

实验项目5 电信用户流失分类

该实例数据同样来自 kaggle⁴，它的每一条数据为一个用户的信息，共有 21 个有效字段，其中，最后一个字段 Churn 标志该用户是否流失。

1. 数据初步分析

可用 pandas 的 `read_csv()` 函数来读取数据，用 DataFrame 的 `head()`、`shape`、`info()`、`duplicated()`、`nunique()` 等来初步观察数据，详情可参考随书资源“kaggle 电信用户流失分类.ipynb”文件，不再贴出占用过多篇幅。

用户信息可分为个人信息、服务订阅信息和帐单信息三类。

1) 个人信息包括 `gender`（性别）、`SeniorCitizen`（是否老年用户）、`Partner`（是否伴侣用户）和 `Dependents`（是否亲属用户）。

2) 服务订阅信息包括 `tenure`（在网时长）、`PhoneService`（是否开通电话服务业务）、`MultipleLines`（多线业务服务：Yes、No 或 No phoneservice）、`InternetService`（互联网服务：No、DSL 数字网络或光纤网络）、`OnlineSecurity`（网络安全服务：Yes、No 或 No internetserive）、`OnlineBackup`（在线备份业务服务：Yes、No 或 No internetserive）、`DeviceProtection`（设备保护业务服务：Yes、No 或 No internetserive）、`TechSupport`（技术支持服务：Yes、No 或 No internetserive）、`StreamingTV`（网络电视服务：Yes、No 或 No internetserive）、`StreamingMovies`（网络电影服务：Yes、No 或 No internetserive）。

服务订阅信息具有一定的关联性，比如 `PhoneService` 为 No 的话，`MultipleLines` 的值只能为 No phoneservice。同样，`InternetService` 的值对后面的几个特征的取值有相同的影响。因此，在对它们进行编码时要特别考虑这一点。

3) 帐单信息包括 `Contract`（签订合同方式：月、一年或两年）、`PaperlessBilling`（是否开通电子账单）、`PaymentMethod`（付款方式：bank transfer、credit card、electronic check 或 mailed check）、`MonthlyCharges`（月费用）、`TotalCharges`（总费用）。

2. 流失用户与非流失用户特征分析

该数据明确给出了用户是否流失类别，因此，可以通过观察不同特征的取值变化来分析流失用户与非流失用户的特点，从而为制定营销策略提供参考。

⁴ <https://www.kaggle.com/radmirzosimov/telecom-users-dataset>

1) 对于用来描述分类的对象型特征的分布, 可用统计图来直观显示。

代码 5-1 画出不同类用户的个人信息分布图 (kaggle 电信用户流失分类.ipynb)

```
1 import matplotlib.pyplot as plt
2 import seaborn as sns
3 fig, axes = plt.subplots(2, 2, figsize=(10, 8))
4 sns.countplot(x='gender', data=df, hue='Churn', ax=axes[0][0])
5 sns.countplot(x='SeniorCitizen', data=df, hue='Churn', ax=axes[0][1])
6 sns.countplot(x='Partner', data=df, hue='Churn', ax=axes[1][0])
7 sns.countplot(x='Dependents', data=df, hue='Churn', ax=axes[1][1])
```

代码 5-1 的作用是画出两类用户的个人信息分布, 结果如图 5-1 所示。

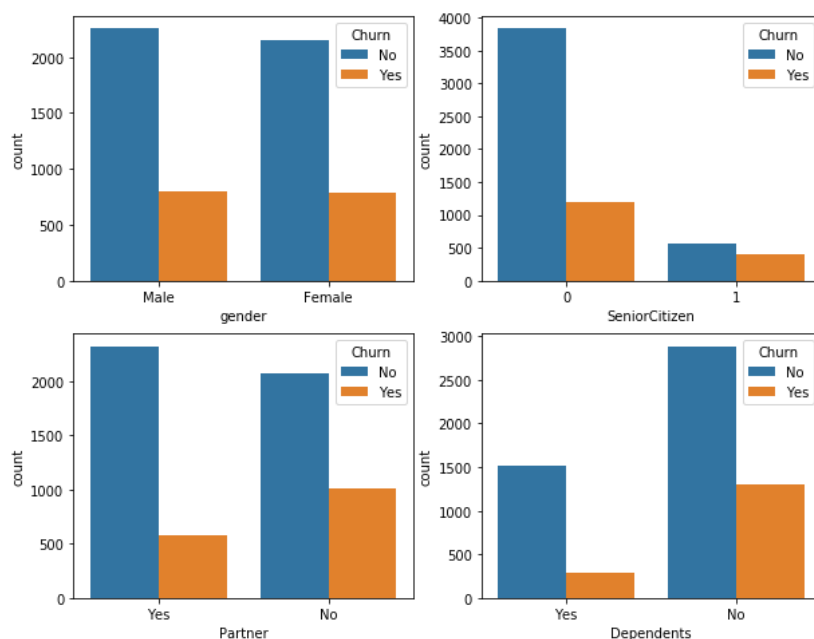


图 5-1 不同类用户的个人信息分布图

从图中可知, 老年用户的流失率要高一些, 伴侣用户的流失率要低一些, 亲属用户的流失率要低一些。

同样可画出其他分类特征与是否流失的分布图, 参见随书资源 “kaggle 电信用户流失分类.ipynb” 文件, 不再赘述。

2) 对于数值型特征的分布, 可用密度图来直观显示。

代码 5-2 画出在网时长密度图 (kaggle 电信用户流失分类.ipynb)

```
1 plt.rc('font', family='SimHei')
2 plt.title("在网时长密度图")
3 ax1 = sns.kdeplot(df[df['Churn'] == 'Yes']['tenure'], color='r',
4                   linestyle='-', label='Churn:Yes')
```

```
4 ax1 = sns.kdeplot(df[df['Churn'] == 'No']['tenure'], color='b',  
                    linestyle='--', label='Churn:No')
```

代码 5-2 的作用是画出在网时长密度分布，结果如图 5-2 所示。

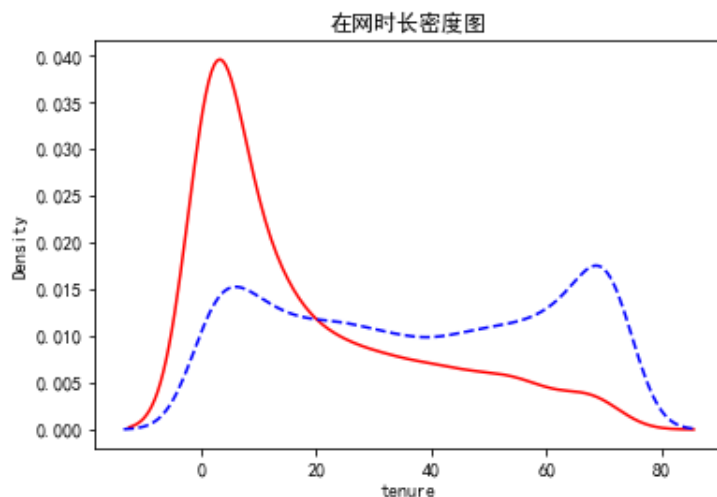


图 5-2 在网时长密度图

图中实线表示流失用户，虚线表示非流失用户。可见，新用户流失率要高一些。

同样可画出月费用密度图和总费用密度图。

要注意的是，原数据中，总费用是用对象型数据，需要先转换成数值型。在转换的过程中发现有个别数据为空格字符串，可采用删除数据或用 0 代替总费用特征的办法进行处理。

3. 分类预测

数据的类型分为对象型和数值型两类。对象型是离散的类别数据，需要对它们进行编码才能形成训练模型的特征。

如果是二值的对象型数据，可以直接用 0 和 1 来对它们进行编码。如果取值类别个数多于 2，一般可用独热编码。

对于需要进行距离计算的模型,一般还需要对数值型特征进行归一化处理或标准化处理。

经过上述处理后,采用保持法将训练样本切分为训练集和验证集,用来建模并验证模型。分别采用了多项式朴素贝叶斯模型、高斯朴素贝叶斯模型、逻辑回归模型、决策树模型、随机森林模型、装袋决策树模型、极端随机树模型、梯度提升树模型和多层全连接层神经网络模型。相关代码见随书资源“kaggle 电信用户流失分类.ipynb”文件,不再贴出。

不同模型对用户是否流失的预测的准确度和 AUC 值如所示。

表 5-1 各模型对用户流失预测的准确度和 AUC 值

模型	多项式朴素贝叶斯模型	高斯朴素贝叶斯模型	逻辑回归模型	决策树模型	随机森林模型	装袋决策树模型	极端随机树模型	梯度提升树模型	多层全连接层神经网络
----	------------	-----------	--------	-------	--------	---------	---------	---------	------------

指标\模型	型								络模型
准确度	0.7188	0.6765	0.8051	0.7293	0.8023	0.7934	0.7639	0.7861	0.8034
AUC	0.7473	0.7341	0.7223	0.6600	0.6980	0.6983	0.6581	0.6122	0.7379

可见，在这些模型中，多层全连接层神经网络模型取得了综合最好的预测效果，其次是随机森林模型和多项式朴素贝叶斯模型。

实验项目6 中文分词

6.1. 训练样本预处理

本节中文分词示例的语料库由已经分好词的句子组成，如“着力 促进 有 能力 在 城镇 稳定 就业 和 生活 的 常住 人口 有序 实现 市民化”，词与词之间用空格间隔。为了方便后续统计频率，先将它们转换成合适的标记格式，代码如代码 6-1 所示。

代码 6-1 训练样本预处理（中文分词示例.ipynb）

```
1 import numpy as np
2
3 file = open("traindata.txt", encoding='utf-8')
4 new_sents = []
5 sents_labels = []
6 for line in file.readlines():
7     line = line.split()
8     new_sent = ''
9     sent_labels = ''
10    for word in line:
11        if len(word) == 1:
12            new_sent += word
13            sent_labels += 'S'
14        elif len(word) >= 2:
15            new_sent += word
16            sent_labels += 'B' + 'M'*(len(word)-2) + 'E'
17    if new_sent != '':
18        new_sents.append([new_sent])
19        sents_labels.append([sent_labels])
20 print("训练样本准备完毕！")
21 print('共有数据 %d 条' % len(new_sents))
22 print('平均长度: ', np.mean([len(d[0]) for d in new_sents]))
23 >>>训练样本准备完毕！
24 共有数据 62946 条
25 平均长度: 8.67100371747212
```

处理完成后，不用空格间隔的句子存放于 new_sents 列表中，对应的 SBME 四标签序列存放于 sents_labels 列表中。

6.2. 隐马模型建模及预测

采用隐马尔可夫模型作为分词模型时，输入序列是观测值序列，输出的标签序列是隐状态序列。

应用隐马尔可夫模型来分词，先要用大量的训练样本来训练模型，属于监督学习。在自然语言处理领域，训练样本称为语料。

用语料来有监督训练隐马尔可夫模型比较容易理解。用概率论的极大似然估计法和矩估计法都可以得到用频率来估计概率的结论。也就是说，对隐马尔可夫模型的三要素 A, B, π ，只需要统计它们中的每个元素在语料库中出现的频率即可。为了防止出现概率为 0 的情况，可采取如多项式朴素贝叶斯分类器中用到的平滑技术。

统计频率得到隐马尔可夫模型并用来预测的代码见代码 6-2。

代码 6-2 中文分词隐马尔可夫模型示例（中文分词示例.ipynb）

```
1 # 统计初始概率矩阵 pi
2 state = ['S', 'B', 'M', 'E']
3 pi = np.zeros(4)
4 for i in range(len(sents_labels)):
5     if sents_labels[i][0][0] == 'S':
6         pi[0] += 1
7     if sents_labels[i][0][0] == 'B':
8         pi[1] += 1
9 pi /= np.sum(pi)
10
11 # 统计转移概率矩阵 A 和观测概率矩阵 B
12 A = np.zeros((4, 4))
13 B = np.zeros((4, 65536)) # GB2312 编码
14 for i in range(len(sents_labels)):
15     for j in range(len(sents_labels[i][0])):
16         B[state.index(sents_labels[i][0][j]), ord(new_sents[i][0][j])]
17         += 1 # 观测频率加 1
18         for j in range(len(sents_labels[i][0]) - 1):
19             A[state.index(sents_labels[i][0][j]),
20               state.index(sents_labels[i][0][j+1])] += 1 # 转移频率加 1
21
22 for i in range(4):
23     if np.sum(A[i]) != 0:
24         A[i] = A[i] / np.sum(A[i])
25
26 print(A)
27
28 >>> [[0.33211976 0.66788024 0.          0.          ]
```

```

25     [0.         0.         0.13974717 0.86025283]
26     [0.         0.         0.29698601 0.70301399]
27     [0.34046515 0.65953485 0.         0.         ]]
28
29     for i in range(4):
30         B[i] /= np.sum(B[i])
31
32     from hmmlearn import hmm
33     model = hmm.MultinomialHMM(n_components=4)
34     model.startprob_ = pi
35     model.emissionprob_ = B
36     model.transmat_ = A
37
38     test_str = "中国首次火星探测任务天问一号探测器实施近火捕获制动"
39     test_data = []
40     for i in range(len(test_str)): # 得到编码
41         test_data.append(ord(test_str[i]))
42     test_data = np.array(test_data).reshape(-1, 1)
43     states = model.predict(test_data)
44     print(states)
45     >>> [1 3 1 3 1 3 1 3 1 3 0 1 2 3 1 2 3 1 3 1 3 1 3 1 3]
46
47     test_out = ""
48     for i in range(len(states)):
49         test_out += test_str[i]
50         if states[i] == 0 or states[i] == 3:
51             test_out += ' '
52     test_out = test_out.strip()
53     print(test_out)
54     >>>中国 首次 火星 探测 任务 天 问一号 探测器 实施 近火 捕获 制动

```

第 13 行 0 初始化观测概率矩阵。这里仅限于对 GB2312 汉字编码字符集中的汉字进行统计。

第 32 行从 `hmmlearn` 扩展库中导入 `hmm` 隐马尔可夫模型模块。

第 33 行到第 36 行，先序列化一个多项式隐马尔可夫模型，然后将它的三要素分别赋值。

第 43 行是用该模型对测试语句进行解码，得到隐状态序列如第 45 行所示，它的值对应第 2 行定义的各种状态。

最后根据预测状态对测试语句进行间隔处理，方便直接观看。

6.3. CRF++工具建模及预测

CRF++工具是一个简单、可定制、开源的条件随机场工具，可用于序列数据的标注任务，广泛应用于自然语言处理任务中。CRF++用 C++语言实现，在 Windows 平台上，使用工具包中的 `crf_learn.exe`、`crf_test.exe`、`libcrfpp.dll` 三个文件即可完成模型的训练和预测。

在使用工具之前，要将训练语料和测试语句转换成符合 CRF++要求的格式，如下表所示。

col row	0	1
0	要	S
1	坚	B
2	持	E
3	以	B
4	人	M
5	为	M
6	本	E

实现代码如代码 6-3 所示。

代码 6-3 CRF++中文分词数据预处理（中文分词示例.ipynb）

```
1 # 将训练语料改成 crf++的格式，并写入文件 crf_train_file
2 crf_train_file = "crf_train_file"
3 output_file = open(crf_train_file, 'w', encoding='utf-8')
4 for i in range(len(new_sents)):
5     for j in range(len(new_sents[i][0])):
6         output_file.write(new_sents[i][0][j] + ' ' + sents_labels[i][0][j]
7 + '\n')
8     output_file.write('\n')
9 output_file.close()
10
11 # 将测试文本改成 crf++的格式，并写入文件 crf_test_file
12 crf_test_file = "crf_test_file"
13 output_file = open(crf_test_file, 'w', encoding='utf-8')
14 for i in range(len(test_str)):
15     output_file.write(test_str[i] + '\n')
16 output_file.close()
```

将 `crf_learn.exe`、`crf_test.exe`、`libcrfpp.dll` 文件拷贝到工作目录下，定义一个模板文件“template”。

在控制台环境下，执行“`crf_learn template crf_train_file crf_model`”命令进行训练，得到模型文件“`crf_model`”。

在控制台环境下，执行“`crf_test -m crf_model crf_test_file > crf_test_output`”命令得到测试语句的输出文件“`crf_test_output`”。

用 `python` 程序将测试语句的分词输出转换成习惯的词间隔的形式，如代码 6-4 所示。

代码 6-4 CRF++中文分词数据后处理（中文分词示例.ipynb）

```
1 # 将测试语句的分词输出改写方便观看的格式。
2 crf_test_output = "crf_test_output"
3 input_file = open(crf_test_output, encoding='utf-8')
4 str = ""
5 for line in input_file.readlines():
6     line = line.split()
7     if len(line) == 2:
8         if line[1] == 'E' or line[1] == 'S':
9             str += line[0] + ' '
10        else:
11            str += line[0]
12    input_file.close()
13    print(str)
14    >>>中国 首次 火星 探测 任务 天问 一 号 探测器 实施 近火 捕获 制动
```

从第 14 行的输出可见，CRF++能够对测试语句进行较好的分词，对专有名词“天问一号”的处理似乎比 `hmmlearn` 更合理。当然，这里仅是示例应用，这种比较并不全面。

6.4. 循环神经网络建模及预测

主要可分为四步，TensorFlow2 框架下的实现代码见代码 6-5。

1) 提取训练语料中的所有字，形成字典

该步的主要目的是给训练语料中用到的字进行编号，见第 13 行到第 23 行。

2) 将语料中的句子转化为训练样本

模型对每个输入训练样本的长度要求一致，因此，可以指定一个固定长度，过长的句子应截断后面过长的部分。过短的句子在后面填充 0，并指定一个新的标签“X”与之对应。通过字典将句子的汉字序列转换为数字序列。标签用独热编码表示。

3) 搭建模型进行训练

采用深度双向循环神经网络模型，见第 52 行到第 59 行。

第 53 行是词向量层。所谓词向量可以简单地理解为用指定维度的向量来对汉字进行编码。在本示例中，用一个的 64 维向量来表示一个汉字。词向量各维度的值是要学习的参数，

假设有 4684 个汉字，每个汉字用 64 维的向量来表示，则词向量层需要学习的参数共 $4684 \times 64 = 299776$ 个。词向量在自然语言处理领域是很重要的技术，得到了广泛应用。

第 54 行是双向 LSTM 层，第 55 行是 Dropout 层。第 56 行和第 57 行再增加一层双向 LSTM 和 Dropout。第 58 行是对每步都输出到一个全连接层，全连接层的输出是独热码表示的预测标签值。

4) 利用训练好的模型进行分词

先将待分词的句子转换成适合模型输入的形式，再用模型进行分词。

对“中国首次火星探测任务天问一号探测器实施近火捕获制动”一句的分词结果为：“中国 首次 火星 探测 任务 天问 一 号 探测器 实施 近 火 捕 获 制动”。

代码 6-5 TensorFlow2 框架下实现中文分词示例（中文分词示例.ipynb）

```
1 import re
2
3 # 重要参数
4 tags = {'S': 0, 'B': 1, 'M': 2, 'E': 3, 'X': 4} # 标签
5 embedding_size = 32 # 词向量大小
6 maxlen = 32 # 序列长度，长于则截断，短于则填充 0
7 hidden_size = 32
8 batch_size = 64
9 epochs = 1
10 checkpointfilepath = 'weights.best.hdf5' # 中间结果保存文件
11 modelpath = 'dz.h5' # 模型保存文件
12
13 # 1.提取出所有用到的字，形成字典
14 stat = {}
15 for i in range(len(new_sents)):
16     for v in new_sents[i][0]:
17         stat[v] = stat.get(v, 0) + 1
18 stat = sorted(stat.items(), key=lambda x:x[1], reverse=True)
19 vocab = [s[0] for s in stat]
20 print("不同字的个数: " + str(len(vocab)))
21 char2id = {c : i + 1 for i, c in enumerate(vocab)} # 编号 0 为填充值，因此从 1 开始编号
22 id2char = {i + 1 : c for i, c in enumerate(vocab)}
23 print("字典创建完毕!")
24 >>>不同字的个数: 3878
25 字典创建完毕!
26
27 # 2.将训练语句转化为训练样本
28 trainX = []
```

```

29 trainY = []
30 for i in range(len(new_sents)):
31     x = [0] * maxlen # 默认填充值
32     y = [4] * maxlen # 默认标签 x
33     sent = new_sents[i][0]
34     labe = sents_labels[i][0]
35     replace_len = len(sent)
36     if len(sent) > maxlen:
37         replace_len = maxlen
38     for j in range(replace_len):
39         x[j] = char2id[sent[j]]
40         y[j] = tags[labe[j]]
41     trainX.append(x)
42     trainY.append(y)
43 trainX = np.array(trainX)
44 trainY = tf.keras.utils.to_categorical(trainY, 5)
45 print("训练样本准备完毕，训练样本共" + str(len(trainX)) + "句。")
46 >>>训练样本准备完毕，训练样本共 62947 句。
47
48 # 3.搭建模型，并训练
49 from tensorflow.keras.layers import Input, Dense, Embedding, LSTM, Dropout,
TimeDistributed, Bidirectional
50 from tensorflow.keras.models import Model
51 from tensorflow.keras.callbacks import ModelCheckpoint
52 X = Input(shape=(maxlen,), dtype='int32')
53 embedding = Embedding(input_dim=len(vocab)+1, output_dim=embedding_size,
input_length=maxlen, mask_zero=True)(X)
54 blstm = Bidirectional(LSTM(hidden_size, return_sequences=True),
merge_mode='concat')(embedding)
55 blstm = Dropout(0.4)(blstm)
56 blstm = Bidirectional(LSTM(hidden_size, return_sequences=True),
merge_mode='concat')(blstm)
57 blstm = Dropout(0.4)(blstm)
58 output = TimeDistributed(Dense(5, activation='softmax'))(blstm)
59 model = Model(X, output)
60 model.summary()
61 >>> Model: "model_2"
62
63 _____
64 Layer (type)                Output Shape                Param #
65 =====
66 input_3 (InputLayer)        [(None, 32)]                0
67 _____
68 embedding_2 (Embedding)      (None, 32, 32)              124128
69 _____

```

```

69     bidirectional_4 (Bidirection (None, 32, 64)          16640
70     _____
71     dropout_4 (Dropout) (None, 32, 64)          0
72     _____
73     bidirectional_5 (Bidirection (None, 32, 64)          24832
74     _____
75     dropout_5 (Dropout) (None, 32, 64)          0
76     _____
77     time_distributed_2 (TimeDist (None, 32, 5)          325
78     =====
79     Total params: 165,925
80     Trainable params: 165,925
81     Non-trainable params: 0
82
83     import os
84     if os.path.exists(checkpointfilepath): # 与下面的 checkpoint 起到及时保存训练结果的作用
85         print("加载前次训练模型参数。。。")
86         model.load_weights(checkpointfilepath)
87     model.compile(loss='categorical_crossentropy', optimizer='adam',
88     metrics=['accuracy'])
89     checkpoint = ModelCheckpoint(checkpointfilepath, monitor='acc',
90     verbose=1, save_best_only=True,
91     mode='max')
92     model.fit(trainX, trainY, batch_size=batch_size, epochs=epochs,
93     callbacks=[checkpoint])
94     model.save(modepath)
95     #print(model.evaluate(trainX, trainY, batch_size=batch_size))
96     >>> Train on 62947 samples
97     62912/62947 [=====>.] - ETA: 0s - loss: 0.1944 -
98     accuracy: 0.7093 - ETA: 57s - loss: 0.2171 - - - ETA: 30s - loss: 0 -
99     ETA: 24s - loss: 0.2034 - accuracy: 0 - ETA: - - ETA: - ETA: 2s - loss:
100     0.1952 WARNING:tensorflow:Can save best model only with acc available,
101     skipping.
102     62947/62947 [=====] - 241s 4ms/sample - loss:
103     0.1943 - accuracy: 0.7094
104
105     # 4.利用训练好的模型进行分词
106     def predict(testsent):
107         # 将汉字句子转换成模型需要的输入形式
108         x = [0] * maxlen
109         replace_len = len(testsent)
110         if len(testsent) > maxlen:
111             replace_len = maxlen

```

```
104     for j in range(replace_len):
105         x[j] = char2id[testsent[j]]
106     # 调用模型进行预测
107     label = model.predict([x])
108     # 根据模型预测结果对输入句子进行切分
109     label = np.array(label)[0]
110     s = ''
111     for i in range(len(testsent)):
112         tag = np.argmax(label[i])
113         if tag == 0 or tag == 3: # 单字和词结尾加空格切分
114             s += testsent[i] + ' '
115         elif tag == 1 or tag == 2:
116             s += testsent[i]
117     print(s)
118 predict(test_str)
119 >>> 中国 首次 火星 探测 任务 天问 一 号 探测器 实施 近 火 捕 获 制 动
```