# 实验一：Python语言简介及回归分类实验

任务：

1. 熟悉Python编程环境
2. 熟悉数据集，了解基本对数据集的统计和可视化操作；
3. 完成对波士顿房价数据的回归预测，并进行统计学分析；
4. 完成对手写体数字图片的分类和分析。

## Ⅰ Python语言简介

Python语言主页：<https://www.python.org/>. 主页有大家想知道的所有关于Python的知识，强烈推荐需要掌握Python语言的同学仔细学习。这里能找到最权威最详细的资料。包括这篇指导书也是从里面精炼出来，包含一部分主页的类容，仅作抛砖引玉之作用，更多的请前往主页查找。

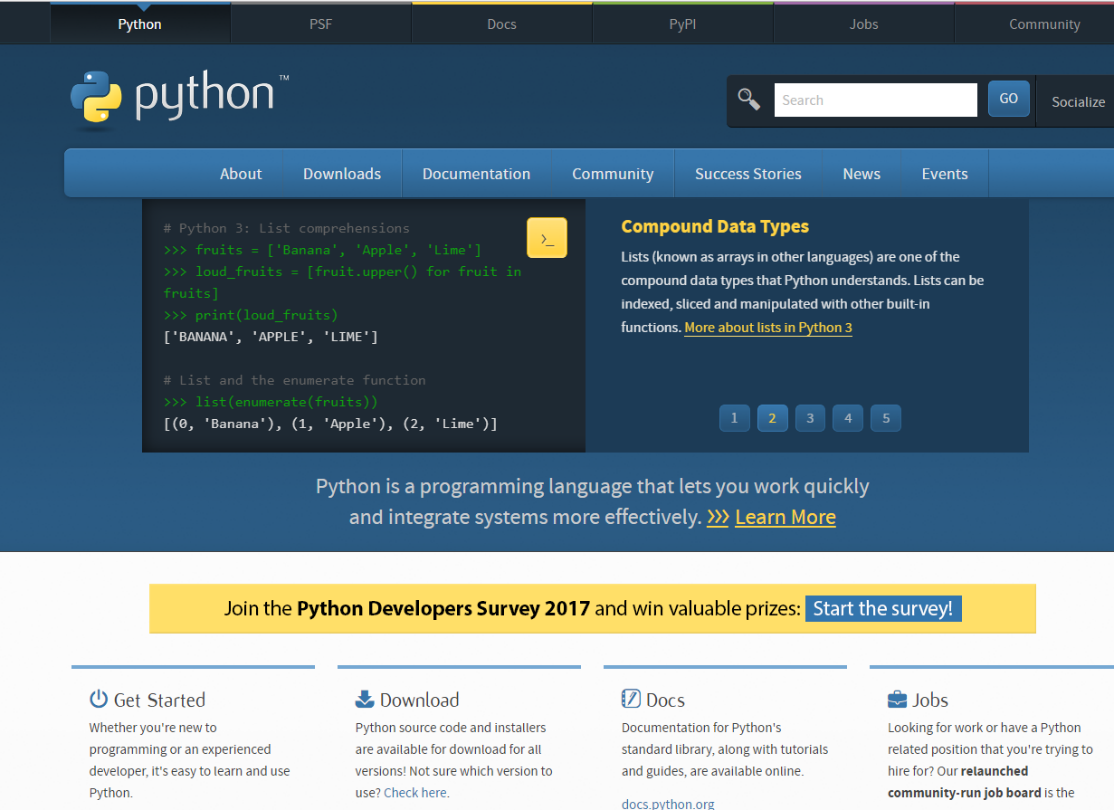


图1 Python语言主页

Python is a clear and powerful object-oriented programming language, comparable to Perl, Ruby, Scheme, or Java.

Some of Python's notable features:

* Uses an elegant syntax, making the programs you write easier to read.
* Is an easy-to-use language that makes it simple to get your program working. This makes Python ideal for prototype development and other ad-hoc programming tasks, without compromising maintainability.
* Comes with a large standard library that supports many common programming tasks such as connecting to web servers, searching text with regular expressions, reading and modifying files.
* Python's interactive mode makes it easy to test short snippets of code. There's also a bundled development environment called IDLE.
* Is easily extended by adding new modules implemented in a compiled language such as C or C++.
* Can also be embedded into an application to provide a programmable interface.
* Runs anywhere, including [Mac OS X](https://www.python.org/downloads/mac-osx/), [Windows](https://www.python.org/downloads/windows/), [Linux](https://docs.python.org/3/using/unix.html), and [Unix.](https://docs.python.org/3/using/unix.html)
* Is free software in two senses. It doesn't cost anything to download or use Python, or to include it in your application. Python can also be freely modified and re-distributed, because while the language is copyrighted it's available under [an open source license](http://www.python.org/psf/license/).

Some programming-language features of Python are:

* A variety of basic data types are available: numbers (floating point, complex, and unlimited-length long integers), strings (both ASCII and Unicode), lists, and dictionaries.
* Python supports object-oriented programming with classes and multiple inheritance.
* Code can be grouped into modules and packages.
* The language supports raising and catching exceptions, resulting in cleaner error handling.
* Data types are strongly and dynamically typed. Mixing incompatible types (e.g. attempting to add a string and a number) causes an exception to be raised, so errors are caught sooner.
* Python contains advanced programming features such as generators and list comprehensions.
* Python's automatic memory management frees you from having to manually allocate and free memory in your code.

**为什么选择Python？**

**简单回答：好写、支持全面、好调、速度不慢。**

1. Python是解释语言，容易上手。不需要编译，这对机器学习这种需要大量算法设计和迭代的研究方向是非常有益工作效率的。对比于matlab，Python免费可以未来工业应用时无缝对接。

2. Python的开发生态成熟，有很多非常有用的库可以用。如数值计数的NumPy，还有SciPy、NLTK、os（自带）等等不一而足。机器学习相关的库非常非常多！这对做机器学习的人很重要。因为经常需要对模型进行各种各样的修改，这在编译语言里很可能是牵一发而动全身的事情，Python里通常可以用很少的时间实现。

3. Python的效率不差。解释语言的发展已经大大超过许多人的想象，得益于Python对C的接口，很多像gnumpy, theano这样高效、Python接口友好的库可以加速程序的运行，在强大团队的支撑下，这些库的效率可能比一个不熟练的程序员用C写一个月调优的效率还要高。

**Compiled languages: C, C++, Fortran. . .**

**Pros**

• Very fast. For heavy computations, it’s difficult to outperform these languages.

**Cons**

• Painful usage: no interactivity during development, mandatory compilation steps, verbose syntax, manual memory management. These are **difficult languages** for non programmers.

**Matlab scripting language**

**Pros**

• Very rich collection of libraries with numerous algorithms, for many different domains. Fast execution because these libraries are often written in a compiled language.

• Pleasant development environment: comprehensive and help, integrated editor, etc.

• Commercial support is available.

**Cons**

• Base language is quite poor and can become restrictive for advanced users.

• Not free.

**Julia**

**Pros**

• Fast code, yet interactive and simple.

• Easily connects to Python or C.

**Cons**

• Ecosystem limited to numerical computing.

• Still young.

**Other scripting languages: Scilab, Octave, R, IDL, etc.**

**Pros**

• Open-source, free, or at least cheaper than Matlab.

• Some features can be very advanced (statistics in R, etc.)

**Cons**

• Fewer available algorithms than in Matlab, and the language is not more advanced.

• Some software are dedicated to one domain. Ex: Gnuplot to draw curves. These programs

are very powerful, but they are restricted to a single type of usage, such as plotting.

**Python**

**Pros**

• Very rich scientific computing libraries

• Well thought out language, allowing to write very readable and well structured code: we

“code what we think”.

• Many libraries beyond scientific computing (web server, serial port access, etc.)

• Free and open-source software, widely spread, with a vibrant community.

• A variety of powerful environments to work in, such as IPython, Spyder, Jupyter notebooks, Pycharm

**Cons**

• Not all the algorithms that can be found inmore specialized software or toolboxes

**Should I use Python 2 or Python 3 for my development activity?**

What are the differences?

*Short version: Python 2.x is legacy, Python 3.x is the present and future of the language*

Python 3.0 was released in 2008. The final 2.x version 2.7 release came out in mid-2010, with a statement of extended support for this end-of-life release. The 2.x branch will see no new major releases after that. 3.x is under active development and has already seen over five years of stable releases, including version 3.3 in 2012, 3.4 in 2014, 3.5 in 2015, and 3.6 in 2016. This means that all recent standard library improvements, for example, are only available by default in Python 3.x.

Guido van Rossum (the original creator of the Python language) decided to clean up Python 2.x properly, with less regard for backwards compatibility than is the case for new releases in the 2.x range. The most drastic improvement is the better Unicode support (with all text strings being Unicode by default) as well as saner bytes/Unicode separation.

Besides, several aspects of the core language (such as print and exec being statements, integers using floor division) have been adjusted to be easier for newcomers to learn and to be more consistent with the rest of the language, and old cruft has been removed (for example, all classes are now new-style, "range()" returns a memory efficient iterable, not a list as in 2.x).

The [What's New in Python 3.0](http://docs.python.org/py3k/whatsnew/3.0.html) document provides a good overview of the major language changes and likely sources of incompatibility with existing Python 2.x code. Nick Coghlan (one of the CPython core developers) has also created a [relatively extensive FAQ](http://python-notes.curiousefficiency.org/en/latest/python3/questions_and_answers.html) regarding the transition.

However, the broader Python ecosystem has amassed a significant amount of quality software over the years. The downside of breaking backwards compatibility in 3.x is that some of that software (especially in-house software in companies) still doesn't work on 3.x yet.

Which version should I use?

Which version you ought to use is mostly dependent on what you want to get done.

If you can do exactly what you want with Python 3.x, great! There are a few minor downsides, such as very slightly worse library support[1](https://wiki.python.org/moin/Python2orPython3#fnref-4fcb00922956f144dafd52e4f75d82631e225d1b) and the fact that some current Linux distributions and Macs still use 2.x as default, but as a language Python 3.x is definitely mature and ready for use. As long as Python 3.x is installed on your user's computers (which ought to be easy, since many people reading this may only be developing something for themselves or an environment they control) and you are writing things where you know none of the Python 2.x modules are needed, it is an excellent choice. Also, most Linux distributions have Python 3.x already installed, and nearly all have it available for end-users. One somewhat painful exception is Red Hat Enterprise Linux through version 7: Python 3 does exist in the EPEL repository, but some users may not be allowed by company policy to install anything from such add-on locations - see note on "control" below. Some distributions are phasing out Python 2 as preinstalled default.[2](https://wiki.python.org/moin/Python2orPython3#fnref-729f005f65eb8709309aad20cf8a9911eb532aa9)

In particular, instructors introducing Python to new programmers should consider teaching Python 3 first and then introducing the differences in Python 2 afterwards (if necessary), since Python 3 [eliminates many quirks](http://python-notes.curiousefficiency.org/en/latest/python3/questions_and_answers.html#why-is-python-3-considered-a-better-language-to-teach-beginning-programmers) that can unnecessarily trip up beginning programmers trying to learn Python 2.

However, there are some key issues that may require you to use Python 2 rather than Python 3.

* Firstly, if you are deploying to an environment you don't control, that may require use of a specific version, rather than allowing you a free selection from the available versions.
* Secondly, if you want to use a specific third party package or utility that doesn't yet have a released version that is compatible with Python 3, and porting that package is a non-trivial task, you may choose to use Python 2 in order to retain access to that package.

## Ⅱ Python安装和环境配置

**Windows：**

1、下载安装包

<https://www.python.org/downloads/> （根据自己电脑配置选择32位或者64位）

2、安装

默认安装路径：C:\python27

3、配置环境变量

【右键计算机】--》【属性】--》【高级系统设置】--》【高级】--》【环境变量】--》【在第二个内容框中找到 变量名为Path 的一行，双击】 --> 【Python安装目录追加到变值值中，用 ； 分割】

**Linux（Ubuntu）：**

无需安装，原装Python环境

Python版本更新：

**Windows：**

下载新版覆盖安装

**Linux（Ubuntu）：**

查看默认Python版本

python -V

1、安装gcc，用于编译Python源码

    yum install gcc

2、下载源码包，https://www.python.org/ftp/python/

3、解压并进入源码文件

4、编译安装

    ./configure

    make all

    make install

5、查看版本

    /usr/local/bin/python3.\* -V

6、修改默认Python版本

    mv /usr/bin/python /usr/bin/python2.\*

    ln -s /usr/local/bin/python3.\* /usr/bin/python

7、防止yum执行异常，修改yum使用的Python版本

    vi /usr/bin/yum

将头部 #!/usr/bin/python 修改为 #!/usr/bin/python3.\*

Python的包管理工具pip的安装与使用

**Windows：**

下载pip的安装包get-pip.py，下载地址：[https://pip.pypa.io/en/latest/installing.html#id7](https://pip.pypa.io/en/latest/installing.html" \l "id7)

**Linux（Ubuntu）：**

sudo apt-get install python-pip

Pip 更新

**Windows：**

python -m pip install -U pip

**Linux（Ubuntu）：**

pip install -U pip

使用：

pip install SomePackage

交互计算环境：

* **IPython**, an advanced **Python console**<http://ipython.org/>
* **Jupyter**, **notebooks** in the browser <http://jupyter.org/>
* Anaconda （<https://www.anaconda.com/download/>）
* <https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive/>
* WinPython（<https://winpython.github.io/>）
* Spyder
* Pycharm 收费，比较好用一款IDE。 建议用免费community版

常用的库：

* **pandas, statsmodels, seaborn** for [statistics](http://www.scipy-lectures.org/packages/statistics/index.html#statistics)
* **sympy** for [symbolic computing](http://www.scipy-lectures.org/packages/sympy.html#sympy)
* **scikit-image** for [image processing](http://www.scipy-lectures.org/packages/scikit-image/index.html#scikit-image)
* **scikit-learn** for [machine learning](http://www.scipy-lectures.org/packages/scikit-learn/index.html#scikit-learn-chapter)

## 数据集

数据集可参考： [https://archive.ics.uci.edu/ml/](http://rrurl.cn/nkMA8l)， scikit-learn是Python的一个开源机器学习模块（<http://scikit-learn.org/dev/>）

安装完毕后，通过sklearn.datasets可调用里面集成了这个网站里的部分数据，　scikit-learn载入数据集实例：

方法1：

from sklearn import datasets

iris = datasets.load\_iris() # 鸢尾花数据， 较有名的分类数据集

print iris.data

方法2：

import sklearn.datasets

boston = sklearn.datasets.load\_boston()

print boston.DESCR #查询具体数据说明

1、**Boston house prices**

波士顿房屋价格数据。 一共506组数据，13维特征，比如第一个维度的特征是犯罪率，第六个是每个房子平均多少房间等等。

boston.data 获取这506 \* 13的特征数据；

boston.target 获取对应的506 \* 1的对应价格

- 'RM' 是该地区中每个房屋的平均房间数量；

- 'LSTAT' 是指该地区有多少百分比的房东属于是低收入阶层（有工作但收入微薄）；

- 'PTRATIO' 是该地区的中学和小学里，学生和老师的数目比（学生/老师）。

2、**The digits dataset**

手写数字识别数据集。 （多类分类，10个类别，从0-9）digits。

总体样本量：1797，每个类别大约180个样本，每个手写数字是一个8\*8的图片，每个像素是0-16的整数值。

大家可以借助下列链接开始学习：

https://www.datacamp.com/community/tutorials/machine-learning-python#gs.vsrEMn4

## Ⅲ 实验

### 线性回归

#### 简单线性回归

用Boston数据集进行简单线性回归。

#### 库

sklearn.datasets, sklearn.linear\_model, numpy（ numpy.random，numpy.linalg ）lm(), matplotlib

#### 要求及步骤

1. 划分数据集，分训练集和测试集； 用sklearn.linear\_model.LinearRegression()完成一个简单线性回归，了解预测变量和响应变量之间的关系，关系强弱，正负相关性。
2. 绘制响应变量和预测变量关系图，绘制最小二乘回归线。
3. 使用LinearRegression模型自带的评估模块，并输出评估结果。

from sklearn.metrics import r2\_score,mean\_squared\_error,mean\_absolute\_error

使用线性回归模型LinearRegression和SGDRegressor分别对波士顿房价数据进行训练及预测，给出评估结果

#### http://upload-images.jianshu.io/upload_images/623192-a192b60b51933598.jpg?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240

#### 多元线性回归

用Boston房价数据集进行多元线性回归。

#### 要求及步骤

1. 作出数据集中的所有变量的散点图矩阵； matplotlib.pyplot.scatter(a,b)
2. 计算变量之间的相关系数矩阵。
3. 用polynominalData = sklearn.preprocessing.PolynomialFeatures(degree=2).fit\_transform(boston.data)进行多元线性回归，给出性能评估。
4. 进行交叉验证分析。可参考：

http://scikit-learn.org/stable/auto\_examples/model\_selection/plot\_cv\_predict.html#sphx-glr-auto-examples-model-selection-plot-cv-predict-py

1. 随着通过划分比例观察训练数据量的增加，对训练的性能评分的变化和测试评分的变化？绘制一个曲线。

#### 岭回归与Lasso回归 （参考下面链接掌握这部分内容）

用Boston房价数据集进行岭回归和Lasso回归。

#### 1）要求及步骤

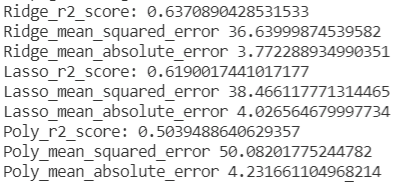
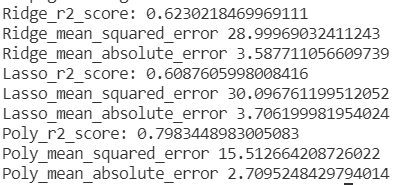
a)分别用sklearn.linear\_model. Ridge ()和sklearn.linear\_model. Lasso()实现岭回归和Lasso回归进行模型训练，进一步理解数据，分析不同输入特征与输出变量之间的关系强弱和相关性。

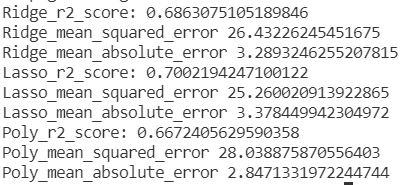
b) 在测试集上完成预测，并输出评估结果，与一般的多元线性回归的结果进行对比。

c) 改变岭回归和Lasso回归中的参数alpha的值，绘制回归系数随alpha的变化图，观察预测效果的变化和不同输入变量对预测结果的影响力。

可参考：<https://www.cnblogs.com/magle/p/5878967.html>

<https://zhuanlan.zhihu.com/p/165493873>





### 分类

根据手写体数据集，熟悉如何对图像进行分类。

#### 要求及步骤

1. 认识数据集 （可视化数据）

# Import matplotlib

import matplotlib.pyplot as plt

# Join the images and target labels in a list

images\_and\_labels = list(zip(digits.images, digits.target))

# for every element in the list

for index, (image, label) in enumerate(images\_and\_labels[:8]):

# initialize a subplot of 2X4 at the i+1-th position

plt.subplot(2, 4, index + 1)

# Don't plot any axes

plt.axis('off')

# Display images in all subplots

plt.imshow(image, cmap=plt.cm.gray\_r,interpolation='nearest')

# Add a title to each subplot

plt.title('Training: ' + str(label))

# Show the plot

plt.show()

1. 参考

<https://github.com/htygithub/machine-learning-python/blob/master/Classification/ex1_Recognizing_hand-written_digits.md>

运行：<http://scikit-learn.org/stable/_downloads/plot_digits_classification.py>

给出注释。

1. 用KNN分类模型，对手写体数据集进行识别。讨论k变化时分类性能变化。
2. 用SVM(课堂没讲，请直接调用Scikit-learn模块)分类模型，对比与最好KNN模型的性能好坏。

参考

http://scikit-learn.org/stable/auto\_examples/classification/plot\_classifier\_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py