



# 数据科学理论基础：Python基础

北京邮电大学

智能感知与计算教研中心

数据科学中心

# Python简要介绍

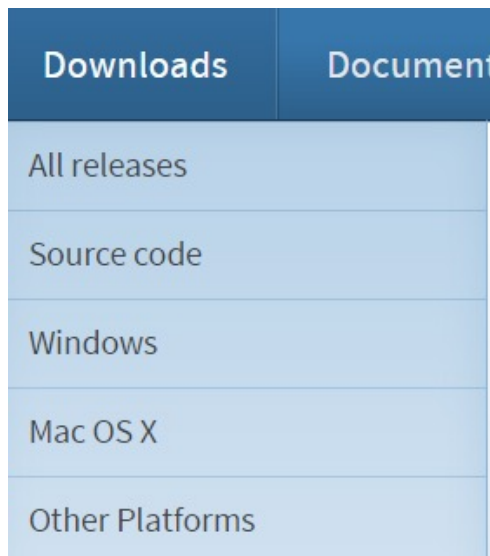
# Python简要介绍

---

NumPy      Sklearn  
OpenCV      Matplotlib  
PyTorch      TensorFlow  
.....



# Python简要介绍



## 开源

Linux, Windows, Macintosh  
pip, Anaconda, Virtualenv  
模块化管理库。

```
class Employee:
    '所有员工的基类'
    empCount = 0

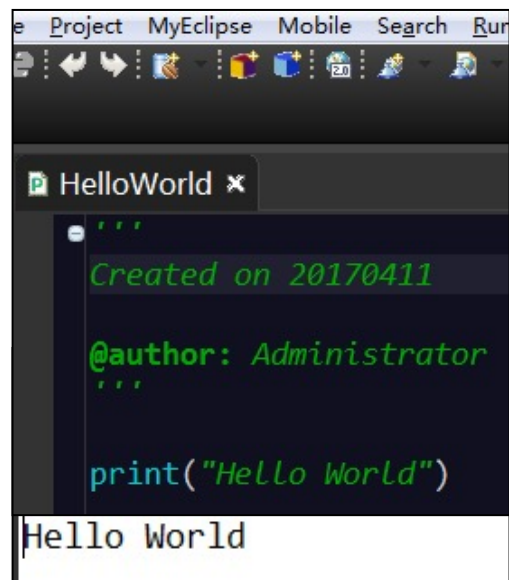
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    def displayCount(self):
        print "Total Employee %d" % Employee.empCount

    def displayEmployee(self):
        print "Name : ", self.name, " , Salary: ", self.salary
```

## 面向对象

数据和函数组成类  
继承、重载、方法重写



## 解释性

解释器把源代码转换成为字节码  
再翻译成机器语言运行

# Python简要介绍



<https://www.python.org/downloads/>

安装可参考: [https://blog.csdn.net/qq\\_39313596/article/details/80664945](https://blog.csdn.net/qq_39313596/article/details/80664945)



# Python简要介绍

```
选择管理员: 命令提示符 - python
Microsoft Windows [版本 10.0.17134.1006]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\ICY>python
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 11:03:09)
Type "help", "copyright", "credits" or "license()"
>>> print("Hello World!")
Hello World!
>>> 1+1
2
>>> a = "asd" + "dsa"
>>> a
'asddsa'
>>> b = [1, 2, 3, 4, "优于", "C++", "python"]
>>> b[0]
1
>>> b[6]+b[4]+b[5]
'python优于C++'
>>>
```

```
H:\hello.py - Notepad++ [Admini...
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T)
工具(O) 宏(M) 运行(R) 插件(P) 窗口(W) ?
X

a x OS自动化安装部署.txt pytorch.txt
1 print("Hello World!!")

C:\Users\ICY>H:
H:\>python hello.py
Hello World!!
```

```
Run (Shift+F10)
demo.py x proposal_layer.py x text_proposal_connector.py x crnn.py x model_train.py x
1 # coding:utf-8
2 import ...
6
7 if __name__ == '__main__':
8
9     # im = Image.open("test/ttttt.png")
10    # img = np.array(im.convert('RGB'))
11    img = "test/ttttt.png"
12    t = time.time()
13
14    # result,img,angel 分别对应- 识别结果, 图像的数组, 文字旋转角度
15    result = model.model(imgname=img, adjust=False)
16    # use ctpn+tensorflow to detect the edges of text segments,
17    print("Totally it takes time:{}s".format(time.time() - t))
18    print("-----")
19    for key in result:
20        print(result[key][1])
21
```

Windows: 命令指示符 >> cmd

在cmd中运行.py文件

IDE: 配置环境后使用,  
例如Pycharm

# Anaconda 安装

Anaconda指的是一个开源的Python发行版本，其包含了conda、Python等180多个科学包及其依赖项，比如：numpy、pandas等



Products ▼

Pricing

Solutions ▼

Resources ▼

Blog

Company ▼

Get Started

## Anaconda Installers

### Windows

Python 3.8

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (397 MB)

### MacOS

Python 3.8

64-Bit Graphical Installer (462 MB)

64-Bit Command Line Installer (454 MB)

### Linux

Python 3.8

64-Bit (x86) Installer (550 MB)

64-Bit (Power8 and Power9) Installer (290 MB)

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

# Anaconda 安装

---

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



清华大学开源软件镜像站

|  |           |                  |
|--|-----------|------------------|
| <a href="#">Anaconda3-5.3.1-Linux-x86.sh</a>       | 527.3 MiB | 2018-11-20 04:00 |
| <a href="#">Anaconda3-5.3.1-Linux-x86_64.sh</a>    | 637.0 MiB | 2018-11-20 04:00 |
| <a href="#">Anaconda3-5.3.1-MacOSX-x86_64.pkg</a>  | 634.0 MiB | 2018-11-20 04:00 |
| <a href="#">Anaconda3-5.3.1-MacOSX-x86_64.sh</a>   | 543.7 MiB | 2018-11-20 04:01 |
| <a href="#">Anaconda3-5.3.1-Windows-x86.exe</a>    | 509.5 MiB | 2018-11-20 04:04 |
| <a href="#">Anaconda3-5.3.1-Windows-x86_64.exe</a> | 632.5 MiB | 2018-11-20 04:04 |



# Anaconda 安装


Anaconda Navigator


File Help


 ANACONDA NAVIGATOR

[Sign in to Anaconda Cloud](#)

 Home

 Environments

 Learning

 Community

[Documentation](#)

[Developer Blog](#)



Applications on

[Channels](#)

[Refresh](#)



JupyterLab

0.34.9

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

[Launch](#)



Jupyter Notebook

5.6.0

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

[Launch](#)



Spyder

3.3.1

Scientific PYTHON Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

[Launch](#)



VS Code

Streamlined code editor with support for development operations like debugging, task running and version control.

[Install](#)

Updating package index and metadata...

# Python —— Numpy

- NumPy(Numerical Python) 是 Python 语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。

## 创建数组

```
In [47]: import numpy as np
```

```
In [50]: x = np.arange(8)
```

```
In [51]: print(x)
```

```
[0 1 2 3 4 5 6 7]
```

## 修改数组形状

```
In [54]: a = np.arange(12)
print ('原始数组: ', a)
print ('\n')
```

```
原始数组: [ 0  1  2  3  4  5  6  7  8  9 10 11]
```

```
In [55]: b = a.reshape(3,4)
print ('修改后的数组: ')
print (b)
```

```
修改后的数组:
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

# 回归分析的典型应用

---

**线性回归LinearRegression分析波士顿房价数据**

# 回归分析的典型应用

---

回归模型在 `Sklearn.linear_model` 子类下，调用sklearn逻辑回归算法步骤主要有3步，即：

- (1) 导入模型。调用逻辑回归`LogisticRegression()`函数。
- (2) `fit()`训练。调用`fit(x,y)`的方法来训练模型，其中x为数据的属性，y为所属类型
- (3) `predict()`预测。利用训练得到的模型对数据集进行预测，返回预测结果。

# 回归分析的典型应用

---

在Sklearn机器学习包中，集成了各种各样的数据集：

```
>> from sklearn.datasets import load_boston, xxx...
```

scikit-learn comes with a few small standard datasets that do not require to download any file from some external website.

They can be loaded using the following functions:

|  |   |
|--|---|
| <code>load_boston</code> ([return_X_y])          | Load and return the boston house-prices dataset (regression).         |
| <code>load_iris</code> ([return_X_y])            | Load and return the iris dataset (classification).                    |
| <code>load_diabetes</code> ([return_X_y])        | Load and return the diabetes dataset (regression).                    |
| <code>load_digits</code> ([n_class, return_X_y]) | Load and return the digits dataset (classification).                  |
| <code>load_linnerud</code> ([return_X_y])        | Load and return the linnerud dataset (multivariate regression).       |
| <code>load_wine</code> ([return_X_y])            | Load and return the wine dataset (classification).                    |
| <code>load_breast_cancer</code> ([return_X_y])   | Load and return the breast cancer wisconsin dataset (classification). |

These datasets are useful to quickly illustrate the behavior of the various algorithms implemented in scikit-learn. They are however often too small to be representative of real world machine learning tasks.



# 回归分析的典型应用——波士顿房价数据集

| 特征名称                                   | feature_names   | 特征名称               | feature_names  |
|--|---|--------------------|--|
| CRIM ( 地区人均犯罪率 )                       | per capita crime rate by town   | DIS ( 与波士顿中心区距离 )  | weighted distances to five Boston employment centres                       |
| ZN ( 住宅用地>25000英尺比例 )                  | proportion of residential land zoned for lots over 25,000 sq.ft.      | RAD ( 与主要公路的接近指数 ) | index of accessibility to radial highways                                  |
| INDUS ( 非零售商业用地比例 )                    | proportion of non-retail business acres per town                      | TAX ( 财产税率 )       | full-value property-tax rate per \$10,000                                  |
| CHAS ( 查尔斯河空变量 ( 地区边界是河, 值取1, 否则为0 ) ) | Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) | PTRATIO ( 师生比 )    | pupil-teacher ratio by town  |
| NOX ( 一氧化氮浓度 )                         | nitric oxides concentration (parts per 10 million)                    | B ( 非洲裔美国人比例 )     | $1000(B_k - 0.63)^2$ where $B_k$ is the proportion of black people by town |
| RM ( 每套住宅平均房间数 )                       | average number of rooms per dwelling                                  | LSTAT ( 低地位人口比例 )  | % lower status of the population   |
| AGE ( 1940年后建成自用房比例 )                  | proportion of owner-occupied units built prior to 1940                | MEDV ( 自住房平均房价 )   | Median value of owner-occupied homes in \$1000's                           |

# 回归分析的典型应用

## 波士顿房价 ( load\_boston ) 数据集

- 13个基本变量
- 1个类别变量
- 506个实例数据

|   | CRIM    | ZN   | INDUS | CHAS | NOX   | RM    | AGE   | DIS    | RAD | TAX   | PTRATIO | B      | LSTAT | label |
|---|---------|------|-------|------|-------|-------|-------|--------|-----|-------|---------|--------|-------|-------|
| 0 | 0.00632 | 18.0 | 2.31  | 0.0  | 0.538 | 6.575 | 65.2  | 4.0900 | 1.0 | 296.0 | 15.3    | 396.90 | 4.98  | 24.0  |
| 1 | 0.02731 | 0.0  | 7.07  | 0.0  | 0.469 | 6.421 | 78.9  | 4.9671 | 2.0 | 242.0 | 17.8    | 396.90 | 9.14  | 21.6  |
| 2 | 0.02729 | 0.0  | 7.07  | 0.0  | 0.469 | 7.185 | 61.1  | 4.9671 | 2.0 | 242.0 | 17.8    | 392.83 | 4.03  | 34.7  |
| 3 | 0.03237 | 0.0  | 2.18  | 0.0  | 0.458 | 6.998 | 45.8  | 6.0622 | 3.0 | 222.0 | 18.7    | 394.63 | 2.94  | 33.4  |
| 4 | 0.06905 | 0.0  | 2.18  | 0.0  | 0.458 | 7.147 | 54.2  | 6.0622 | 3.0 | 222.0 | 18.7    | 396.90 | 5.33  | 36.2  |
| 5 | 0.02985 | 0.0  | 2.18  | 0.0  | 0.458 | 6.430 | 58.7  | 6.0622 | 3.0 | 222.0 | 18.7    | 394.12 | 5.21  | 28.7  |
| 6 | 0.08829 | 12.5 | 7.87  | 0.0  | 0.524 | 6.012 | 66.6  | 5.5605 | 5.0 | 311.0 | 15.2    | 395.60 | 12.43 | 22.9  |
| 7 | 0.14455 | 12.5 | 7.87  | 0.0  | 0.524 | 6.172 | 96.1  | 5.9505 | 5.0 | 311.0 | 15.2    | 396.90 | 19.15 | 27.1  |
| 8 | 0.21124 | 12.5 | 7.87  | 0.0  | 0.524 | 5.631 | 100.0 | 6.0821 | 5.0 | 311.0 | 15.2    | 386.63 | 29.93 | 16.5  |
| 9 | 0.17004 | 12.5 | 7.87  | 0.0  | 0.524 | 6.004 | 85.9  | 6.5921 | 5.0 | 311.0 | 15.2    | 386.71 | 17.10 | 18.9  |

# 回归分析的典型应用

## ● 如何获取基础信息：

✓ `boston['data'].shape` → (506, 13)

✓ `datasets.load_boston().keys()`  
→ ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM'  
'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO' 'B' 'LSTAT']

✓ `datasets.load_boston().DESCR`

Boston house prices dataset

-----  
\*\*Data Set Characteristics:\*\*

:Number of Instances: 506

:Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.

:Attribute Information (in order):

- CRIM per capita crime rate by town
- ZN proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS proportion of non-retail business acres per town
- CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- NOX nitric oxides concentration (parts per 10 million)
- RM average number of rooms per dwelling
- AGE proportion of owner-occupied units built prior to 1940
- DIS weighted distances to five Boston employment centres
- RAD index of accessibility to radial highways
- TAX full-value property-tax rate per \$10,000
- PTRATIO pupil-teacher ratio by town
- B  $1000(Bk - 0.63)^2$  where Bk is the proportion of black people by town
- LSTAT % lower status of the population
- MEDV Median value of owner-occupied homes in \$1000's

:Missing Attribute Values: None

:Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.

<https://archive.ics.uci.edu/ml/machine-learning-databases/housing/>

# 回归分析的典型应用

```
import matplotlib.pyplot as plt
import numpy as np

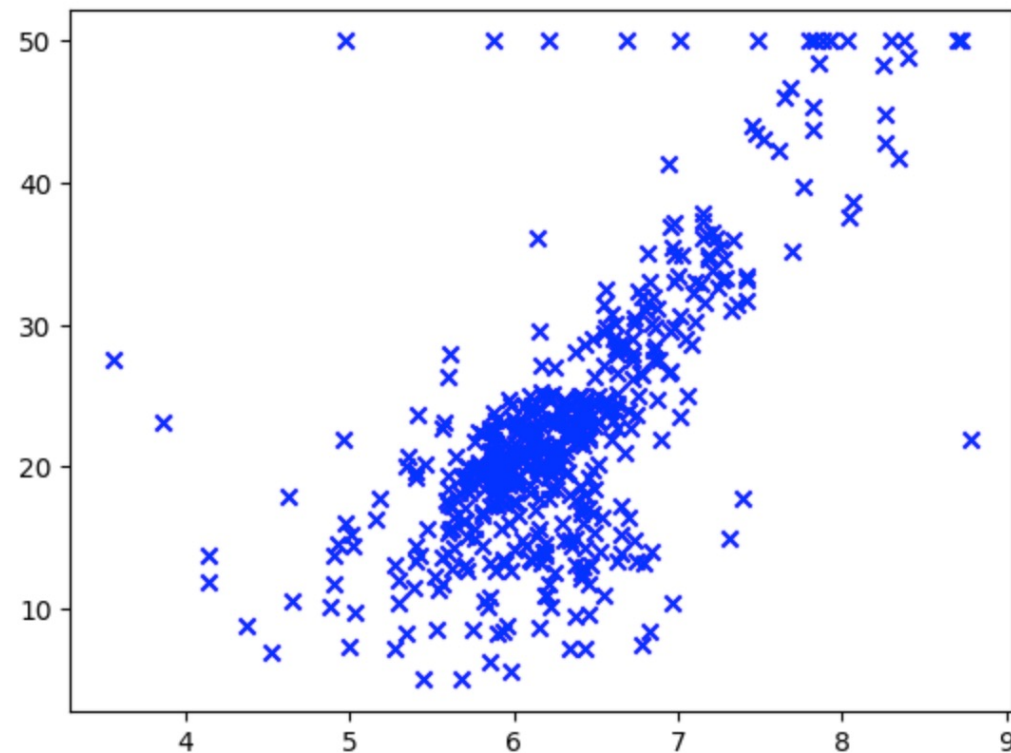
from sklearn import datasets, linear_model
#导入数据集

diabetes = datasets.load_boston()
#载入数据集

boston_X = boston.data[:, 5]
#第六列（房屋数量）逗号前面是行，后面是列

plt.scatter(boston_X, boston.target, color='blue',
marker='x')
#画散点图

plt.show()
```



这里以第六列每个住宅的平均房间数(X)来预测房中位数(Median Value)(Y)。从图中可以看出，数据集可以直接进行线性回归，下面采用线性回归对其进行回归预测。

# 回归分析的典型应用

## 导入线性回归模型，并进行回归预测与模型评估

```
# 将数据分成训练集和验证集
import random
train_ratio=0.9 #设定训练集和测试集比例
train_index=random.sample(range(boston_X.shape[0]),
int(train_ratio*boston_X.shape[0])) #获取训练集序号
test_index=list(set(range(boston_X.shape[0]))-
set(train_index)) #获取测试集序号
boston_X_train = boston_X[train_index]
boston_y_train = boston.target[train_index]
boston_X_test = boston_X[test_index]
boston_y_test = boston.target[test_index]
```

- 设定训练集和测试集比例
- 随机抽取测试集，反集为测试集
- 创建模型
- 训练（拟合过程）
- 进行预测



# 回归分析的典型应用

## 导入线性回归模型，并进行回归预测与模型评估

```
# 使用训练集训练模型
boston_X_train=np.array(boston_X_train).reshape(-1,1)
#这是由于在新版的sklearn中，所有的数据都应该是二维矩阵，
哪怕它只是单独一行或一列
#所以需要使用.reshape(1,-1)进行转换
print(boston_X_train.shape)

regr.fit(boston_X_train, boston_y_train)
# 使用测试集进行预测
boston_X_test=np.array(boston_X_test).reshape(-1,1)
boston_y_pred = regr.predict(boston_X_test)
# boston_y_pred_train = regr.predict(boston_X_train)
```

- 设定训练集和测试集比例
- 随机抽取测试集，反集为测试集
- 创建模型
- 训练（拟合过程）
- 进行预测

# 回归分析的典型应用

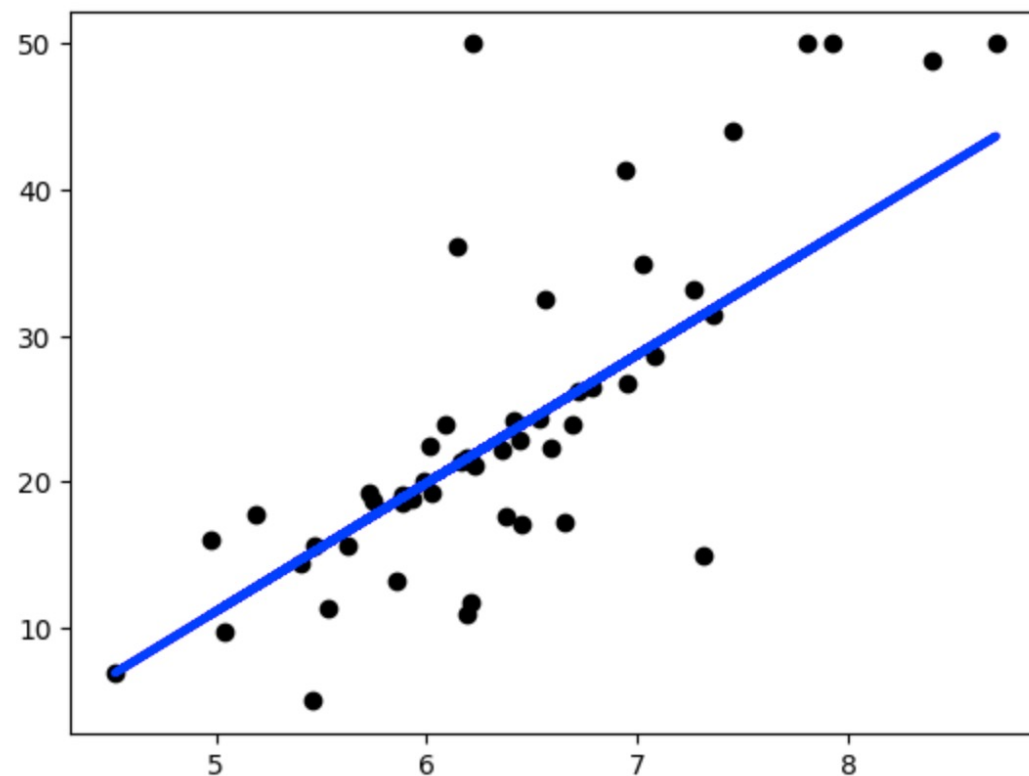
## 最后进行模型评估

```
from sklearn.metrics import
mean_squared_error
# 回归系数
print('Coefficients: ', regr.coef_)
# 均方误差
print("Mean squared error: %.2f"
      % mean_squared_error(boston_y_test,
                           diabetes_y_pred))
```

## 绘图

```
from sklearn.metrics import
mean_squared_error
# 回归系数
print('Coefficients: ', regr.coef_)
# 均方误差
print("Mean squared error: %.2f"
      % mean_squared_error(boston_y_test,
                           boston_y_pred))
```

Coefficients: [8.78376183]  
Mean squared error: 23645.27



# 逻辑回归与线性回归的区别

- 线性回归

- ◆ 线性回归直接将 $b_0 + b_1x_1 + \dots + b_nx_n$ 作为因变量，即

$$y = b_0 + b_1x_1 + \dots + b_nx_n$$

- ◆ 如前面提到的，使用 BMI（数值）预测糖尿病发展情况（数值），目标是进行回归（预测），使用线性回归

- 逻辑回归

- ◆ 广义线性回归模型，与线性回归模型都具有  $y = b_0 + b_1x_1 + \dots + b_nx_n$

- ◆ 逻辑回归使用将 $b_0 + b_1x_1 + \dots + b_nx_n$ 作为间接量，即  $g(y) = \frac{1}{1+e^{-y}}$

- ◆ 逻辑回归将 $b_0 + b_1x_1 + \dots + b_nx_n$ 映射到 $[0,1]$ ：属于某一类的可能性多大

- ◆ 如前面提到的，使用花萼长度与宽度（数值）预测花的种类（分类标签），目标是进行分类，使用逻辑回归

# 回归分析的典型应用

---

**逻辑回归LogisticRegression分析鸢尾花数据**

# 回归分析的典型应用

---

LogisticRegression回归模型在Sklearn.linear\_model子类下，调用sklearn逻辑回归算法步骤主要有3步，即：

- (1) 导入模型。调用逻辑回归LogisticRegression()函数。
- (2) fit()训练。调用fit(x,y)的方法来训练模型，其中x为数据的属性，y为所属类型
- (3) predict()预测。利用训练得到的模型对数据集进行预测，返回预测结果。



# 回归分析的典型应用

在Sklearn机器学习包中，集成了各种各样的数据集，这里引入的是鸢尾花卉（Iris）数据集，它是很常用的一个数据集。鸢尾花有三个亚属，分别是山鸢尾（Iris-setosa）、变色鸢尾（Iris-versicolor）和维吉尼亚鸢尾（Iris-virginica）。该数据集一共包含4个特征变量，1个类别变量。共有150个样本。

| 列名          | 说明                                | 类型    |
|-------------|-----------------------------------|-------|
| SepalLength | 花萼长度                              | float |
| SepalWidth  | 花萼宽度                              | float |
| PetalLength | 花瓣长度                              | float |
| PetalWidth  | 花瓣宽度                              | float |
| Class       | 类别变量。0 表示山鸢尾，1 表示变色鸢尾，2 表示维吉尼亚鸢尾。 | int   |

# 回归分析的典型应用

---

导入逻辑回归模型

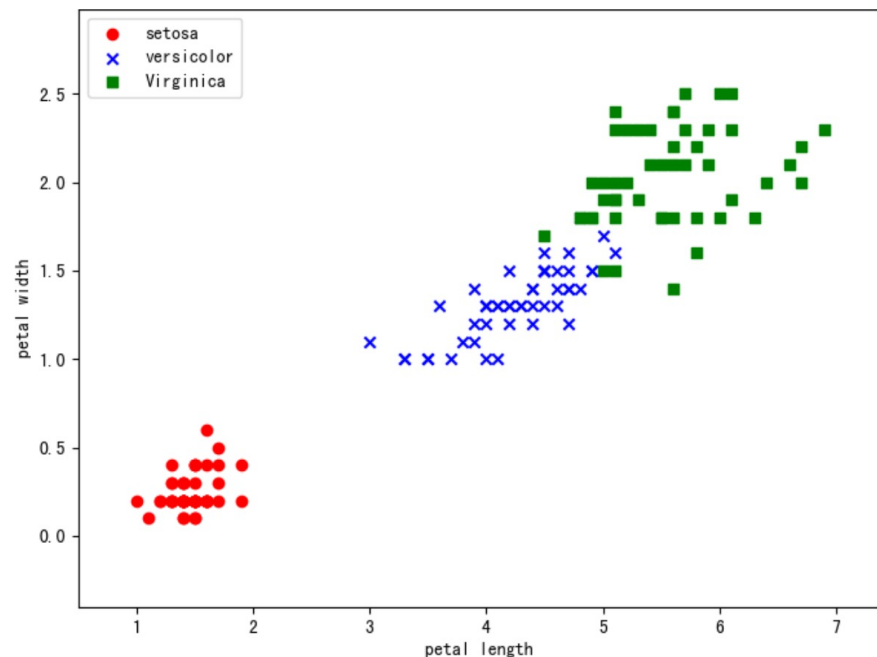
```
from sklearn.linear_model import LogisticRegression #导入逻辑回归模型
clf = LogisticRegression()
print(clf)
```

导入数据集

```
from sklearn.datasets import load_iris #导入数据集iris
iris = load_iris() #载入数据集
print(iris.data) #数据
print(iris.target) #对应的标签
```

# 回归分析的典型应用

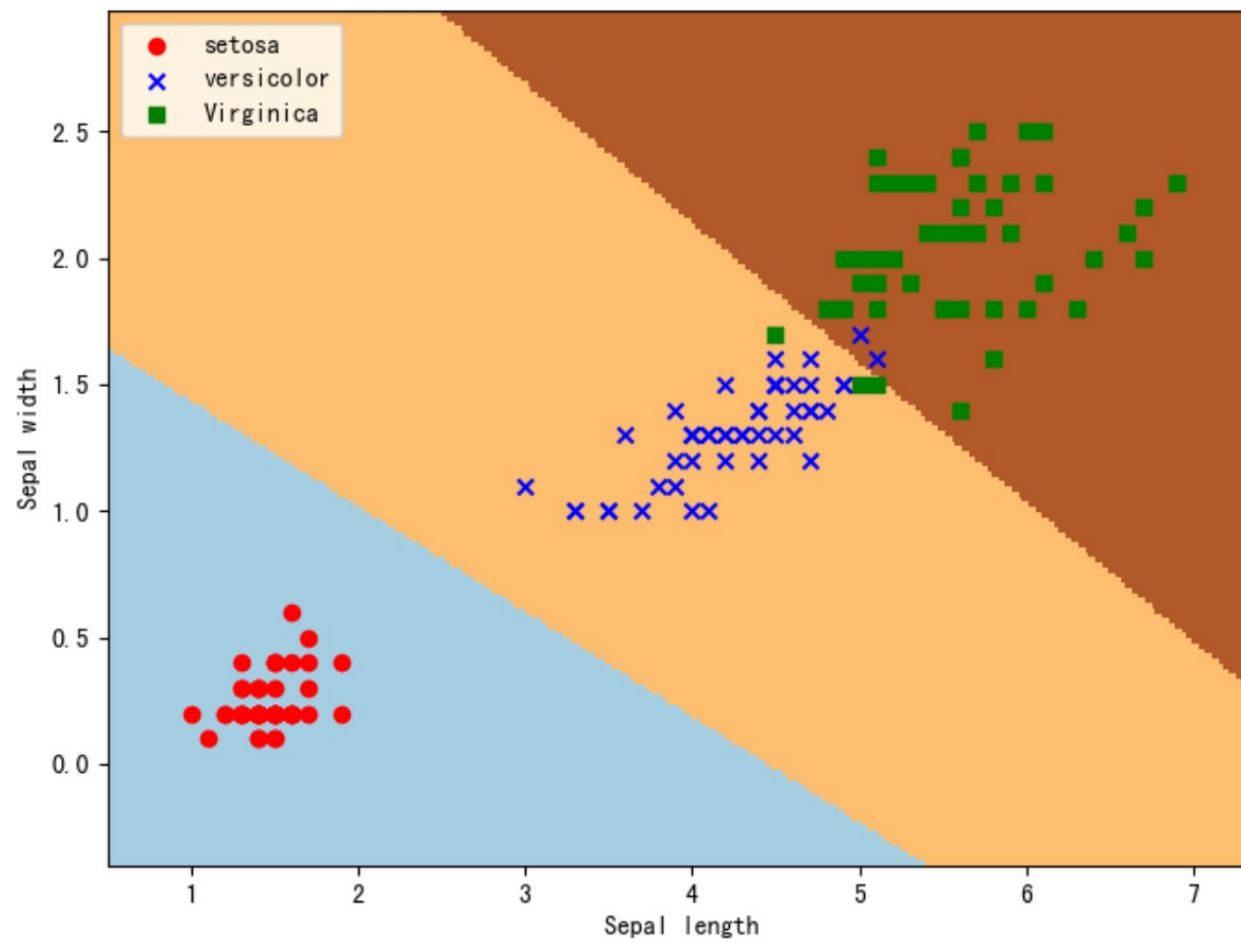
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import load_iris    #导入数据集iris
iris = load_iris() #载入数据集
print(iris.data)           #输出数据集
print(iris.target)         #输出真实标签
X = iris.data[:, -2:]      #获取花卉后两列数据集
Y = iris.target
plt.scatter(X[:50], Y[:50], color='red',
            marker='o', label='setosa') #前50个样本,代表第一类
plt.scatter(X[50:100], Y[50:100], color='blue',
            marker='x', label='versicolor') #中间50个,代表第二类
plt.scatter(X[100:], Y[100:], color='green',
            marker='+', label='Virginica') #后50个样本,代表第三类
plt.legend(loc=2) #左上角
plt.show()
```



这里以前两列为主要特征进行分析。从图中可以看出，数据集线性可分的，可以划分为3类，分别对应三种类型的鸢尾花，下面采用逻辑回归对其进行分类预测。

# 回归分析的典型应用

## 分类结果



# 回归分析的典型应用

---

**K-means聚类分析鸢尾花数据**



# K-means实现鸢尾花数据的聚类

---

导入数据集

```
from sklearn.datasets import load_iris    #导入数据集iris
iris = load_iris()    #载入数据集
print(iris.data)    #数据
print(iris.target)    #对应的标签
```

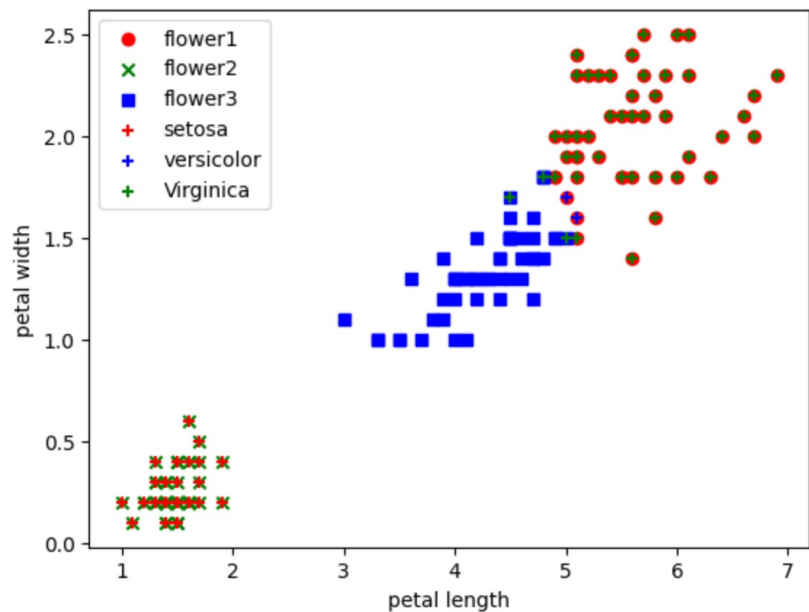
导入数据集

```
import matplotlib.pyplot as plt #导入绘图包组建
import numpy as np #导入数组计算库
from sklearn.cluster import Kmeans #导入K-means聚类模型
```

# K-means实现鸢尾花数据的聚类

## 构造聚类模型

```
estimator = KMeans(n_clusters=3)
#构造聚类器
estimator.fit(X)
#聚类
label_pred = estimator.labels_
#获取聚类标签
```



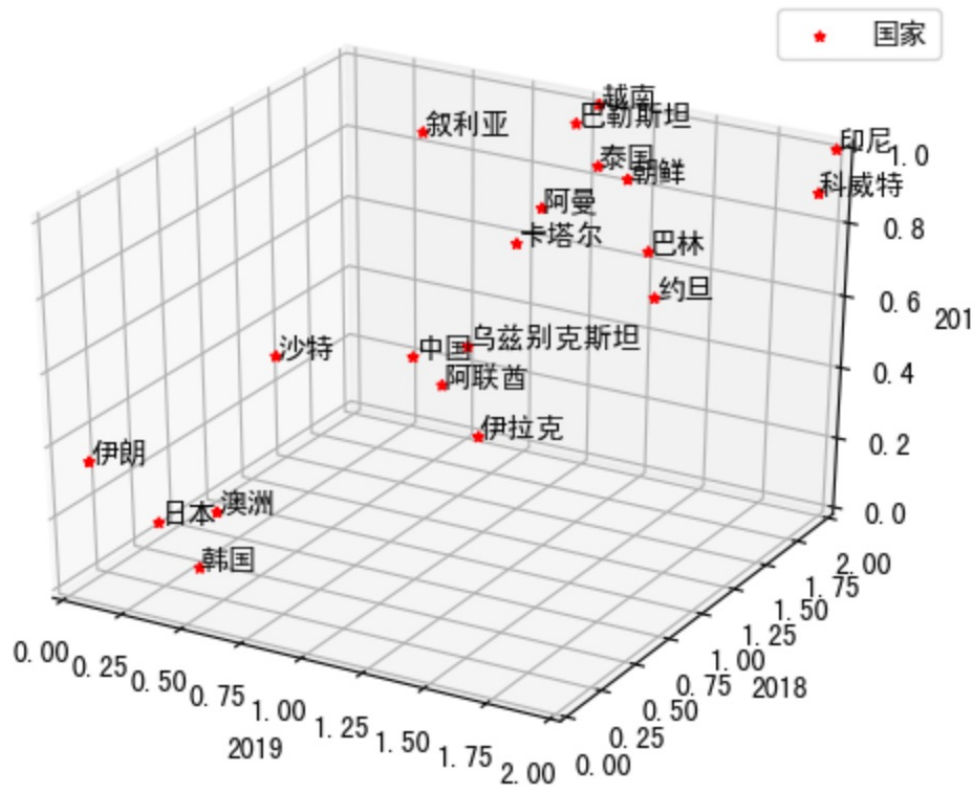
## 绘制k-means结果

```
x0 = X[label_pred == 0]
x1 = X[label_pred == 1]
x2 = X[label_pred == 2]
plt.scatter(x0[:, 0], x0[:, 1], c = "red",
            marker='o', label='flower1')
plt.scatter(x1[:, 0], x1[:, 1], c =
            "green", marker='x', label='flower2')
plt.scatter(x2[:, 0], x2[:, 1], c =
            "blue", marker='s', label='flower3')
plt.xlabel('petal length')
plt.ylabel('petal width')
plt.legend(loc=2)
plt.show()
```

**K-means案例：中国男足到底在亚洲处于几流水平？**

# K-means案例—— 中国男足到底在亚洲处于几流水平？

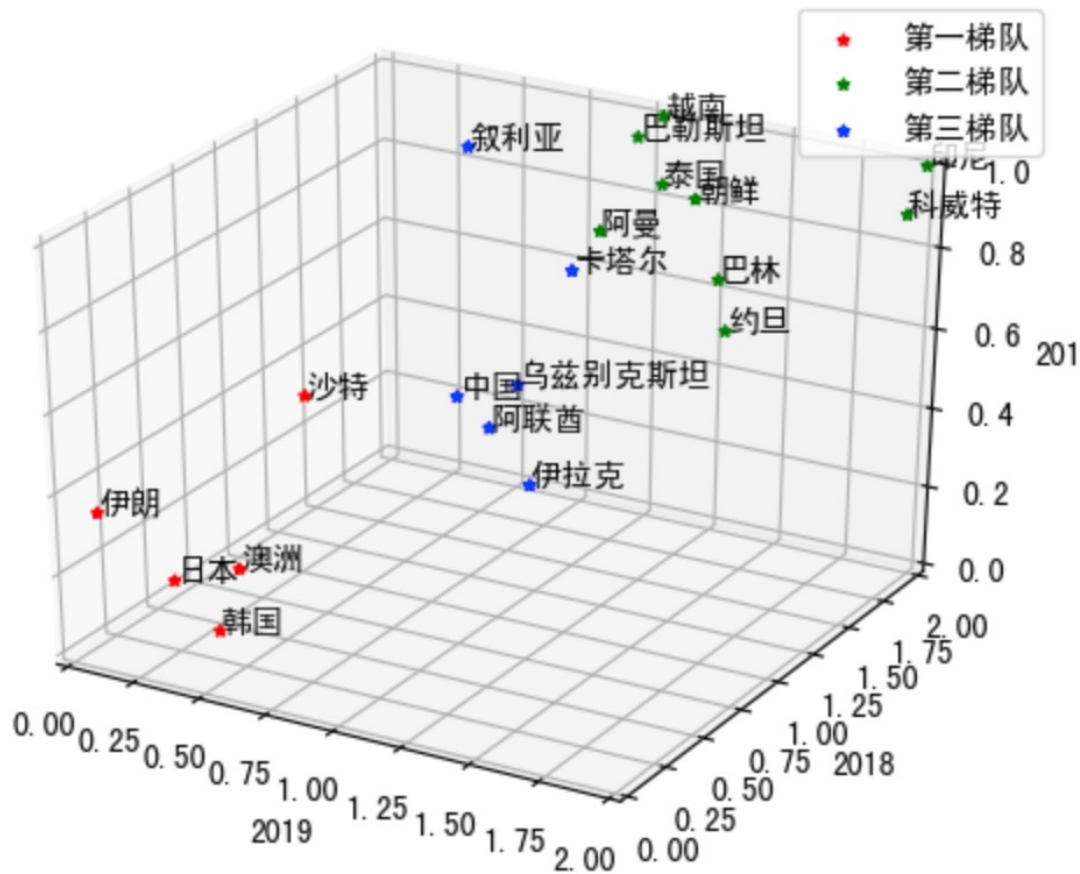
- 2019国际排名权重：2
- 2018世界杯权重：2
- 2015亚洲杯权重:1



|    | 球队     | 2019年国际排名 | 2018世界杯 | 2015亚洲杯 |
|----|--------|-----------|---------|---------|
| 0  | 中国     | 73        | 40      | 7       |
| 5  | 伊拉克    | 91        | 40      | 4       |
| 7  | 阿联酋    | 81        | 40      | 6       |
| 8  | 乌兹别克斯坦 | 88        | 40      | 8       |
| 17 | 约旦     | 118       | 50      | 9       |
| 16 | 叙利亚    | 76        | 40      | 17      |
| 14 | 印尼     | 164       | 50      | 17      |
| 13 | 朝鲜     | 110       | 50      | 14      |
| 12 | 巴林     | 116       | 50      | 11      |
| 11 | 阿曼     | 87        | 50      | 12      |
| 9  | 泰国     | 122       | 40      | 17      |
| 18 | 科威特    | 160       | 50      | 15      |
| 6  | 卡塔尔    | 101       | 40      | 13      |
| 10 | 越南     | 102       | 50      | 17      |
| 19 | 巴勒斯坦   | 96        | 50      | 16      |
| 4  | 沙特     | 67        | 26      | 10      |
| 3  | 伊朗     | 34        | 18      | 6       |
| 15 | 澳洲     | 40        | 30      | 1       |
| 2  | 韩国     | 61        | 19      | 2       |
| 1  | 日本     | 60        | 15      | 5       |

2015-2019 年亚洲球队的排名

# K-means案例—— 中国男足近几年到底在亚洲处于几流水平？



|    | 国家     | 2019国际排名             | 2018世界杯             | 2015亚洲杯 |
|----|--------|----------------------|---------------------|---------|
| 0  | 中国     | 0.3                  | 0.7142857142857143  | 0.375   |
| 1  | 伊拉克    | 0.43846153846153846  | 0.7142857142857143  | 0.1875  |
| 2  | 阿联酋    | 0.36153846153846153  | 0.7142857142857143  | 0.3125  |
| 3  | 乌兹别克斯坦 | 0.4153846153846154   | 0.7142857142857143  | 0.4375  |
| 4  | 约旦     | 0.6461538461538462   | 1.0                 | 0.5     |
| 5  | 叙利亚    | 0.3230769230769231   | 0.7142857142857143  | 1.0     |
| 6  | 印尼     | 1.0                  | 1.0                 | 1.0     |
| 7  | 朝鲜     | 0.5846153846153846   | 1.0                 | 0.8125  |
| 8  | 巴林     | 0.6307692307692307   | 1.0                 | 0.625   |
| 9  | 阿曼     | 0.4076923076923077   | 1.0                 | 0.6875  |
| 10 | 泰国     | 0.676923076923077    | 0.7142857142857143  | 1.0     |
| 11 | 科威特    | 0.9692307692307692   | 1.0                 | 0.875   |
| 12 | 卡塔尔    | 0.5153846153846153   | 0.7142857142857143  | 0.75    |
| 13 | 越南     | 0.5230769230769231   | 1.0                 | 1.0     |
| 14 | 巴勒斯坦   | 0.47692307692307695  | 1.0                 | 0.9375  |
| 15 | 沙特     | 0.25384615384615383  | 0.3142857142857143  | 0.5625  |
| 16 | 伊朗     | 0.0                  | 0.08571428571428572 | 0.3125  |
| 17 | 澳洲     | 0.046153846153846156 | 0.42857142857142855 | 0.0     |
| 18 | 韩国     | 0.2076923076923077   | 0.11428571428571428 | 0.0625  |
| 19 | 日本     | 0.2                  | 0.0                 | 0.25    |

图：[0,1]规格化后的数据

# 作业要求——任务运行结果截图，导出为pdf后发送到助教邮箱

## ➤ 必做：

- numpy创建数组，数组形状修改结果截图
- 输出波士顿房价数据集的所有属性类型及其样本数量
- 数据散点图，自变量DIS（与波士顿中心区距离），因变量房价（学号尾号为基数，散点图为红色方形；学号尾号为偶数，散点图为蓝色圆形）
- 线性回归的回归结果折线图及散点图展示
- `boston_X_train=np.array(boston_X_train).reshape(-1,1)`句的意义？

## 基于 'sepal length (cm), sepal width (cm)' 两个维度进行逻辑回归与聚类实验

- 鸢尾花数据集逻辑回归散点图
- 输出逻辑回归系数
- 对鸢尾花数据进行K-means聚类，绘制聚类中心为3的聚类结果图（基于 'sepal length (cm)', 'sepal width (cm)' 两个维度聚类）
- 选做
  - 基于给出亚洲足球数据集进行聚类，分析中国男足水平（给出运行代码，3D可视化截图）
  - 分析不同权重对于结果影响；分析有无数据正则化对结果影响；
  - Matplotlib使用、聚类数改变.....

