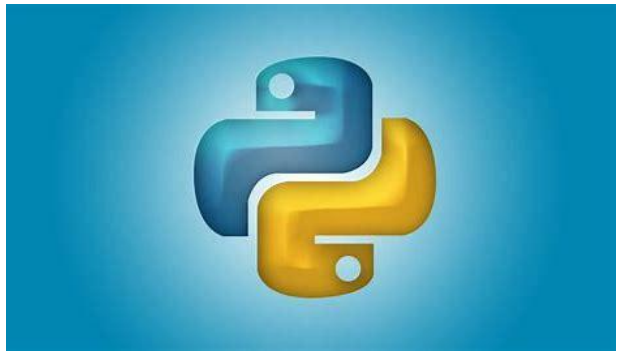
Python运用之
数据预处理与机器学习

分享人：庞国飞
东南大学数学学院
2025年12月5日

Python基础+进阶	学习任务细化	需要掌握的Python包
Python语言基础	解释性语言	—
	IDE/文件/模块	—
	程序控制语句/函数	—
	画图 & 强推的Python在线教程	matplotlib、 seaborn
Python科学计算	数组与矩阵	numpy
	插值	scipy
	数值积分	scipy
	优化	scipy
Python数据科学	数据预处理： 缺失值/异常值处理、归一化、标准化 合并、分组与聚合	numpy/pandas/scikit-learn
	时间序列预测	statsmodels/pytorch
	机器学习： 监督学习（回归与分类） 无监督学习（降维、概率密度估计）	scikit-learn pytorch或tensorflow

Python基础+进阶	学习任务细化	需要掌握的Python包
Python语言基础	解释性语言	—
	IDE/文件/模块	—
	程序控制语句/函数	—
	画图 & 强推的Python在线教程	matplotlib、 seaborn
Python科学计算	数组与矩阵	numpy
	插值	scipy
	数值积分	scipy
	优化	scipy
Python数据科学	数据预处理： 缺失值/异常值处理、归一化、标准化 合并、分组与聚合	numpy/pandas/scikit-learn
	时间序列预测	statsmodels/pytorch
	机器学习： 监督学习（回归与分类） 无监督学习（降维、概率密度估计）	scikit-learn pytorch或tensorflow

编程语言	编译型	解释性	面向对象编程
C++	Yes		Yes
Python		Yes	Yes



Monty Python （六人喜剧团）

使用场景	首选 IDE	备选 IDE
零基础入门	Thonny / IDLE	VS Code（简化配置）
企业级 Python 开发	PyCharm 专业版	VS Code
数据分析 / 机器学习	Jupyter Lab + VS Code	PyCharm 社区版
全栈开发（Python + 前端）	VS Code	PyCharm 专业版
科学计算 / 工程建模	Spyder	Jupyter Notebook

Python文件	Python模块
以.py结尾的文件，比如常用的 画图接口文件：pyplot.py 或者自定义的某个文件，如 my_file.py	（1）python文件本身，文件名即为模块名，比如my_module.py 或者 （2）包含python文件的文件夹，文件夹名即为模块名,比如python自带的包——文件夹 matplotlib ■ 使用模块的方式之一为 import 模块名 as 模块名的缩写 比如import matplotlib.pyplot as plt 这里matplotlib是一个文件夹，内含python文件 pyplot.py

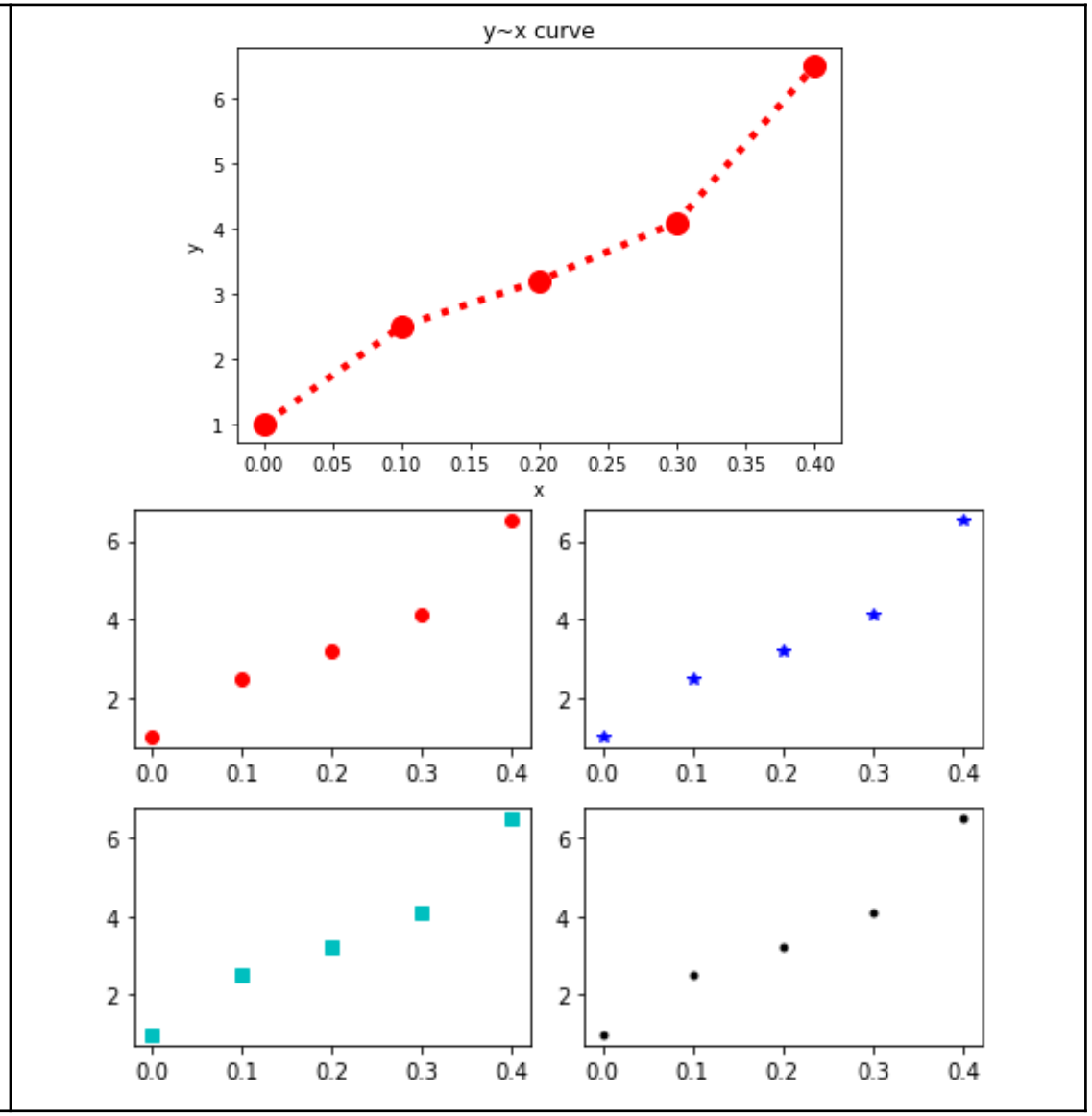
三大程序控制语句 & 函数

```
###顺序
a = 3
print(a + 3)


###分支
if a > 3:
    print(f"结果是{a+1}\n")
elif a <= 3:
    print(f"结果是{a+2}\n")
```

```
###循环
counter = 0
for i in range(10):
    counter = counter +1
print(counter)

###函数
def adding(a, b):
    return a + b

print(adding(2.3, 4.7))
```

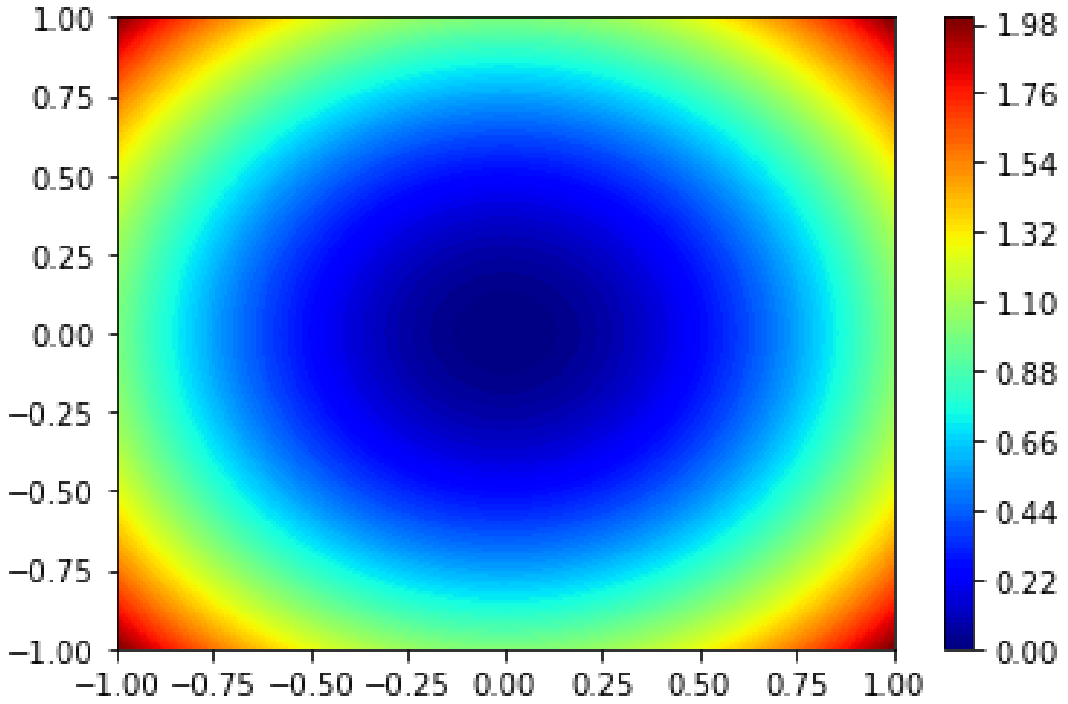
画线图 plot()　&子图　subplot()

```
import matplotlib.pyplot as plt  //调画图包
import numpy as np               //调数组包
x = np.array([0.0, 0.1, 0.2, 0.3, 0.4])
y = np.array([1.0, 2.5, 3.2, 4.1, 6.5])
plt.plot(x, y, 'ro:', linewidth = 4, markersize=12)
plt.xlabel('x')
plt.ylabel('y')
plt.title('y~x curve')
plt.show()
plt.subplot(2,2,1)    //画子图
plt.plot(x,y,'ro')
plt.subplot(2,2,2)
plt.plot(x,y,'b*')
plt.subplot(2,2,3)
plt.plot(x,y, 'cs')
plt.subplot(2,2,4)
plt.plot(x,y,'k.')
plt.tight_layout()
plt.show()
```



画等值线图

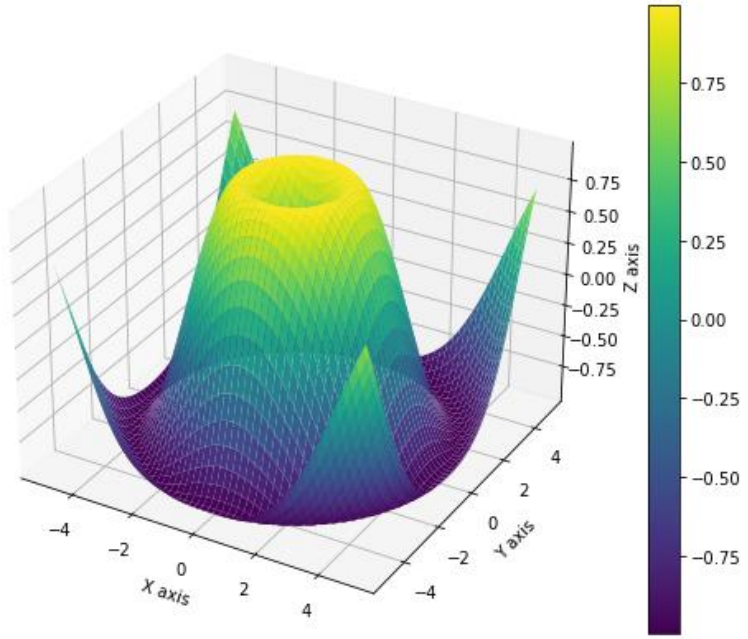
```
import matplotlib.pyplot as plt
import numpy as np

x_vec = np.linspace(-1, 1, 31)
y_vec = np.linspace(-1, 1, 31)
x_grid, y_grid = np.meshgrid(x_vec,
y_vec)
z = x_grid**2 + y_grid**2

plt.contourf(x_grid, y_grid, z, 100,
cmap='jet')
plt.colorbar()
plt.show()
```



画曲面图

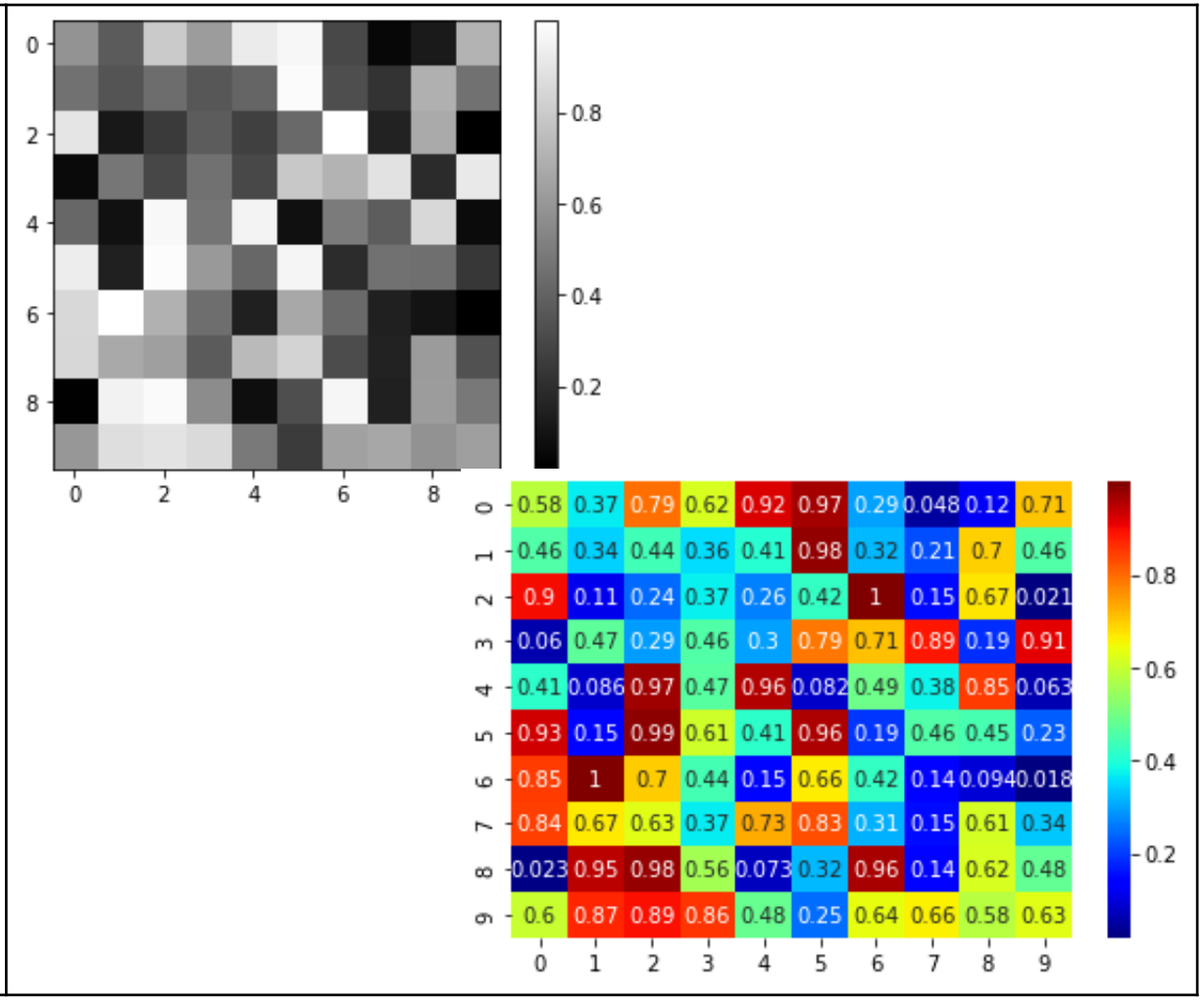
```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
# 定义网格范围和分辨率
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
# 定义 Z 值为函数 f(X, Y)
Z = np.sin(np.sqrt(X**2 + Y**2))
fig = plt.figure(figsize=(10,7))
ax = fig.add_subplot(111, projection='3d')
# 绘制曲面图，设置颜色映射
surf = ax.plot_surface(X, Y, Z, cmap='viridis')
```

```
# 添加颜色条
fig.colorbar(surf)
# 设置坐标轴标签
ax.set_xlabel('X axis')
ax.set_ylabel('Y axis')
ax.set_zlabel('Z axis')
plt.show()
```



画灰度图&热力图

```
import matplotlib.pyplot as plt
import numpy as np
matrix = np.random.rand(10, 10)

plt.imshow(matrix, cmap = "gray")
plt.show()
# seaborn是基于matplotlib的高级可视化库
import seaborn as sns
sns.heatmap(matrix, annot = True, cmap = "jet")
plt.show()
```



强推的Python学习网站：Python3菜鸟教程

Python基础+进阶	学习任务细化	需要掌握的Python包
Python语言基础	解释性语言	—
	IDE/文件/模块	—
	程序控制语句/函数	—
	画图 & 强推的Python在线教程	matplotlib、 seaborn
Python科学计算	数组与矩阵	numpy
	插值	scipy
	数值积分	scipy
	优化	scipy
Python数据科学	数据预处理： 缺失值/异常值处理、归一化、标准化 合并、分组与聚合	numpy/pandas/scikit-learn
	时间序列预测	statsmodels/pytorch
	机器学习： 监督学习（回归与分类） 无监督学习（降维、概率密度估计）	scikit-learn pytorch或tensorflow

数组与矩阵

- 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。
- NumPy 是一个运行速度非常快的数学库，主要用于数组计算，包含：

  一个强大的N维数组对象 ndarray

  线性代数、傅里叶变换、随机数生成等功能

  numpy的简单使用见 numpy_array.py

插值、数值积分、优化

- SciPy 是一个开源的 Python 算法库和数学工具包。
- Scipy 是基于 Numpy 的科学计算库，用于数学、科学、工程学等领域，很多有一些高阶抽象和物理模型需要使用 Scipy。
- SciPy 包含的模块有**最优化、线性代数、积分、插值、特殊函数、快速傅里叶变换、信号处理和图像处理、常微分方程求解**和其他科学与工程中常用的计算。

用SciPy做插值

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

x = np.array([0.0, 0.1, 0.2, 0.3, 0.4])
y = np.array([1.0, 2.5, 3.2, 4.1, 6.5])

from scipy.interpolate import interp1d
interp_fun = interp1d(x, y)      #基于插值点对x和y，构造出插值函数
x_test = 0.266                   #估计该点的函数值
print(interp_fun(x_test))
```

用SciPy做数值积分

```
from scipy.integrate import quad    #导入数值积分函数quad
import numpy as np

result = quad(lambda x: np.exp(x**2), 0, 1)

print(result)
```

$$\int_0^1 e^{x^2}\,\mathrm{d}x = ?$$

用SciPy做最优化

```
# 求解带约束的优化问题
from scipy.optimize import minimize
fun = lambda x: (x[0] - 1)**2 + (x[1] - 2.5)**2 #以匿名函数定义目标函数
bounds = ((0, None), (0, None))
constraints = [
  {'type': 'ineq', 'fun': lambda x:  x[0] - 2 * x[1] + 2},
  {'type': 'ineq', 'fun': lambda x: -x[0] - 2 * x[1] + 6},
  {'type': 'ineq', 'fun': lambda x: -x[0] + 2 * x[1] + 2}
  ]
x0 = (5.0, 0.0)
result = minimize(fun = fun, x0 = x0,
                  method = 'SLSQP', bounds = bounds,
                  constraints = constraints)
print(result.x)
```

例3：求如下极小值问题

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 2.5)^2$$

其中，

$$\begin{cases} x_1 - 2x_2 + 2 \geq 0; \\ -x_1 - 2x_2 + 6 \geq 0; \\ -x_1 + 2x_2 + 2 \geq 0; \\ x_1 > 0, x_2 > 0. \end{cases}$$

用SciPy做最优化

场景	推荐库	特点
常规数学优化（无约束 / 有约束）	scipy.optimize	入门首选，接口简单
运筹优化（LP/IP/MIP）	Pyomo/Gurobi/CPLEX	支持整数 / 混合整数规划
深度学习梯度优化	torch.optim/tf.keras.optimizers	适配自动求导，高效
凸优化专用	cvxpy	语法简洁，专注凸问题
非线性 / 无梯度优化（启发式算法，如遗传算法、模拟退火）	nlopt/DEAP	算法丰富，适配非凸场景

Python基础+进阶	学习任务细化	需要掌握的Python包
Python语言基础	解释性语言	—
	IDE/文件/模块	—
	程序控制语句/函数	—
	画图 & 强推的Python在线教程	matplotlib、 seaborn
Python科学计算	数组与矩阵	numpy
	插值	scipy
	数值积分	scipy
	优化	scipy
Python数据科学	数据预处理： 缺失值/异常值处理、归一化、标准化 合并、分组与聚合	numpy/pandas/scikit-learn
	时间序列预测	statsmodels/pytorch
	机器学习： 监督学习（回归与分类） 无监督学习（降维、概率密度估计）	scikit-learn pytorch或tensorflow

缺失值处理——使用numpy vs 使用pandas

```
import pandas as pd
import numpy as np

data = pd.read_csv("Salary_Data.csv")#读入数据
```

Salary_Data.csv

	A	B
1	Year	Salary
2	1.1	3.9343
3	1.3	4.6205
4	1.5	3.7731
5	2	4.3525
6	2.2	3.9891
7	2.9	5.6642
8	3	6.015
9	3.2	5.4445
10	3.2	6.4445
11		5.7189
12	3.9	6.3218
13	4	5.5794
14	4	5.6957
15	4.1	5.7081
16	4.5	6.1111
17	4.9	6.7938

缺失值处理——使用pandas

```
# 以numpy去处理空值
data_np = data.to_numpy()  #转换为numpy数组
c1, c2 = data_np[:,0:1], data_np[:,1:2] #逐列处理
#用列均值填充空值
c11 = np.nan_to_num(c1, nan=np.nanmean(c1))
c22 = np.nan_to_num(c2, nan=np.nanmean(c2))
#将填充后的列重新拼起来
data_filled = np.concatenate((c11,c22),axis=1)
#直接删除带有空值的行
data_cleared =
data_np[~np.isnan(data_np).any(axis=1)]
```

```
#以pandas处理空值

c3, c4 = data['Year'], data['Salary']#逐列处理
#用列均值填充空值
c33 = c3.fillna(c3.mean())
c44 = c4.fillna(c4.mean())
#将填充后的列重新拼起来
data_filled_pd = pd.concat([c33, c44],axis=1)
#直接删除带有空值的行
data_cleared_pd =
data.dropna(axis=0, how='any')
```

数据的合并——使用pandas

美赛2022年C题：
比特币/黄金组合投资策略问题

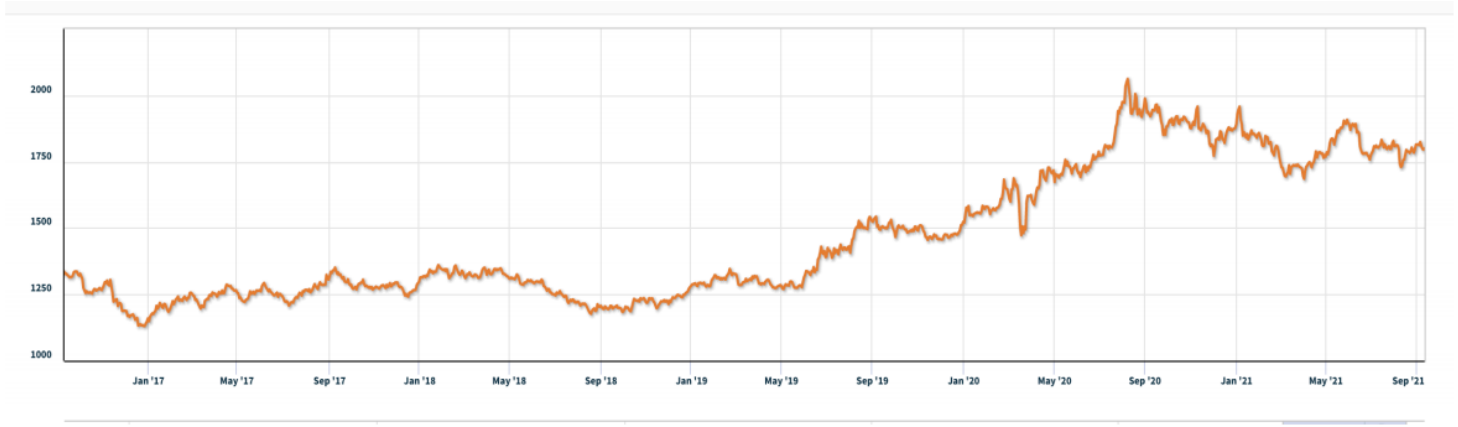


Figure 1: Gold daily prices, U.S. dollars per troy ounce. Source: London Bullion Market Association, 9/11/2021

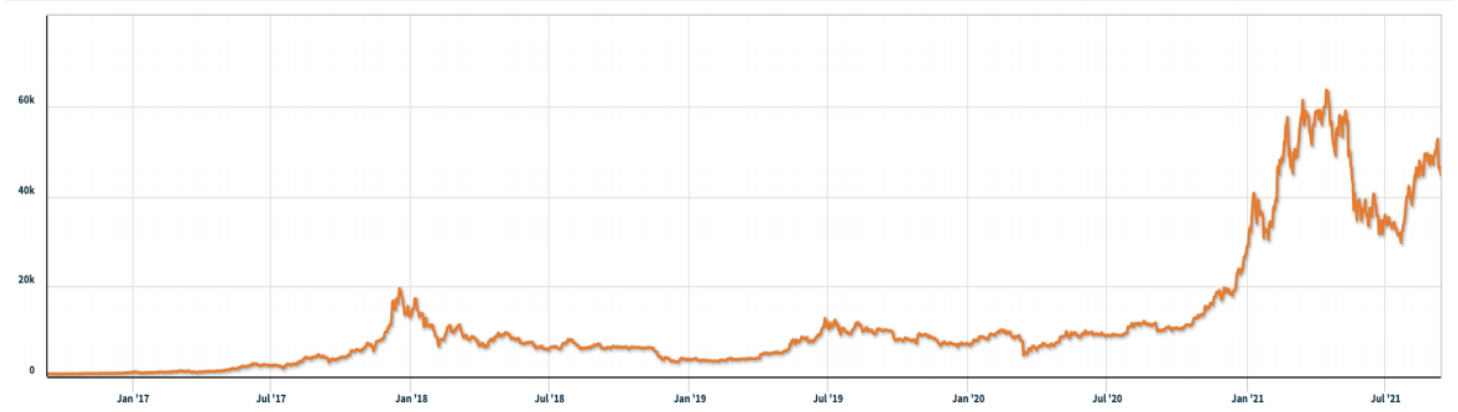
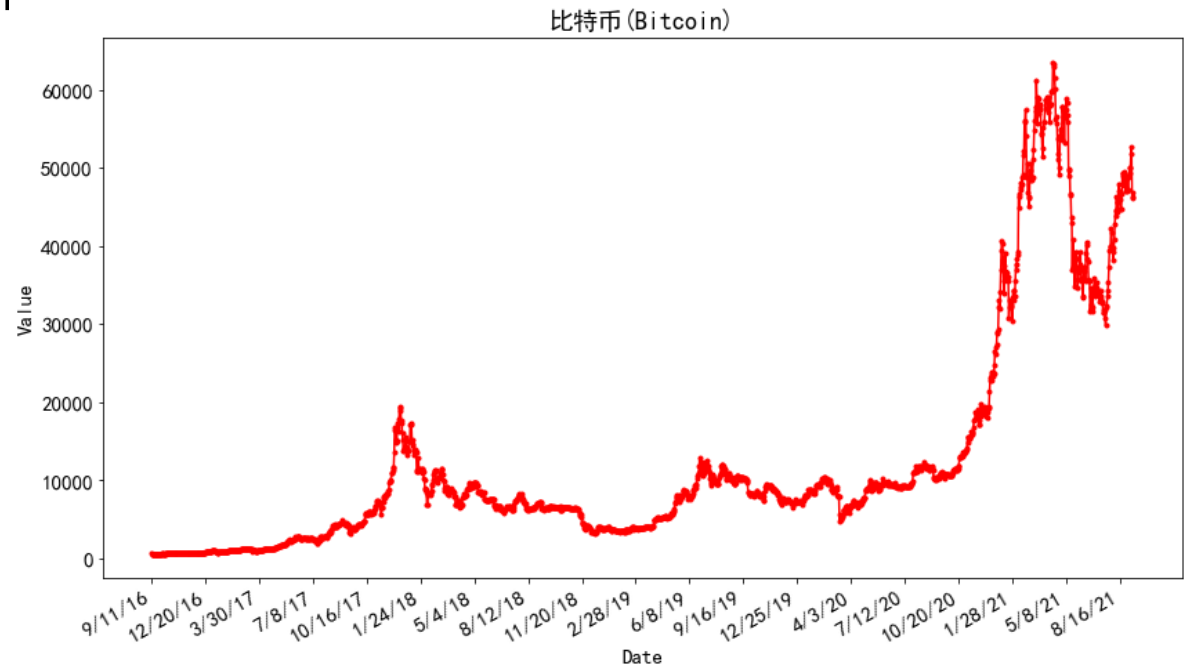


Figure 2: Bitcoin daily prices, U.S. dollars per bitcoin. Source: NASDAQ, 9/11/2021

数据的合并——使用pandas

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
B = pd.read_csv("BCHAIN-MKPRU.csv")
G = pd.read_csv("LBMA-GOLD.csv")
# 设置全局字体
plt.rcParams['font.sans-serif'] = ['SimHei']
# 设置为黑体
plt.rcParams['font.size'] = 15#设置字体大小为15号字
plt.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
```

```
### 时间序列可视化
plt.figure(figsize=(14,8))
plt.plot(B['Date'], B['Value'],'r.-')
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('比特币(Bitcoin)')
plt.xticks(B['Date'][::100])
plt.gcf().autofmt_xdate()
plt.show()
```



数据的合并——使用pandas

```
#将比特币和黄金的交易价格按日期合并
B['Date'] = pd.to_datetime(B['Date'],
format='%m/%d/%y')
G['Date'] = pd.to_datetime(G['Date'],
format='%m/%d/%y')

BG = pd.merge(B, G, how="outer",
on=["Date"])
##  将双休日的黄金价格取为周五的收盘价
  (即向前填充)
BG0 = BG.fillna(method = 'ffill', axis=0)
##首个日期无前置价格，故采用个向后填充
BG1 =  BG0.fillna(method = 'bfill', axis=0)
```

BG1 - DataFrame

Index	Date	Value	USD (PM)
0	2016-09-11 00:00:00	621.65	1324.6
1	2016-09-12 00:00:00	609.67	1324.6
2	2016-09-13 00:00:00	610.92	1323.65
3	2016-09-14 00:00:00	608.82	1321.75
4	2016-09-15 00:00:00	610.38	1310.8
5	2016-09-16 00:00:00	609.11	1308.35
6	2016-09-17 00:00:00	607.04	1308.35
7	2016-09-18 00:00:00	611.58	1308.35
8	2016-09-19 00:00:00	610.19	1314.85
9	2016-09-20 00:00:00	608.66	1313.8
10	2016-09-21 00:00:00	598.88	1326.1
11	2016-09-22 00:00:00	597.42	1339.1
12	2016-09-23 00:00:00	594.08	1338.65
13	2016-09-24 00:00:00	603.88	1338.65
14	2016-09-25 00:00:00	601.74	1338.65
15	2016-09-26 00:00:00	598.98	1340.5
16	2016-09-27 00:00:00	605.96	1327
17	2016-09-28 00:00:00	605.67	1322.5
18	2016-09-29 00:00:00	603.85	1318.1
19	2016-09-30 00:00:00	609.39	1322.5
20	2016-10-01 00:00:00	614.82	1322.5

数据的合并、分组、聚合——使用pandas

国赛2023年C题： 湖北某商超的定价策略与补货策略

	A	B	C	D
1	单品编码	单品名称	分类编码	分类名称
2	102900005115168	牛首生菜	1011010101	花叶类
3	102900005115199	四川红香椿	1011010101	花叶类
4	102900005115625	本地小毛白菜	1011010101	花叶类
5	102900005115748	白菜苔	1011010101	花叶类
6	102900005115762	苋菜	1011010101	花叶类
7	102900005115779	云南生菜	1011010101	花叶类
8	102900005115786	竹叶菜	1011010101	花叶类
9	102900005115793	小白菜	1011010101	花叶类
10	102900005115816	南瓜尖	1011010101	花叶类
11	102900005115823	上海青	1011010101	花叶类
12	102900005115854	萝卜叶	1011010101	花叶类
13	102900005115861	牛首油菜	1011010101	花叶类
14	102900005115878	茼蒿	1011010101	花叶类
15	102900005115885	蔡甸藜蒿	1011010101	花叶类
16	102900005115908	菜心	1011010101	花叶类
17	102900005115946	木耳菜	1011010101	花叶类

数据集1： 251条记录

A1 销售日期

	A	B	C	D	E	F	G
1	销售日期	扫码销售时间	单品编码	销量(千克)	销售单价(元/千克)	销售类型	是否打折销售
2	2020-07-01	09:15:07.924	102900005117056	0.396	7.60	销售	否
3	2020-07-01	09:17:27.295	102900005115960	0.849	3.20	销售	否
4	2020-07-01	09:17:33.905	102900005117056	0.409	7.60	销售	否
5	2020-07-01	09:19:45.450	102900005115823	0.421	10.00	销售	否
6	2020-07-01	09:20:23.686	102900005115908	0.539	8.00	销售	否
7	2020-07-01	09:21:55.556	102900005117056	0.277	7.60	销售	否
8	2020-07-01	09:21:56.536	102900005115779	0.338	8.00	销售	否
9	2020-07-01	09:22:01.274	102900005117056	0.132	7.60	销售	否
10	2020-07-01	09:22:01.476	102900005115779	0.213	8.00	销售	否
11	2020-07-01	09:22:15.998	102900011008522	0.514	8.00	销售	否
12	2020-07-01	09:22:21.264	102900005118824	0.251	10.00	销售	否
13	2020-07-01	09:24:21.833	102900005115984	0.251	6.00	销售	否
14	2020-07-01	09:24:21.905	102900005116530	0.217	18.00	销售	否
15	2020-07-01	09:24:57.873	102900005115984	0.468	6.00	销售	否
16	2020-07-01	09:25:31.342	102900005116226	0.589	8.00	销售	否
17	2020-07-01	09:25:45.811	102900005118824	0.711	10.00	销售	否
18	2020-07-01	09:26:04.530	102900005115779	1.003	8.00	销售	否
19	2020-07-01	09:28:01.983	102900005116530	0.095	18.00	销售	否
20	2020-07-01	09:30:17.045	102900005116233	0.150	10.00	销售	否

数据集2： 约87万条记录

数据的合并、分组、聚合——使用pandas

```
import pandas as pd
import matplotlib.pyplot as plt
from pylab import *
mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False

df1 = pd.read_excel("2023C_data1.xlsx")
df2 = pd.read_excel("2023C_data2.xlsx")
###  按单品编码合并两张表
df12 = pd.merge(df1, df2, how='outer', on='单品编码')
###  将各单品每天的销售流水记录放在一组，  按单品编码排序
grouped = df12.groupby(['单品编码','销售日期'],
                       as_index=False)
###  在分好的组内计算总销量，即为各单品每天的总销量
total_sales = grouped['销量(千克)'].sum()
```

total_sales - DataFrame

Index	单品编码	销售日期	销量(千克)
0	102900005115168	2020-10-30 00:00:00	6.427
1	102900005115168	2020-10-31 00:00:00	7.251
2	102900005115168	2020-11-01 00:00:00	13.126
3	102900005115168	2020-12-04 00:00:00	5.972
4	102900005115168	2021-03-03 00:00:00	11.568
5	102900005115168	2021-03-04 00:00:00	22.605
6	102900005115168	2021-03-05 00:00:00	21.036
7	102900005115168	2021-03-06 00:00:00	18.964
8	102900005115168	2021-03-07 00:00:00	21.69
9	102900005115168	2021-03-09 00:00:00	7.072
10	102900005115168	2021-03-14 00:00:00	5.208
11	102900005115168	2021-05-05 00:00:00	11.395
12	102900005115168	2021-05-06 00:00:00	6.231
13	102900005115168	2021-05-07 00:00:00	7.029
14	102900005115168	2021-05-08 00:00:00	6.771
15	102900005115168	2021-05-09 00:00:00	11.368
16	102900005115168	2021-05-10 00:00:00	17.331
17	102900005115168	2021-05-11 00:00:00	16.672
18	102900005115168	2021-05-12 00:00:00	19.955
19	102900005115168	2021-05-13 00:00:00	6.871

数据的合并、分组、聚合——使用pandas

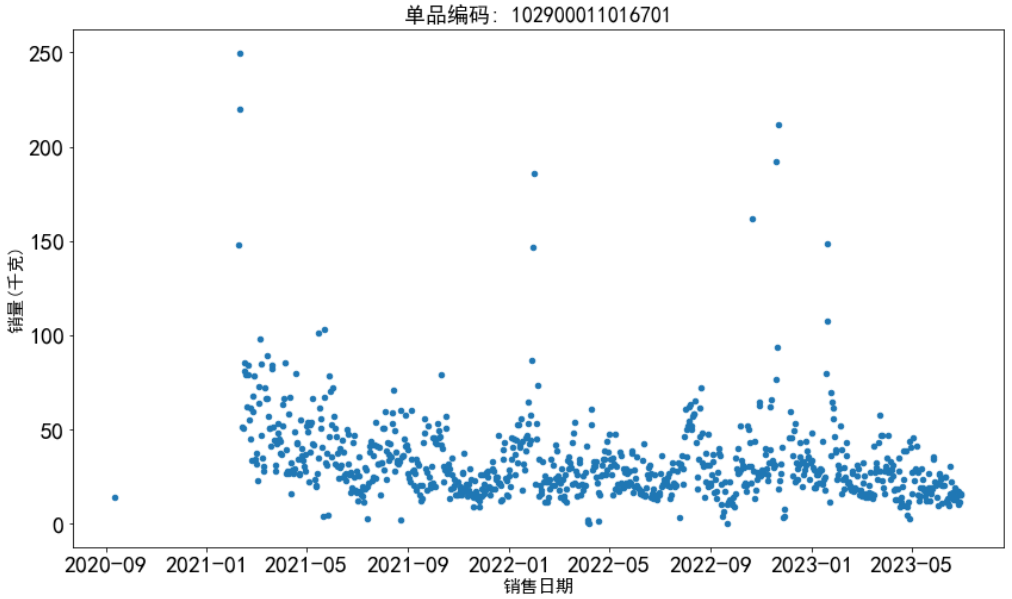
```
### 在分好的组内计算总销量，即为各单品每天的总销量
total_sales = grouped['销量(千克)'].sum()
### 计算各单品在所有日子里的总销量
total_sales1 = total_sales.groupby(['单品编码']).sum()
### 将单品销量从高到低排序
total_sales1.sort_values(by=['销量(千
克)'],ascending=False,inplace=True)

##### 依次画出总销量排名前四的单品的“销量——时间”趋势图
for i in range(4):
  df = \
  total_sales[total_sales['单品编码']==total_sales1.index[i]]
  df = df.iloc[:,1:]
  df.plot(x='销售日期',y='销量(千克)',
          kind = 'scatter',
          figsize=(14,8),
          fontsize=20)
  plt.title('单品编码: '+str(total_sales1.index[i]))
  plt.savefig('aa'+str(i)+ '.png', dpi=300)
  plt.show()
```

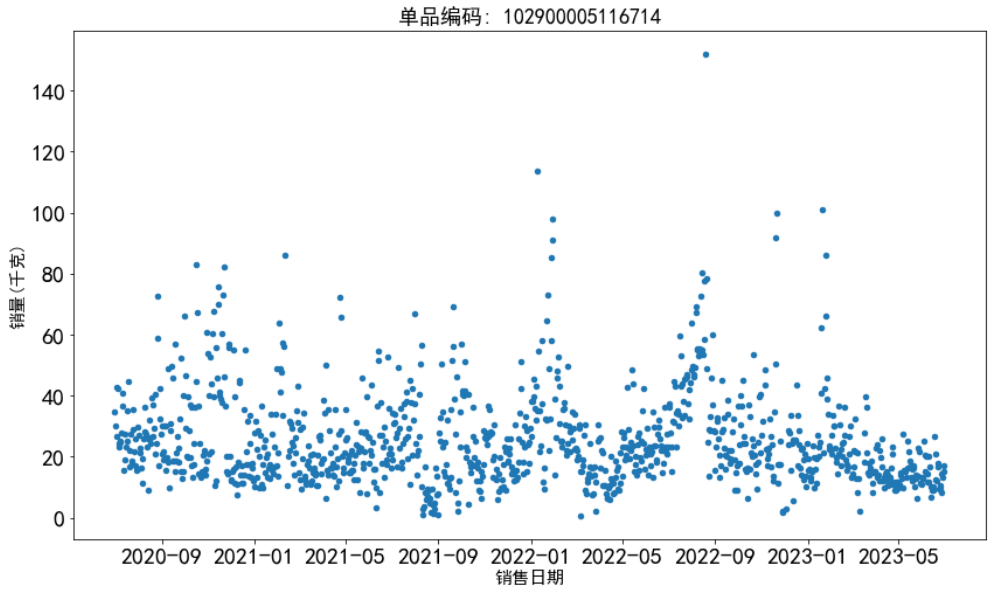
total_sales1 - DataFrame

单品编码	销量(千克)
102900011016701	28164.3
102900005116714	27537.2
102900005116899	27149.4
102900005115960	19187.2
102900005115779	15910.5
106949711300259	15596
102900011030059	14325
102900005116257	13602
102900005116530	11920.2
102900011031100	10833
102900005115984	10305.4
102900005117056	9703.12
102900005118831	8982
102900011030097	8848
102900011009970	8393.79
102900011032251	8235

数据的合并、分组、聚合——使用pandas

```
### 在分好的组内计算总销量，即为各单品每天的总销量
total_sales = grouped['销量(千克)'].sum()
### 计算各单品在所有日子里的总销量
total_sales1 = total_sales.groupby(['单品编码']).sum()
### 将单品销量从高到低排序
total_sales1.sort_values(by=['销量(千
克)'],ascending=False,inplace=True)

##### 依次画出总销量排名前四的单品的“销量——时间”趋势图
for i in range(4):
  df = \
  total_sales[total_sales['单品编码']==total_sales1.index[i]]
  df = df.iloc[:,1:]
  df.plot(x='销售日期',y='销量(千克)',
          kind = 'scatter',
          figsize=(14,8),
          fontsize=20)
  plt.title('单品编码: '+str(total_sales1.index[i]))
  plt.savefig('aa'+str(i)+ '.png', dpi=300)
  plt.show()
```





数据特征	时间序列预测模型	备注
平稳无趋势、无季节性	移动平均 → 简单指数平滑 → ARIMA (0,0,q)	快速建模，作为基准
有趋势、无季节性	Holt 线性趋势模型 → ARIMA (p,1,q) → GRU	优先传统模型，数据量小时避免深度学习
有趋势 + 季节性	Holt-Winters → SARIMA → Prophet	Prophet 无需手动调参，效率最高
多变量联动预测	VAR → VARMA → 多变量 LSTM	先验证变量间相关性（如 Pearson 相关系数）
数据量少（<500 个观测值）	ARIMA/SARIMA → Prophet → 指数平滑	深度学习模型易过拟合，不推荐
数据量多（≥1000 个）	LSTM/GRU → TCN → Transformer + 组合模型	重点优化深度学习模型的泛化能力
含节假日 / 特殊事件	Prophet（自定义节假日） → SARIMA（加入虚拟变量）	需手动标注特殊事件时间点

时间序列预测模型ARIMA的基本思想：

见示例文件 time_series_demo.py

机器学习

机器学习	任务类型	Python包
监督学习	回归	scikit-learn（适用于非深度学习）
	分类	Pytorch/Tensorflow（适用于深度学习）
无监督学习	聚类/降维/特征学习/生成模型	scikit-learn/Pytorch
强化学习	**NA**	**NA**

机器学习——监督学习——回归任务 $f\colon\ x \in \mathbb{R}^p \longrightarrow y = f(x) + \varepsilon \in \mathbb{R}^q$（$\varepsilon$为随机噪声）

在观测到n各输入-输出对$D = \{(x_i, y_i)\}_{i=1}^n$的前提下，选择回归模型$\hat{f}(x;\theta)$，$\theta$为可调参数集合，使得$\hat{f}(x;\theta)$在某种度量下很好地逼近真实的却一般无法知晓表达式的$f(x)$.

（1）训练：$\theta^* = argmin_\theta Loss(\theta) = argmin_\theta \frac{1}{n}\sum_{i=1}^n \left[\hat{f}(x_i;\theta) - f(x_i)\right]^2$（这里选择均方误差作为损失函数，也可以是其它形式）

（2）预测：$\forall x^* \in \mathbb{R}^p,\ y^*(x^*) = \hat{f}(x^*;\theta^*)$

回归模型	数学本质	损失函数	Python包
（广义）线性回归	$\sum_{j=1}^m c_j\phi_j(x) + c_0$，$\phi_j(x)$为事先指定的基函数	均方误差	scikit-learn
高维问题（$p \gg 1$）时的变体：Ridge回归（岭回归）	$\sum_{j=1}^m c_j\phi_j(x) + c_0$ $\phi_j(x)$为事先指定的基函数	均方误差+某个正实数乘以c_j构成向量的L2范数	scikit-learn
高维问题（$p \gg 1$）时的变体：Ridge回归（岭回归）	$\sum_{j=1}^m c_j\phi_j(x) + c_0$，$\phi_j(x)$为事先指定的基函数	均方误差+某个正实数乘以c_j构成向量的L1范数	scikit-learn
决策树/随机森林	分片常数拟合，分片点自动计算+集成学习	均方误差	scikit-learn
支持向量机	无穷个基函数的线性回归	分割平面将两类点分开的程度（最大化）	scikit-learn
深度神经网络	“仿射变换+非线性激活函数”的多轮嵌套	均方误差	pytorch

机器学习——监督学习——回归任务

标准单变量线性回归： $\hat{f}(x;\theta) = c_1 x + c_0$
（注意scikit-learn这个包在python中改名为sklearn）

```
from sklearn import linear_model
import numpy as np
#调用标准线性模型类， 生成模型实例
model = linear_model.LinearRegression()
# 调用模型的成员函数fit()，进行训练
model.fit([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
# 给出测验点
x_test = np.array([[0.5, 0.3]])
# 调用模型的成员函数predict(),进行预测
y_test = model.predict(x_test)
print(y_test)
```

scikit-learn调包套路

选择回归函数：
从scikit-learn调出模型对应的类，如这里的linear_model.LinearRegression()，并利用这个类实例化一个模型对象，如model

↓

训练： 调用类对象model的成员函数fit()

↓

预测： 调用类对象model的成员函数predict()

机器学习——监督学习——回归任务

Lasso回归： $\hat{f}(x;\theta) = c_1 x + c_0$

$$Loss(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left[\hat{f}(x_i;\theta) - f(x_i)\right]^2 + \lambda(|c_0| + |c_1|)$$

```
>>> from sklearn import linear_model
>>> reg = linear_model.Lasso(alpha=0.1)
>>> reg.fit([[0, 0], [1, 1]], [0, 1])
Lasso(alpha=0.1)
>>> reg.predict([[1, 1]])
array([0.8])
```

scikit-learn实现12种回归模型的例子

scikit-learn调包套路

选择回归函数：
从scikit-learn调出模型对应的类，如这里的linear_model.Lasso()，并利用这个类实例化一个模型对象，如model

↓

训练： 调用类对象model的成员函数fit()

↓

预测： 调用类对象model的成员函数predict()

机器学习——监督学习——分类任务

$$f:\ x+\varepsilon \in \mathbb{R}^p \longrightarrow y \in \{C_1, C_2, \ldots, C_K\}$$

这是一个K元分类问题，比如MNIST手写数字数据集。分类模型的构建大致有两种思路：

（1）条件概率： 构造合理的条件概率$\hat{P}(Y=y|X=x;\theta) \approx P((Y=y|X=x))$

（2）概率向量（转化为回归问题）： $y \in \{C_1, C_2, \ldots, C_K\} \longrightarrow \tilde{y} \in \mathbb{R}^K$(例如通过SoftMax)

第一种思路构造的分类模型多通过极大似然估计训练，第二种的分类模型一般以交叉熵作为损失函数

分类模型	模型类型	损失函数	Python包
Logistic模型、LDA和QDA模型、朴素贝叶斯	条件概率	边际似然函数	scikit-learn
决策树/随机森林	条件概率	基尼系数、信息熵、纯度	scikit-learn
支持向量机	NA	分割平面将两类点分开的程度（最大化）	scikit-learn
卷积神经网络/U-net/ResNet	概率向量	交叉熵	pytorch

机器学习——监督学习——分类任务

```
### 支持向量机做二元分类
from sklearn import svm
X = [[0, 0], [1, 1]]
y = [0, 1]
model = svm.SVC()
model.fit(X, y)
print(model.predict([[2., 2.]]))


### 单棵决策树做二元分类
from sklearn import tree
X = [[0, 0], [1, 1]]
Y = [0, 1]
model = tree.DecisionTreeClassifier()
model.fit(X, Y)
print(model.predict([[2., 2.]]))
```

scikit-learn调包套路

选择回归函数：
从scikit-learn调出模型对应的类，如这里的svm.SVC()，并利用这个类实例化一个模型对象，如model

↓

训练： 调用类对象model的成员函数 fit()

↓

预测： 调用类对象model的成员函数 predict()

机器学习——无监督

```
from sklearn.cluster import KMeans
import numpy as np
X = np.array([[1, 2], [1, 4], [1, 0],
              [10, 2], [10, 4], [10, 0]])
model = KMeans(n_clusters=2,
random_state=0)
kmeans = model.fit(X)
print(kmeans.labels_)
print(kmeans.predict([[0, 0], [12, 3]]))
print(kmeans.cluster_centers_)
```

更多无监督模型的scikit-learn实现请见：
scikit-learn中文社区

scikit-learn调包套路

选择回归函数：
从scikit-learn调出模型对应的类，如这里的KMeans(n_clusters=2, random_state=0)，并利用这个类实例化一个模型对象，如model

↓

训练： 调用类对象model的成员函数fit()

↓

预测： 调用类对象model的成员函数predict()

小结：**重复！重复！重复！**时间序列预测+机器学习+最优化算法（基于梯度的算法和启发式的算法、动态规划）

这个世界没有什么天才，大部分的成功都来自于极致的重复。

Python基础+进阶	学习任务细化	需要掌握的Python包
Python语言基础	解释性语言	—
	IDE/文件/模块	—
	程序控制语句/函数	—
	画图 & 强推的Python在线教程	matplotlib、 seaborn
Python科学计算	数组与矩阵	numpy
	插值	scipy
	数值积分	scipy
	优化	scipy
Python数据科学	数据预处理： 缺失值/异常值处理、归一化、标准化 合并、分组与聚合	numpy/pandas/scikit-learn
	时间序列预测	statsmodels/pytorch
	机器学习： 监督学习（回归与分类） 无监督学习（降维、概率密度估计）	scikit-learn pytorch或tensorflow

年份	题目	研究内容	数学模型	算法
2024	MCM A题	研究海洋鳗鳜性别比例与资源可用性的关系，开发模型探讨其优劣势	Lotka-Volterra模型、费舍尔性别比例理论、响应曲线模型、蒙特卡洛模拟	粒子群优化(PSO)、贝叶斯推断、A*搜索、模拟退火
	MCM B题	定位失踪潜水器，准备搜索设备，确定搜索模式等	随机过程模型、贝叶斯网络模型、路径规划模型	A*算法、Dijkstra算法、粒子滤波、蒙特卡洛模拟
	MCM C题	开发模型捕捉网球比赛流程，评估教练关于比赛动力的假设	动力学方程、马尔科夫链、回归分析	神经网络、支持向量机、梯度下降法、蒙特卡洛方步
	ICM D题	为五大湖及其连接河流流量建立网络模型	水文模型、随机过程、流量分配模型	网络流优化(Ford-Fulkerson)、Dijkstra算法、线性规划
	ICM E题	确定如何最佳定位房产风险	风险理论模型、蒙特卡洛模拟、定价模型	最大似然估计、决策树、最优化算法
	ICM F题	减少非法野生动物贸易的相关问题	博弈论、网络模型、社会网络分析	社会网络分析算法、博弈论模型、最大流算法

2023年	MCM A题	受干旱影响的植物群落，建立数学模型预测植物群落随时间变化	Lotka-Volterra模型、种群动力学模型	时间序列分析、回归分析、蒙特卡洛模拟
	MCM B题	重新构想马赛马拉，确定管理马赛马拉保护区资源的替代方法	资源分配优化模型、线性规划、动态规划	多目标优化算法、线性规划、遗传算法
	MCM C题	预测Wordle结果	回归分析、马尔科夫链、蒙特卡洛模拟	蒙特卡洛方法、支持向量机、随机森林
	MCM D题	优先考虑联合国可持续发展目标	多目标优化模型、线性规划	非线性优化、动态规划、遗传算法
	MCM E题	光污染	光源优化模型、最优化算法	遗传算法、模拟退火算法、动态规划
	MCM F题	绿色GDP	绿色经济模型、回归分析	最优化算法、遗传算法、模拟退火

2022年	MCM A题	肿瘤生长与治疗相关问题	Logistic回归、肿瘤生长模型、药物剂量响应模型	最优化算法、回归分析、数值求解方法
	MCM B题	沙堡的建造和稳定性问题	物理建模、有限元分析、力学模型	数值解法、蒙特卡洛模拟、有限元法
	MCM C题	数据处理和分析，评估与预测	统计分析、回归分析、机器学习	聚类算法、回归分析、K-均值聚类
	ICM D题	物流配送网络的优化	最短路径算法、线性规划、网络流优化	Dijkstra算法、Ford-Fulkerson算法、分支限界法
	ICM E题	水资源的管理和分配	线性规划、水文模型	决策树、回归分析、蒙特卡洛模拟
	ICM F题	促进可再生能源发展的政策制定	经济学模型、多目标优化	政策分析模型、最优化算法、线性规划

2021年	MCM A题	淡水湖的流入流出量及水位变化的建模分析	水文模型、流量分析模型、最优化算法	线性规划、动态规划、遗传算法
	MCM B题	夏威夷海洋保护区最佳监测方案	动态规划、资源分配模型	线性规划、启发式搜索、粒子群优化
	MCM C题	电动汽车对电网的影响	电力系统模型、负荷预测模型	最优化算法、回归分析、动态系统仿真
	ICM D题	音乐演变分析	时间序列分析、马尔科夫链	主成分分析、隐马尔科夫模型，网络流算法
	ICM E题	减少食物浪费的策略	多目标优化、回归分析	线性规划、模拟退火算法、遗传算法
	ICM F题	疫苗分配策略优化	博弈论、最优化算法	博弈论算法、最优化算法、数据挖掘

2020年	MCM A题	帝王蝶的迁徙模式分析	迁徙模型、最短路径算法、随机过程模型	最优化算法、粒子群优化、遗传算法
	MCM B题	海底稀土元素分布及开采策略	资源分配优化、经济模型	动态规划、最优化算法、模拟退火
	MCM C题	篮球比赛的精品阵容和战术安排	数学规划、回归分析	最优化算法、遗传算法、动态规划
	ICM D题	自然灾害对电力系统的影响	电力系统恢复模型、最优化算法	网络流优化、分支限界法、遗传算法
	ICM E题	全球气候变化对海洋生态系统的影响	生态系统模型、气候变化预测模型	数值解法、回归分析、蒙特卡洛模拟
	IOM F题	提高公众对可持续发展的参与度	行为预测模型、社会网络分析	社交网络分析、聚类算法、回归分析

AI辅助编程

- AI只能锦上添花，很难雪中送炭；
- 程序大框架由人来定；局部编程细节可由AI辅助，而且需要对AI的代码进行及时Debug；
- 在AI使用报告中诚实地说明如何利用AI辅助编程。