

Advanced Data Analytics Using Python
With Machine Learning, Deep Learning and NLP Examples

Python高级数据分析

机器学习、深度学习和NLP实例

[印] 萨扬·穆霍帕迪亚 (Sayan Mukhopadhyay) 著

罗佳 译

包含数据分析实例

涵盖了从基础统计学到ETL、深度学习和物联网的广泛领域
给出了产业分析项目各个技术方面的概念



机械工业出版社
China Machine Press

数据分析与决策

技术丛书

Advanced Data Analytics Using Python
With Machine Learning, Deep Learning and NLP Examples

Python高级数据分析

机器学习、深度学习和NLP实例

[印] 萨扬·穆霍帕迪亚 (Sayan Mukhopadhyay) 著

罗佳 译



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

Python 高级数据分析: 机器学习、深度学习和 NLP 实例 / (印) 萨扬·穆霍帕迪亚 (Sayan Mukhopadhyay) 著; 罗佳译. —北京: 机械工业出版社, 2019.1

(数据分析与决策技术丛书)

书名原文: Advanced Data Analytics Using Python: With Machine Learning, Deep Learning and NLP Examples

ISBN 978-7-111-61702-0

I. P… II. ① 萨… ② 罗… III. 软件工具 - 程序设计 IV. TP311.561

中国版本图书馆 CIP 数据核字 (2018) 第 297098 号

本书版权登记号: 图字 01-2018-8015

First published in English under the title

Advanced Data Analytics Using Python: With Machine Learning, Deep Learning and NLP Examples

by Sayan Mukhopadhyay

Copyright © Sayan Mukhopadhyay, 2018

This edition has been translated and published under licence from APress Media, LLC, part of Springer Nature.

Chinese simplified language edition published by China Machine Press, Copyright © 2019.

This edition is licensed for distribution and sale in the People's Republic of China only, excluding Hong Kong, Taiwan and Macao and may not be distributed and sold elsewhere.

本书原版由 Apress 出版社出版。

本书简体字中文版由 Apress 出版社授权机械工业出版社独家出版。未经出版者预先书面许可, 不得以任何方式复制或抄袭本书的任何部分。

此版本仅限在中华人民共和国境内 (不包括香港、澳门特别行政区及台湾地区) 销售发行, 未经授权的本书出口将被视为违反版权法的行为。

Python 高级数据分析 机器学习、深度学习和 NLP 实例

出版发行: 机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码: 100037)

责任编辑: 刘 锋

责任校对: 殷 虹

印 刷: 北京文昌阁彩色印刷有限责任公司

版 次: 2019 年 1 月第 1 版第 1 次印刷

开 本: 186mm × 240mm 1/16

印 张: 10.5

书 号: ISBN 978-7-111-61702-0

定 价: 59.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88379426 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzsj@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

华章 IT
HZBOOKS | Information Technology



作为当今社会的热门职位，数据科学家通过对大量数据的合理使用，引起了一大批新的应用甚至是新的行业的产生。作为数据科学家使用的方法和工具，数据分析技术通过对收集来的大量数据进行详细研究和概括总结，让数据开口说话，从数据中提取有用信息并形成相应的结论，最终帮助人们将数据转化为可以付诸行动的见解。

鉴于已有许多数据分析方面的入门型书籍，本书侧重于从各个方面展示数据分析的高级内容。基于此，本书包含了数据分析领域较全面的方法和技术，包括最新的数据库技术、监督学习方法、无监督学习方法、深度学习和神经网络、时间序列以及大数据分析等内容。本书的另一特色在于给出了大量的实例，便于读者在实例基础上深入理解相关内容和方法，并在自己的项目中引用这些实例作为示例代码。

本书适合在数据分析领域已有一定基础，需要进一步提高的读者。

感谢华章公司的刘锋编辑不辞辛苦地和我沟通相关细节内容，同时感谢他在翻译本书过程中给予的诸多帮助。

限于本人水平，难免会对本书中部分内容的理解或中文语言表达存在不当之处，敬请读者批评指正，以便能够不断改进。

罗佳

2018年10月15日于上海

作者简介



Sayan Mukhopadhyay 拥有超过13年的行业经验，并与瑞信、PayPal、CA Technologies、CSC 和 Mphasis 等公司建立了联系。他对投资银行、在线支付、在线广告、IT 架构和零售等领域的数据分析应用有着深刻的理解。他的专业领域是在分布

式和数据驱动的环境（如实时分析、高频交易等）中，实现高性能计算。

他在印度贾达普大学获得电子和仪器工程学位，在班加罗尔的印度科学研究所获得硕士学位，研究方向为计算和数据科学。

| 技术审核员简介 |

Sundar Rajan Raman 在机器学习、深度学习和自然语言处理方面拥有超过 14 年的全栈 IT 经验。他拥有 6 年的大数据开发和架构经验，包括使用 Hadoop 及其生态系统以及 MongoDB 和 Cassandra 等其他 NoSQL 技术。事实上，他一直是关于这些主题的书籍的技术审核员。他也对使用设计思维原则的策略以及指导别人感兴趣。



致谢

感谢 Labonic Chakraborty (Ripa) 和 Kusumika Mukherjee。

| 目录 |

译者序

作者简介

技术审核员简介

致谢

第 1 章 简介	001
1.1 为何选择 Python	001
1.2 何时避免使用 Python	002
1.3 Python 中的面向对象编程	002
1.4 在 Python 中调用其他语言	010
1.5 将 Python 模型作为微服务	011
1.6 高性能 API 和并发编程	014
第 2 章 Python 结构化数据提取、转换和加载	019
2.1 MySQL	020
2.1.1 如何安装 MySQLdb	020
2.1.2 数据库连接	020
2.1.3 INSERT 操作	020
2.1.4 READ 操作	021
2.1.5 DELETE 操作	022

2.1.6	UPDATE 操作	023
2.1.7	COMMIT 操作	023
2.1.8	ROLL-BACK 操作	024
2.2	Elasticsearch	026
2.3	Neo4j Python 驱动	029
2.4	neo4j-rest-client	029
2.5	内存数据库	029
2.6	Python 版本 MongoDB	030
2.6.1	将数据导入集合	031
2.6.2	使用 pymongo 创建连接	031
2.6.3	访问数据库对象	032
2.6.4	插入数据	032
2.6.5	更新数据	032
2.6.6	删除数据	032
2.7	Pandas	033
2.8	Python 非结构化数据提取、转换和加载	034
2.8.1	电子邮件解析	034
2.8.2	主题爬取	036
第 3 章	基于 Python 的监督学习	043
3.1	使用 Python 实现降维	043
3.1.1	相关性分析	044
3.1.2	主成分分析	046
3.1.3	互信息	048
3.2	使用 Python 进行分类	049
3.3	半监督学习	050
3.4	决策树	050
3.4.1	哪个属性优先	050
3.4.2	随机森林分类器	052
3.5	朴素贝叶斯分类器	052
3.6	支持向量机	054

3.7	最近邻分类器	055
3.8	情绪分析	056
3.9	图像识别	057
3.10	使用 Python 进行回归	058
	3.10.1 最小二乘估计	059
	3.10.2 逻辑回归	060
3.11	分类和回归	060
3.12	使模型高估或低估	061
3.13	处理分类型数据	062
第 4 章	无监督学习—聚类	067
4.1	K 均值聚类	068
4.2	选择 K —肘部法则	071
4.3	距离或相似性度量	071
	4.3.1 属性	072
	4.3.2 一般及欧氏距离	072
	4.3.3 平方欧氏距离	074
	4.3.4 字符串之间的编辑距离	074
4.4	文档上下文的相似性	076
4.5	什么是层次聚类	077
	4.5.1 自下而上的方法	078
	4.5.2 聚类之间的距离	079
	4.5.3 自上而下的方法	080
	4.5.4 图论方法	084
4.6	如何判断聚类结果是否良好	085
第 5 章	深度学习和神经网络	087
5.1	反向传播	088
	5.1.1 反向传播方法	088
	5.1.2 广义 Delta 规则	088
	5.1.3 输出层权重更新	089

5.1.4	隐藏层权重更新	090
5.1.5	反向传播网络小结	091
5.2	反向传播算法	092
5.3	其他算法	094
5.4	TensorFlow	094
5.5	递归神经网络	099
第 6 章	时间序列	107
6.1	变化的分类	107
6.2	包含趋势的序列分析	107
6.2.1	曲线拟合	108
6.2.2	从时间序列中去除趋势	109
6.3	包含周期性的序列数据分析	110
6.4	从时间序列中去除周期性	111
6.4.1	滤波	111
6.4.2	差分	112
6.5	转换	112
6.5.1	稳定方差	112
6.5.2	使周期效应累加	113
6.5.3	使数据呈正态分布	113
6.6	平稳时间序列	114
6.6.1	平稳过程	114
6.6.2	自相关和相关图	114
6.6.3	自协方差和自相关函数的估计	115
6.7	使用 Python 进行时间序列分析	116
6.7.1	有用的方法	116
6.7.2	自回归过程	118
6.7.3	估计 AR 过程的参数	119
6.8	混合 ARMA 模型	122
6.9	集成 ARMA 模型	123

6.10	傅里叶变换	124
6.11	一个特殊的场景	125
6.12	数据缺失	127
第 7 章	大数据分析	129
7.1	Hadoop	129
7.1.1	MapReduce 编程	129
7.1.2	partitioning 函数	130
7.1.3	combiner 函数	131
7.1.4	HDFS 文件系统	140
7.1.5	MapReduce 设计模式	140
7.2	Spark	146
7.3	云分析	148
7.4	物联网	156

1

第 1 章

简介

本书假设读者已经熟悉了 Python 编程的基本内容。在这个介绍性的章节中，将解释为什么数据科学家应选择 Python 作为编程语言，同时强调在哪些情况下 Python 不是一个好的选择。最后，描述了应用程序开发中的一些好的实例，并给出了数据科学家在日常工作中需要的一些编码示例。

1.1 为何选择 Python

那么，为什么要选择 Python 呢？

- Python 具有很多的库。对于任何类型的应用，你总能在 Python 中找到一个现成的库。从统计编程到深度学习，再到网络应用、网络爬虫以及嵌入式系统，总有一个用 Python 编写的库。如果你学习这种语言，则不必盯着特定的范例。R 虽然拥有丰富的分析库，但如果你正在编写物联网 (IoT) 应用程序，并且需要在设备端嵌入式系统中进行编码，那么用 R 会很困难。

- ❑ Python 有非常高的性能。Java 也是一种通用语言，拥有大量库，但 Java 代码在 Java 虚拟机上运行，这增加了额外的延迟层。Python 使用其他语言构建的高性能库，例如，SciPy 使用 LAPACK，它是线性代数应用程序的 Fortran 库。TensorFlow 使用 CUDA，它是用于 GPU 并行处理的 C 语言库。
- ❑ Python 很简洁，为你提供了很多编码自由。Python 语法就像一种自然语言，很容易记住，并且它没有关于变量的约束（如常量或公共 / 私有变量）。



1.2 何时避免使用 Python

Python 也有一些缺点：

- ❑ 当编写非常特殊的代码时，Python 可能并不是最佳选择。例如，如果你编写的代码仅处理统计信息，则 R 是更好的选择。如果你只编写 MapReduce 代码，那么 Java 是比 Python 更好的选择。
- ❑ Python 为你提供了很多编码上的自由。因此当许多开发人员同时进行大型应用程序的开发时，Java/C++ 是更好的选择，因为一个开发人员或架构师可以使用公共 / 私有和常量关键字对其他开发人员的代码进行约束。
- ❑ 对于要求极高性能的应用程序，除了 C / C++ 之外别无选择。



1.3 Python 中的面向对象编程

在开始讲述这门语言之前，我将解释面向对象编程（OOP）在 Python 中的一些特性。

现代应用程序的最基本元素是对象。对于程序员或架构师来说，世界是一个对象的集合。对象由两种类型的成员组成：属性和方法。成员可以是私有的、公共的或受保护的。类是对象的数据形式。每个对象都是一个类的实例，类可以被子类继承，两个类可以使用复合（composition）关联起来。

在 Python 中，没有 `public`、`private` 或 `protected` 关键字，因此封装（在外部世界中将成员隐藏）并不包含在 Python 中。与 C++ 一样，它支持多级和多继承。与 Java 一样，它有一个 `abstract` 关键字。类和方法都可以是抽象的。

下面的代码是一个通用网络爬虫程序的示例，它可以作为在 Skytrax 网站爬取航空公司数据的网络爬虫程序，也可用作爬取 Mouthshut.com 网站的零售数据的爬虫程序。我们将在第 2 章中阐述网络爬虫这个主题。

```

from abc import ABCMeta, abstractmethod
import BeautifulSoup
import urllib
import sys
import bleach
##### Root Class (Abstract) #####
class SkyThoughtCollector(object):
    __metaclass__ = ABCMeta

    baseURLString = "base_url"
    airlinesString = "air_lines"
    limitString = "limits"

baseURL = ""
airlines = []
limit = 10

@abstractmethod
def collectThoughts(self):
    print "Something Wrong!! You're calling
        an abstract method"

@classmethod

```