大数据、云计算、人工智能、信息安全人才培养丛书

“互联网+”新形态一体化精品教材

# 机器学习

JIQI XUEXI

主　编◎谢　椿　戴　敏　李文强

扫一扫
学习资源库

- ◆微课视频
- ◆教学课件
- ◆电子教案
- ◆考试题库

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

大数据、云计算、人工智能、信息安全人才培养丛书
“互联网+”新形态一体化精品教材

# 机器学习

JIQI XUEXI

主　编　谢　椿　戴　敏　李文强
副主编　陈　联　戴发术

扫一扫
学习资源库

◆微课视频
◆教学课件
◆电子教案
◆考试题库

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

内容提要

机器学习是人工智能的重要技术基础，涉及的内容十分广泛。本书内容涵盖机器学习的基础知识和高级应用，主要包括初识机器学习、Python 语言在机器学习中的应用、线性回归模型及应用、逻辑回归模型及应用、贝叶斯模型及应用、$k$ 近邻模型及应用、决策树模型及应用、隐马尔可夫模型及应用、支持向量机模型及应用、推荐系统算法及应用、主成分分析算法及应用、机器学习工程实践。

本书可作为高等院校机器学习、数据分析、数据挖掘等课程的教材，也可作为机器学习爱好者的参考用书，同时对程序员、数据分析师、统计学家、数据科学家解决实际问题也有参考价值。

图书在版编目（CIP）数据

机器学习 / 谢椿，戴敏，李文强主编 . — 上海：上海交通大学出版社，2020（2022.10 重印）

ISBN 978-7-313-23405-6

Ⅰ. ①机… Ⅱ. ①谢… ②戴… ③李… Ⅲ. ①机器学习 Ⅳ. ① TP181

中国版本图书馆 CIP 数据核字（2020）第 104954 号

机器学习
JIQI XUEXI

主　　编：谢椿　戴敏　李文强
出版发行：上海交通大学出版社
邮政编码：200030
印　　制：北京荣玉印刷有限公司
开　　本：889 mm × 1194 mm　1/16
字　　数：252 千字
版　　次：2020 年 7 月第 1 版
书　　号：978-7-313-23405-6
定　　价：49.80 元

地　　址：上海市番禺路 951 号
电　　话：6407 1208

经　　销：全国新华书店
印　　张：11.5

印　　次：2022 年 10 月第 2 次印刷

# 大数据、云计算、人工智能、信息安全人才培养丛书

| 人工智能系列专家团队 | | |
|---|---|---|
| 姓名 | 职称 | 研究方向 |
| 刘增良 | 教授 | 人工智能 |
| 祝　恩 | 教授 | 人工智能深度学习 |
| 杨欣亮 | 教授 | 人工智能语音识别 |
| 邵泽德 | 教授 | 智能控制 |
| 刘文贞 | 教授 | 智能应用 |
| 俞俊承 | 教授 | 机器人 imrobotic |
| 孙东辰 | 教授 | 人工智能安全 |
| 吴　坚 | 教授 | 工业互联 |
| 冯德川 | 教授 | 工控安全 |
| 杨晓光 | 教授 | 智能控制 |
| 王文辉 | 教授 | 工业网络 |
| 侍欢迎 | 教授 | 智能终端 |
| 李书生 | 教授 | 人脸识别 |
| 王贤刚 | 教授 | 语音识别 |
| 王一粟 | 教授 | 计算机视觉 |
| 邓三鹏 | 教授 | 自动化运维 |
| 高　峰 | 教授 | 协作机器人 |
| 周旺发 | 教授 | 智能控制 |
| 张海蛟 | 教授 | 工业网络 |
| 陈昌安 | 教授 | 智能终端 |
| 彭泳群 | 教授 | 人脸识别图像识别 |

# 序

人类社会正处在数字化、网络化、智能化快速推进的时代。社会运行与网络空间深度融合，从根本上改变着人们的生产生活方式，重塑着社会发展的新格局。全球主要国家都在加强人工智能布局，抢占智能经济制高点。2019 年以来，我国更加重视人工智能技术应用落地，先后批复启动国家新一代人工智能创新发展试验区、人工智能创新应用先导区，发挥地方主体作用，鼓励支持先行先试，探索促进人工智能与经济社会发展深度融合的新路径。在新一轮产业升级和生产力飞跃过程中，人才作为第一位的资源将发挥关键作用。

新工科代表着新的生产力、新的发展方向。随着人工智能产业的高速发展，社会对人工智能应用型人才产生了迫切需求，也促使高等教育产生革命性变化。高等院校的教育模式、形态、内容和学习方式都发生深刻变革，开始以学习者为中心，以应用需求为目标，注重能力培养，促进人的全面发展。个性化学习的理念日益深入人心。

高等院校贯彻《新一代人工智能发展规划》，为产业、行业培养更多高素质的人工智能应用型人才，急需拓宽人工智能专业教育内容，推动人工智能与计算机、信息、数学等学科专业教育的交叉融合，提高学生的实践运用能力，提升院校计算机相关专业学生的就业竞争力，赋能高等教育创新培育能力，打造人工能智能人才培养及智能经济战略高地，提升智能经济建设水平，加快构建人工智能产业生态。

本套大数据、云计算、人工智能、信息安全人才培养丛书中人工智能方向的教材明确了适用专业、培养目标、培养规格、课程体系、师资队伍、教学条件、质量保障等各方面要求，是联合多所高校及多家知名企业共同编撰而成的校企合作教材，充分体现了产教深度融合、校企协同育人的理念；是在校企合作机制和人才培养模式的协同创新下打造的精品教材，既适合高等院校相关专业学生使用，也适用于企业相关岗位专业人才的培养。

中国人民解放军国防大学教授 / 博导
中国人工智能学会智能系统工程专业委员会主任

# 前言

以机器学习为核心的人工智能已经成为新一代生产力发展的主要驱动因素。新的技术正在向各行各业渗透，大有变革各个领域的趋势。传统产业向智慧产业的升级迫使原行业从业人员逐渐转型，市场上对相关学习材料的需求也日益增长。帮助广大学习者更好地理解和掌握机器学习，是编写本书的目的。

本书针对机器学习领域中常见的一类问题——有监督学习，采用项目任务式的体例结构，分 12 个项目介绍机器学习的基础知识和高级应用，包括初识机器学习、Python 语言在机器学习中的应用、线性回归模型及应用、逻辑回归模型及应用、贝叶斯模型及应用、$k$ 近邻模型及应用、决策树模型及应用、隐马尔可夫模型及应用、支持向量机模型及应用、推荐系统算法及应用、主成分分析算法及应用、机器学习工程实践。本书从入门、进阶、深化三个层面，从特殊到普遍，从自然算法到优化算法，从各个角度深入讲解机器学习的相关知识，力求帮助读者循序渐进地掌握机器学习的概念、算法和优化理论。

在编写上，本书具有以下特点：

（1）本书内容翔实、语言流畅、图文并茂、突出实用性，并提供了大量的操作示例和相应代码，较好地将学习与应用结合在一起。内容由浅及深、循序渐进，适合各个层次读者的学习。

（2）本书所引用的实例均与生活密切相关，使读者在学习的时候不会觉得陌生，更容易接受，从而提高学习效率。

（3）本书根据读者的学习特点，通过案例分析辅助知识点分类介绍。考虑到因学生基础参差不齐而给教师授课带来的困扰，本书在编写过程中划分为多个任务，每个任务又划分为多个子任务，以“做”为中心，“教”和“学”都围绕“做”展开，在学中做，在做中学，从而完成知识学习、技能训练，提高学生自我学习、自我管理学习知识体系的能力。

（4）计算机技术发展迅速，本书着重讲解当前主流研究成果和新技术，与行业联系密切，所有内容都紧跟行业技术的发展。

此外，本书作者还为广大一线教师提供了服务于本书的教学资源库，有需要者可致电 13810412048 或发邮件至 2393867076@qq.com。

本书由从事多年机器学习授课的任课教师和相关企业的工程师共同编写而成，可作为高等院校机器学习、数据分析、数据挖掘等课程的教材，也可作为机器学习爱好者的参考用书，同时对程序员、数据分析师、统计学家、数据科学家解决实际问题也有参考价值。

由于计算机技术发展迅速，书中存在的不当和遗漏之处，敬请广大读者批评指正！

# 目录

## 项目 4　逻辑回归模型及应用 / 25



## 项目 5　贝叶斯模型及应用 / 45



## 项目 6　*k* 近邻模型及应用 / 57



## 项目 7　决策树模型及应用 / 71

## 项目 8 隐马尔可夫模型及应用 / 91



## 项目 9 支持向量机模型及应用 / 105



## 项目 10 推荐系统算法及应用 / 125

## 项目 11　主成分分析算法及应用 / 145



## 项目 12　机器学习工程实践 / 163

# 项目 1 初识机器学习

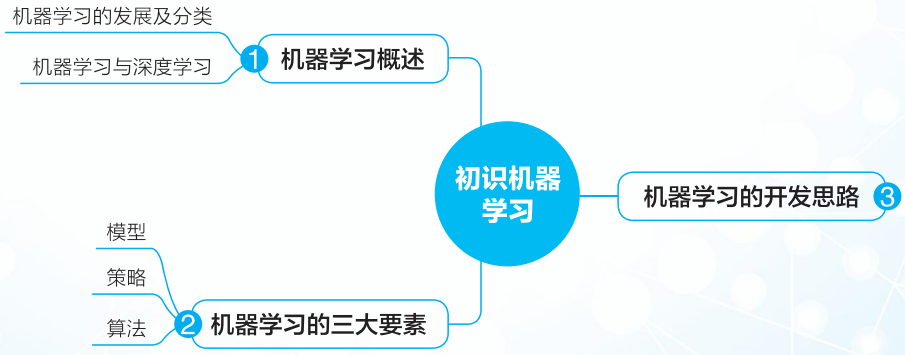
## 知识目标

1. 理解机器学习的基本概念。
2. 了解机器学习的发展历史及分类。
3. 掌握机器学习的三大要素。

## 能力目标

1. 熟悉机器学习应用系统的开发思路。
2. 具有分析机器学习应用系统的能力。
3. 具有开发机器学习应用系统的能力。

## 知识导图



笔记

## 项目导入

自从计算机发明以来，人们就想知道它能不能学习。如果能够理解计算机学习的内在机制，即怎样使它们根据经验提高自身的能力，那么影响将是空前的。想象一下，未来，计算机能从医疗记录中学习，获取治疗新疾病的最有效方法；住宅管理系统可分析住户的用电模式，以降低能源消耗；个人软件助理可跟踪用户的兴趣，并为其选择最感兴趣的在线新闻……对计算机学习的成功理解将开辟出全新的应用领域，并使其计算能力和可定制性上升到新的层次。同时，透彻理解机器学习的信息处理算法，也有助于更好地理解人类的学习能力。

目前，人们还不知道怎样使计算机的学习能力与人类相媲美。然而，一些针对特定学习任务的算法已经产生，关于学习的理论认识已逐步形成。人们开发了很多实践性的计算机程序实现不同类型的学习，一些商业化的应用已经出现。例如，对于语音识别这样的课题，到目前为止，基于机器学习的算法明显胜过其他方法。在数据挖掘领域，机器学习算法理所当然地得到应用，从包含设备维护记录、借贷申请、金融交易、医疗记录等类似信息的大型数据库中发现有价值的信息。随着对计算机理解的日益成熟，机器学习必将在计算机科学和技术中扮演越来越重要的角色。

## 任务 1.1 机器学习概述

机器学习（Machine Learning）是一门多领域交叉学科，涉及概率论、统计学、逼近论、凸分析、算法复杂度理论等多门学科，专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构，使之不断改善自身的性能。简言之，就是计算机从数据中学习规律和模式，应用在新数据上完成预测的任务。

### 子任务 1.1.1 机器学习的发展及分类

#### 1. 机器学习的发展

机器学习可分为 4 个阶段。

第一阶段是 20 世纪 50 年代至 60 年代中期，系统通过自身不断地学习输入、输出反馈，完善系统参数及本身的性能。

第二阶段从 20 世纪 60 年代中期到 70 年代中期，该阶段主要是对系统结构的研究，如通过逻辑结构或者图结构描述机器的内部结构。

第三阶段是从 20 世纪 70 年代中期到 80 年代中期，主要通过研究学习策略和学习方法提高改善学习的效率，同时引入知识数据库。此阶段机器学习取得了长足的发展。

第四阶段从 1986 年至今，随着神经网络的引入，以及人工智能的需要，人们对机器学习的机制进行了研究。

#### 2. 机器学习的分类

从研究领域角度分，机器学习主要分为如下四大类：无监督学习、监督学习、半监督学习和增强学习。

（1）无监督学习是一种自学习的分类方式，对没标记的训练样本进行学习，发掘未知

数据间隐藏的结构关系。无监督学习和核密度估计方法非常相似。常用的无监督学习有关联规则学习和聚类学习。

笔记

（2）监督学习是有人工参与的一种学习。监督学习一般分为 3 步：第一步，标记样本；第二步，训练；第三步，模型概率估计。其大概过程如下：首先，输入样本的特征向量和样本类别标记；其次，训练时通过分析样本的特征向量，将预测结果与训练样本的实际标记情况进行比较；最后，调整预测模型，直到预测模型的准确率和预期的准确率相符为止。

（3）半监督学习是同时使用已标记的样本数据和未标记的样本数据实现的一种预测方法。半监督学习分为直推和归纳两种模式。要先用已标记数据训练分类器模型，学习数据的内在结构联系，以便有效地对数据进行预测。

（4）增强学习是通过与环境的测试性交互优化和估计实际动作，实现序列的决策，输入数据同时作为对模型的反馈。与其他类型的学习相比，强化学习输入的数据直接反馈到模型，模型同时做出相应的调整，并根据状态变化获得某种强化信号，最终实现与环境的交互。常用的增强学习算法有 Q-Learning、时间差学习算法等。

## 子任务 1.1.2 机器学习与深度学习

深度学习（Deep Learning，DL）属于机器学习的子类。它的灵感来源于人类大脑的工作方式，是利用深度神经网络解决特征表达的一种学习过程。深度神经网络本身并非一个全新的概念，可理解为包含多个隐含层的神经网络结构。为了提高深度神经网络的训练效果，人们在神经元的连接方法以及激活函数等方面进行了调整。其目的在于建立、模拟人脑进行分析学习的神经网络，模仿人脑的机制解释数据，如文本、图像、声音等。

图 1-1　机器学习和深度学习的关系

在人工智能高速发展的今天，深度学习就是一种特殊的机器学习。机器学习和深度学习的关系如图 1-1 所示。

# 任务 1.2　机器学习的三大要素

机器学习有三大要素：模型、策略、算法。机器学习研究的是从数据中通过选取合适的算法，自动地归纳逻辑或规则，并根据归纳的结果（模型）与新数据进行预测。模型是机器学习的目的。如何构造模型需要策略。实现这个模型需要算法。

## 子任务 1.2.1 模型

模型就是用来描述客观世界的数学模型。模型是从数据里抽象出来的。进行数据分析时，通常只有数据，然后观察数据找规律，找到的规律就是模型。与猜数字游戏类似：1，4，16，…，（ ），…，256，括号里是什么？只有把这串数抽象成模型，才能知道括号里是什么。其实，我们小时候就接触到机器学习，只是那时忙于考试，没有深入思考。

再举一个例子，购买产品的顾客到达服务台的时间是什么模型？也许是一个泊松分布。文本中某个词项出现的概率是什么模型？也许是隐含狄利克雷分布。股票的价格随时

笔记

间的变化有何变化？是基于布朗运动的二项随机分布……

模型可以是确定性的，也可以是随机的，总之用数学可以描述，只要数学可以描述的，就可以进行预测分析。所以，我们的根本目的是找一个模型，描述我们已经观测到的数据。

## 子任务 1.2.2 策略

例如，想用一个正态分布描述一组数据，就要构造这个正态分布，实际上就是预测这个分布的参数，如均值或方差。但是，需要有一系列标准选择合适的模型，模型不是拍脑袋来的。想用正态分布，理由是什么？想用二项分布，凭什么不能用三角分布？想让正态分布的均值为 0.5，为什么选 0.5 比选 0.2 好？做研究一旦任性，别人会质疑你，所以就需要有一系列标准证明一个模型比另一个模型好，这就是策略。

不同的策略对应不同模型的比较标准和选择标准。就跟选班干部一样，选帅的，那就让吴彦祖当班长；选会唱歌的，没准是周杰伦……所以，最终确定的模型实际上与两方面有关：第一，拿到的数据；第二，选择模型的策略。

说到策略，一般以经验风险最小化作为常用的标准。经验风险最小，是指将这个模型套到已有的观测数据上，基本是靠谱的，但在已有观测数据不足的情况下，也可以用结构风险最小化作为标准。这也是大多数时候我们在机器学习时有意或无意用到的准则。经验风险最小化和结构风险最小化是一个参数优化的过程，需要构造一个损失函数描述经验风险，损失函数可以理解为预测错一个数据给我们带来的代价。每个人对损失函数的定义都不同，所以优化出的结果也不同，这也导致最终学习到的模型会各种各样，解决一个问题的方案有多种。

## 子任务 1.2.3 算法

有了数据和学习模型的策略，然后就可以开始构造模型了，如果已有模型的基本形式，那就是一个优化模型参数的问题了。优化对于学习过确定性模型的朋友并不陌生，但是优化过程往往是复杂的，面对复杂的数学优化问题，通常很难通过简单的求导获得最终的结果，所以就要构造一系列算法。

我们的目标是高效的算法、更小的计算机内存代价、更快的运算速度、更有效的参数优化结果。

# 任务 1.3 机器学习的开发思路

作为数据分析师，经常会做一些报表、图表，计算平均值，偶尔用 SQL。有一天你突然意识到可以对你的数据做更多的事，那么恭喜你，你可以开始做机器学习了，但是当打开搜索引擎，发现有 4360 万个关于“机器学习”的结果，有的技术太深奥，有的只有大段的数学公式或者深不可测的人工智能产品，但是这些对一个新手来说都不够友好。因此，我有了写下面几个建议的想法，帮助梳理机器学习的开发思路。

第一步：找到工作中的哪一个流程是可以通过机器学习解决的。

首先要做的是思考自己的工作，厘清工作的内容，思考哪项工作任务可以通过机器学

习得到更好的结果或者提高工作效率。下面是几个机器学习可以解决的问题。

笔记

（1）预测：例如，哪个客户会流失，哪个客户会归还贷款，谁将变成最重要的客户，谁在说谎，谁会死去（泰坦尼克号项目），当有相关数据的时候，就可以预测这些事情。一般情况下，预测的结果不会针对个人，也不能做到 100% 正确，但是可以帮你从看上去随机的情况中找到可能性更大的情况。

（2）划分：不同的顾客，他们的行为模式是不同的，通过机器学习，可以找出不同顾客之间有意义的不同点，将顾客划分为不同的组，这种做法有助于了解客户，甚至可以想象出他们的样子。另外，细分也有助于销售、生产、客户管理等部门之间的合作。

（3）分类：当做完细分操作后，就可以快速理解你的新客属于哪个类别，这可以通过分类任务完成。对客户进行划分，知道他们的轮廓，拥有一些关于新客户的新数据，就可以很容易地了解新客户了。

（4）联系：它是关于你的新客户可能购买什么样的产品，这样你就可以非常聪明地制订销售策略。

（5）模型探索：你可能对你的公司发展影响最大的某个因素感兴趣，销售、士气、员工忠诚度等，我们称之为模型探索，即发现非明显的因素。例如，我们发现，在电信业上抱怨频率比较高的用户流失的概率更小。

第二步：为你的问题找到最合适的算法。

描述机器算法并不是一件简单的事。尤其对初学者来说，在找到正确的书或者训练之前可能走很多弯路。

（1）决策树：将数据根据不同决策分组。例如，将目标客户分为谁会买，谁不会买。

（2）朴素贝叶斯：朴素的意思是它不会考虑列属性之间的依赖关系。它的作用和决策树类似。

（3）K-means 或者期望最小值：找出数据中隐藏的分类。

（4）逻辑回归：预测一些事情发生的可能性。

（5）神经网络：非常适合用来进行模型探索，尤其是那些复杂且不明显的数据。

（6）线性回归：由于它特别简单，因此使用的频率也最高。可以使用它分析数据之间的依赖性，或者进行一些预测。

（7）关联规则：用来分析有助于销售的产品组合。

以上就是出现频率最高的几个算法，除此之外，还有如支持向量机等算法。值得注意的是，没有最好的算法，只有最合适的算法。

第三步：选择合适的工具。

可选择专业的数据分析软件，如 MATLAB 等，目前已经有很多开源免费的工具。对一个机器学习的新手，Python 是一种非常不错的入门语言，不仅简单易学，而且在各个行业的应用也非常广，作为第一种入门语言非常合适。

第二步中提到的机器学习算法，目前已经有开源的库实现，如 Python 中的 Scikit-Learn 或 TensorFlow，性能都经过了优化。

第四步：如何使用工具。

取得数据之后首先是观察数据，如 Kaggle 高手写的新手教程都是先进行数据展示，他们会先将数据用图的方式展现出来，无论是使用 Excel，还是使用 Python，图可以告诉

笔记

你一组数据最有价值的部分。然后通过观察，排除无效的数据，筛选合适的特征，选取合适的模型，最终整理出的数据才是真正程序需要处理的。

对于新手来说，复杂的数学公式、庞大的数据，或者是高大上的 AlphaGo，只会让人望而却步，无从下手。新手应尝试从自己工作或者生活中的小问题入手，厘清思路、分析数据、选取模型、运用机器学习算法，才能更深入地学习。

# 项目 2 Python 语言在机器学习中的应用

## 知识目标

1. 了解 Python 语言的概况和优缺点。
2. 了解 Python 语言的发展过程。
3. 熟悉 Python 语言在机器学习中的应用。

## 能力目标

1. 学会使用 Python 语言常用的库。
2. 具有阅读 Python 程序的能力。
3. 具有机器学习所需要的 Python 语言编程的基本能力。

## 知识导图

笔记

## 项目导入

人工智能和机器学习逐渐应用于各个渠道、行业，大公司在这些领域进行投资，对机器学习和人工智能领域专家的需求也相应增长。IBM 机器学习部门的 Jean Francois Puget 曾表示：对于人工智能和机器学习，Python 是最受欢迎的语言，而且这一结论是基于 indeed.com 的趋势搜索结果得出的。

另一方面，为什么选择 Python 作为实现机器学习算法的编程语言呢？其原因可以归纳为以下三个：① Python 的语法清晰；②易于操作纯文本文件；③使用广泛，存在大量的开发文档。

## 任务 2.1 Python 概述

Python，英 ['paɪθən]，发音接近“派森”，是一门高级语言，也是目前最受欢迎的编程语言之一。人工智能的兴起让 Python 家喻户晓。Python 是由荷兰计算机程序员吉多·范·罗苏姆（Guido van Rossum，人们都叫他龟叔）在 1989 年圣诞节为了打发无聊时间开始用 C 语言编写的一门编程语言。

图 2-1 是 TIOBE 在 2020 年 4 月公布的编程语言热度排行榜，从排行榜中可以看出，最受欢迎的编程语言依次是 Java、C、Python、C++。没错，Python 已经上升到第三位。

| Apr 2020 | Apr 2019 | Change | Programming Language | Ratings | Change |
|---|---|---|---|---|---|
| 1 | 1 | | Java | 16.73% | +1.69% |
| 2 | 2 | | C | 16.72% | +2.64% |
| 3 | 4 | ^ | Python | 9.31% | +1.15% |
| 4 | 3 | v | C++ | 6.78% | -2.06% |
| 5 | 6 | ^ | C# | 4.74% | +1.23% |
| 6 | 5 | v | Visual Basic | 4.72% | -1.07% |
| 7 | 7 | | JavaScript | 2.38% | -0.12% |
| 8 | 9 | ^ | PHP | 2.37% | +0.13% |
| 9 | 8 | v | SQL | 2.17% | -0.10% |
| 10 | 16 | ^^ | R | 1.54% | +0.35% |
| 11 | 19 | ^^ | Swift | 1.52% | +0.54% |
| 12 | 18 | ^^ | Go | 1.36% | +0.35% |
| 13 | 13 | | Ruby | 1.25% | -0.02% |
| 14 | 10 | vv | Assembly language | 1.16% | -0.55% |
| 15 | 22 | ^^ | PL/SQL | 1.05% | +0.26% |
| 16 | 14 | v | Perl | 0.97% | -0.30% |
| 17 | 11 | vv | Objective-C | 0.94% | -0.57% |
| 18 | 12 | vv | MATLAB | 0.93% | -0.36% |
| 19 | 17 | v | Classic Visual Basic | 0.83% | -0.23% |
| 20 | 27 | ^^ | Scratch | 0.77% | +0.28% |

图 2-1 编程语言热度排行榜

**说明**

TIOBE 编程语言热度排行榜是编程语言当前流行趋势的一个重要指标，每月都会更新，这个排行榜基于全球的技术工程师、课程以及第三方供应商的数量，其中包括流行的搜索引擎和技术社区，如百度、Google、CSDN、Hao 123、必应、维基百科等。

不仅如此，Python 的发展势头很猛，用户增长很快，已经成为目前编程语言最大的赢家。

笔记

# 任务 2.2　Python 语言的优缺点

## 子任务 2.2.1 Python 语言的优点

### 1. 简单

Python 被认为是目前最简单易学的语言，很多高校已经将 Python 作为一门必修的程序设计语言。对于初学者来说，Python 程序不仅简单易懂、入门容易，而且编写较复杂的程序也比较容易。

众所周知，C 语言是基础易学的语言。 Python 语言和 C 语言的对比见表 2-1。

表 2-1　Python 语言和 C 语言的对比

| 对比项目 | Python | C |
| --- | --- | --- |
| 难度 | 上手快、见效快 | 偏底层、不易学 |
| 面向对象 / 过程 | 面向对象<br>提高代码重用性 | 面向过程<br>有时需手动实现函数功能 |
| 跨平台 | 可以 | 不能 |
| 头文件 | 无须写头文件 | 需要写头文件 |
| 定义数据类型 | 无须事先定义变量类型 | 需要事先定义变量类型 |
| 语法规则 | 每条语句结尾无分号 | 每条语句的结尾都有分号 |
| 运算符 | 无 ++、-- 运算符 | 有 ++、-- 运算符 |
| 函数 | 无须事先申明函数 | 需要事先申明函数 |

### 2. 开发速度快、效率高

Python 的底层是 C 语言，具有很多标准库和强大的第三方库，基本上可以满足想在计算机上实现的所有功能。从 Python 的官方库下载相应模块直接调用，可以大大缩短开发周期。

### 3. 免费、开源

可以在网上发布或复制 Python 代码，甚至对代码进行改动。

### 4. 跨平台性

由于 Python 是开源的，所以 Python 已经被移植在很多不同的平台上。可以将 Python 程序移植到市面上的所有平台而不受影响，这些平台包括 Windows、Android、Linux、OS/2 等。

### 5. 可扩展性

如果想使程序中的一段关键代码运行得更快，可以用 C 或者 C++ 编写此段程序，然后在 Python 程序中调用它。

### 6. 丰富的库

子任务 2.3 中将讲述 Python 语言的库。

**注意**

我们将 Python 和 C 作对比，以突出 Python 的简单性，但并不意味着学了 Python 就不用学 C，C 语言程序设计仍旧是专业基础课。C 语言是强大的工具，是目前广泛流行的语言之一。Python 是基于 C 语言的，学好 Python 语言再学 C 语言更容易上手。

笔记

## 子任务 2.2.2 Python 语言的缺点

### 1. 速度慢

Python 的运行速度比 C 和 Java 都慢，但是这里的慢是相对的，如运行一个程序是微秒级的差别，对于一般程序员来说，这个时间完全可以忽略不计。绝大多数情况下，Python 可以满足用户对程序运行速度的要求。

### 2. 代码不能加密

发布 Python 程序，相当于发布源代码，与 Python 不同的是，C 语言不用发布源代码，而是发布编译后的文件（类似 .exe 文件），要从机器码反推源代码是不可能的，所以编译性的语言没有这个问题，解释性的语言需要发布源代码，如果项目要求源代码加密，那么一开始就不建议使用 Python。

### 3. 线程不能利用多 CPU

全局解释器锁（Global Interpreter Lock，GIL），因为它，任何时刻都只有一个线程被计算机执行，即使一个具有多 CPU 的平台，也会被禁止并行执行多线程。

## 任务 2.3 Python 语言的库

Python 语言的库非常丰富，主要分为标准库和第三方库两大类。在安装 Python 的时候标准库自动安装在计算机中，使用它们会让你事半功倍。第三方库，需要下载后安装到 Python 相应目录。无论是标准库还是第三方库，它们的调用方式都一样。表 2-2 是 Python 常用的第三方库。

表 2-2　Python 常用的第三方库

| 名称 | 作用 |
| --- | --- |
| Scrapy | 与爬虫相关的库 |
| Requests | 著名的 http 库 |
| Pillow | PIL（Python 图形库）的分支，图形工作领域人的必备 |
| matplotlib | 绘制数据图的库，对数据分析者很有用 |
| OpenCV | 计算机视觉库，在人脸识别方面很有用 |
| pytesseract | 图片文字识别，同 OCR（光学字符识别） |
| wxPython | GUI（图形用户接口）库 |
| Twisted | 网络开发者必备工具，有非常好的 API |
| SymPy | 用于符号数学的库 |
| SQLAlchemy | 用于数据库的库 |
| SciPy | 算法和数据工具的库 |
| Scapy | 数据表探测和分析库 |
| Pywin32 | 与 Windows 交互的库 |
| PyQT | GUI 库 |
| PyGTK | GUI 库 |
| Pyglet | 用于 3D 动画和游戏开发 |

续表

| 名称 | 作用 |
| --- | --- |
| Pygame | 用于开发 2D 游戏 |
| NumPy | 用于高级计算 |
| nose | 测试工作必不可少的工具 |
| nltk | 提供自然语言的库 |
| IPython | 提示信息的工具 |
| BeautifulSoup | 对 HTML 或 XML 进行解析的工具 |

笔记

标准库主要分为 Python 增强、系统互动、网络三类。

（1）Python 增强类：可以增强 Python 自身已有的一些功能，如文字处理、数据对象、日期和时间、数学运算、存储等。

（2）系统互动类：主要指和操作系统以及文件系统的互动。Python 可以实现操作系统的很多功能，如 Python 运行控制、操作系统、线程与进程等。

（3）网络类：主要指基于 Socket 层的网络应用和互联网应用。

# 项目 3 线性回归模型及应用

## 知识目标

1. 了解线性回归的基本概念与分类。
2. 理解常见线性回归模型的基本数学原理。

## 能力目标

1. 掌握基于 Python 语言的最小二乘法实现。
2. 掌握基于 Python 语言的梯度下降法实现。
3. 掌握基于 Python 语言的多项式与广义线性回归实现。
4. 具有解读线性回归模型从原理到算法的能力。

## 知识导图

笔记

## 项目导入

回归（Regression）是监督学习的另一个重要问题，用于预测输入变量（自变量）和输出变量（因变量）之间的关系，特别是当输入变量的值发生变化时，输出变量的值随之发生的变化，回归模型正是表示从输入变量到输出变量之间映射的函数。回归问题的学习等价于函数拟合：选择一条函数曲线，使其很好地拟合已知数据且很好地预测未知数据。在统计学中，线性回归（Linear Regression）是利用称为线性回归方程的最小平方函数对一个或多个自变量和因变量之间关系进行建模的一种回归分析，这种函数是一个或多个被称为回归系数的模型参数的线性组合。只有一个自变量的情况称为简单回归，大于一个自变量情况的称为多元回归。

## 任务 3.1 线性回归模型概述

在线性回归中，数据使用线性预测函数建模，并且未知的模型参数也是通过数据估计，这些模型称为线性模型。最常用的线性回归建模是给定 $x$ 值的 $y$ 的条件均值是 $x$ 的仿射函数。不太一般的情况，线性回归模型可以是一个中位数或一些其他的给定的条件下的条件分布的分位数作为 $x$ 的线性函数表示。像所有形式的回归分析一样，线性回归也把焦点放在给定 $x$ 值的 $y$ 的条件概率分布，而不是 $x$ 和 $y$ 的联合概率分布（多元分析领域）。

线性回归是回归分析中第一种经过严格研究并在实际应用中广泛使用的模型。这是因为线性依赖于其未知参数的模型比非线性依赖于其未知参数的模型更容易拟合，而且产生的估计的统计特性也更容易确定。

线性回归模型经常用最小二乘逼近拟合，但它们也可能用别的方法拟合，例如用最小化“拟合缺陷”在一些其他规范里（如最小绝对误差回归）。相反，最小二乘逼近可以用来拟合非线性的模型，因此，尽管“最小二乘法”和“线性模型”是紧密相连的，但它们是不能画等号的。

### 子任务 3.1.1 线性回归模型的基本概念

线性回归是机器学习中最简单、最重要的模型之一，其模型建立遵循流程：获取数据、数据预处理、训练模型、应用模型。

回归模型可以理解为：存在一个点集，用一条曲线拟合它分布的过程。如果拟合曲线是一条直线，则称为线性回归；如果拟合曲线是一条二次曲线，则被称为二次回归。线性回归是回归模型中最简单的一种。

在线性回归中有几个基本概念需要掌握。

#### 1. 假设函数

假设函数是指用数学的方法描述自变量和因变量之间的关系，它们之间可以是一个线性函数或非线性函数。在线性回归模型中，假设函数 $\hat{Y}=aX+b$ 表示模型的预测结果，$\hat{Y}$ 用来和真实的 $Y$ 区分。模型要学习的参数即 $a$，$b$。

#### 2. 损失函数

损失函数是指用数学的方法衡量假设函数预测结果与真实值之间的误差。这个差距越小，预测越准确，而算法的任务就是使这个差距越来越小。建立模型后，需要给模型一个

优化目标，使得学到的参数能够让预测值 $\hat{Y}$ 尽可能地接近真实值 $Y$。输入任意一个数据样本的目标值 $Y_i$ 和模型给出的预测值 $\hat{Y}_i$，损失函数输出一个非负的实数值，这个实值通常用来反映模型误差的大小。

笔记

对于线性模型，最常用的损失函数是均方误差（Mean Squared Error，MSE）。

$$MSE=\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i-Y_i)^2 \tag{3.1}$$

即对于一个大小为 $n$ 的测试集，$MSE$ 是一个数据预测结果误差平方的均值。

### 3. 优化算法

在模型训练中优化算法也是至关重要的，它决定了一个模型的精度和运算速度。本项目的线性回归实例中主要使用最小二乘法、梯度下降法进行优化。

需要指出的是，梯度下降是深度学习中非常重要的概念。值得庆幸的是，它也十分容易理解。损失函数 $J(w,b)$ 可以理解为变量 $w$ 和 $b$ 的函数。实际上，可能是更高维的向量，但是为了方便说明，这里假设 $w$ 和 $b$ 都是一个实数。算法的最终目标是找到损失函数的最小值，而这个寻找过程就是不断地微调变量 $w$ 和 $b$ 的值，一步一步试出最小值。而试的方法是沿着梯度方向逐步移动，直到取到最小值或逼近最小值。

因为是凸函数，所以无论初始化在曲面上的哪一点，最终都会收敛到同一点或者相近的点。

## 子任务 3.1.2　线性回归的一般模型

线性回归遇到的问题一般是这样的：假设有 $m$ 个样本，每个样本对应 $n$ 维特征和一个结果输出，如

$(x_1^{(0)},x_2^{(0)},\cdots,x_n^{(0)},y_0)$，$(x_1^{(1)},x_2^{(1)},\cdots,x_n^{(1)},y_1)$，$\cdots$，$(x_1^{(m)},x_2^{(m)},\cdots,x_n^{(m)},y_m)$

现在的问题是，对于一个新的观测 $(x_1^{(x)},x_2^{(x)},\cdots,x_n^{(x)})$，它对应的 $y_x$ 是多少呢？ 如果这个问题里的 $y$ 是连续的，则是一个回归问题，否则是一个分类问题。

对于 $n$ 维特征的样本数据，如果决定使用线性回归，那么对应的模型是这样的：

$$\boldsymbol{h}_\theta(x_1,x_2,\cdots,x_n)=\theta_0+\theta_1x_1+\theta_2x_2+\cdots+\theta_nx_n \tag{3.2}$$

式中：$\theta_i(i=0,1,2,\cdots,n)$ 为模型参数；$x_i(i=0,1,2,\cdots,n)$ 为每个样本的第 $i$ 个特征值。

模型式（3.2）的表示可以简化。增加一个特征 $x_0=1$，式（3.2）可以表示为：

$$\begin{aligned}\boldsymbol{h}_\theta(x_0,x_1,x_2,\cdots,x_n)&=\theta_0x_0+\theta_1x_1+\theta_2x_2+\cdots+\theta_nx_n\\&=\sum_{i=0}^{n}\theta_ix_i\end{aligned} \tag{3.3}$$

进一步，用矩阵形式表达更加简洁，具体如下：

$$\boldsymbol{h}_\theta(\boldsymbol{X})=\boldsymbol{X}\theta \tag{3.4}$$

式中：假设函数 $\boldsymbol{h}_\theta(\boldsymbol{X})$ 为 $m\times1$ 的向量；$\boldsymbol{\theta}$ 为 $(n+1)\times1$ 的向量；$\boldsymbol{X}$ 为 $m\times(n+1)$ 维矩阵。$m$ 代表样本的个数，$n$ 代表样本的特征数。

由上得到了模型，现在要求需要的损失函数，一般线性回归可用均方误差作为损失函

笔记

数。损失函数的代数法表示如下：

$$\boldsymbol{J}(\theta_0,\theta_1,\cdots,\theta_n)=\sum_{i=1}^{m}[\boldsymbol{h}_\theta(x_0^{(i)},x_1^{(i)},\cdots,x_n^{(i)})-y_i]^2 \tag{3.5}$$

进一步，用矩阵形式表达损失函数：

$$\boldsymbol{J}(\boldsymbol{\theta})=\frac{1}{2}(\boldsymbol{X\theta}-\boldsymbol{Y})'(\boldsymbol{X\theta}-\boldsymbol{Y}) \tag{3.6}$$

由于矩阵法表达比较简洁，后面将统一采用矩阵方式表达模型函数和损失函数。

线性模型形式简单、易于建模，但却蕴涵着机器学习中一些重要的基本思想。许多功能更强大的非线性模型可在线性模型的基础上通过引入层级结构或高维映射而得。此外，由于 $\boldsymbol{\theta}$ 直观表达了各属性在预测中的重要性，因此线性模型有很好的可解释性。

## 任务 3.2 最小二乘法

对于线性回归的损失函数式（3.5）或式（3.6），常用两种方法求损失函数最小化时的参数 $\boldsymbol{\theta}$：一种是最小二乘法；一种是梯度下降法。考虑到便于编程实现，下面以模型的矩阵表示为基础介绍这两种基本算法。

### 子任务 3.2.1 最小二乘法的原理与要解决的问题

最小二乘是一种优化思想（“平方”在古时候的称谓为“二乘”），最小二乘法是最小化平方和的优化方法；这里的平方和指的是误差（真实目标对象与拟合目标对象的差）的平方。“最小二乘法”的核心是保证所有数据偏差的平方和最小，使得拟合对象最大限度逼近目标对象。

最小二乘法是勒让德于 19 世纪发现的，原理的一般形式很简单，当然发现的过程是非常艰难的，形式如下：

$$\text{目标函数}=\sum(\text{观测值}-\text{理论值})^2 \tag{3.7}$$

观测值就是多组样本，理论值就是假设拟合函数。目标函数也就是在机器学习中常说的损失函数。最小二乘法的目标是得到使目标函数最小化时的拟合函数的模型。下面举一个最简单的线性回归的例子。例如，有 $m$ 个只有一个特征的样本：

$$(x^{(1)},y^{(1)}),\ (x^{(2)},y^{(2)}),\ \cdots,\ (x^{(m)},y^{(m)})$$

样本采用下面的拟合函数：

$$\boldsymbol{h}_{\boldsymbol{\theta}}(x)=\theta_0+\theta_1 x \tag{3.8}$$

这样的样本有一个特征，对应的拟合函数有两个参数 $\theta_0$ 和 $\theta_1$ 需要求出。

目标函数为：

$$\begin{aligned}J(\theta_0,\theta_1)&=\sum_{i=1}^{m}[y^{(i)}-h_\theta(x^{(i)})]^2\\&=\sum_{i=1}^{m}[y^{(i)}-\theta_0-\theta_1x^{(i)}]^2\end{aligned} \tag{3.9}$$

用最小二乘法使 $J(\theta_0,\theta_1)$ 最小，求出使 $J(\theta_0,\theta_1)$ 最小的 $\theta_0$ 和 $\theta_1$，这样拟合函数就得出了。

笔记

那么，最小二乘法怎么才能使 $J(\theta_0,\theta_1)$ 最小呢？下面以上文所述线性回归的一般模型的矩阵表示为基础，介绍最小二乘法的矩阵法解法。

## 子任务 3.2.2 最小二乘法的矩阵法解法

矩阵法比代数法简洁，且矩阵运算可以取代循环，所以现在很多书和机器学习库都是用矩阵法做最小二乘法。

模型假设函数如式（3.4）、损失函数如式（3.6）所示。其中 $\boldsymbol{Y}$ 是样本的输出向量，维度为 $m\times1$。常数 $\frac{1}{2}$ 在这里主要是为了求导后系数为 1，方便计算。

根据最小二乘法的原理，要对这个损失函数对 $\boldsymbol{\theta}$ 向量求导取 0，详细的求解过程用到矩阵求导链式法则的相关原理，可查阅相关资料（如张贤达的《矩阵分析与应用》一书），不在此赘述。其结果为

$$\frac{\partial \boldsymbol{J}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}=\boldsymbol{X}'(\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{Y})=0 \tag{3.10}$$

整理上述求导等式后可得：

$$\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\theta}=\boldsymbol{X}'\boldsymbol{Y} \tag{3.11}$$

两边同时左乘 $(\boldsymbol{X}'\boldsymbol{X})^{-1}$ 可得：

$$\boldsymbol{\theta}=(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y} \tag{3.12}$$

这样就求出了 $\boldsymbol{\theta}$ 向量表达式的公式，免去了代数法一个个求导的麻烦。只要给出数据，就可以用式（3.12）算出 $\boldsymbol{\theta}$ 向量。

## 子任务 3.2.3 最小二乘法的局限性和适用场景

从上面可以看出，使用最小二乘法简洁高效，比使用梯度下降迭代法方便很多。但是，这里要谈一下最小二乘法的局限性。

（1）最小二乘法需要计算 $\boldsymbol{X'X}$ 的逆矩阵，有可能它的逆矩阵不存在，这样就没办法直接用最小二乘法了，此时梯度下降法仍然可以使用。当然，也可以通过对样本数据进行整理，去掉冗余特征。让 $\boldsymbol{X'X}$ 的行列式不为 0，然后继续使用最小二乘法。

（2）当样本特征 $n$ 非常大的时候，计算 $\boldsymbol{X'X}$ 的逆矩阵是一个非常耗时的工作，甚至不可行，此时以梯度下降为代表的迭代法仍然可以使用，那这个 $n$ 到底多大就不适合最小二乘法呢？如果没有很多分布式大数据计算资源，建议超过 10000 个特征就用迭代法；或者通过主成分分析降低特征的维度后再用最小二乘法。

（3）如果拟合函数不是线性的，这时无法使用最小二乘法，通过一些技巧转化为线性才能使用，此时梯度下降法仍然可以用。

（4）最后讲一些特殊情况。当样本量 $n$ 很少，小于特征数 $n$ 的时候，这时拟合方程是欠定的，常用的优化方法都无法拟合数据。当样本量 $m$ 等于特征数 $n$ 的时候，用方程组求解就可以了。当 $m$ 大于 $n$ 时，拟合方程是超定的，也就是常用于最小二乘法的场景了。

笔记

# 任务 3.3 梯度下降法

在求解机器学习算法的模型参数，即无约束优化问题时，梯度下降（Gradient Descent）是最常采用的方法之一。在机器学习算法中，在最小化损失函数时，可以通过梯度下降法一步步地迭代求解，得到最小化的损失函数和模型参数值。反过来，如果需要求解损失函数的最大值，这时就需要用梯度上升法迭代了。

梯度下降法和梯度上升法是可以互相转化的。例如，求解损失函数$J(\theta)$的最小值，需要用梯度下降法迭代求解。实际上，也可以反过来求解损失函数$J(\theta)$的最大值，这时梯度上升法就派上用场了。下面详细介绍梯度下降法。

## 子任务 3.3.1 梯度下降的直观解释

首先看梯度下降的一个直观的解释。例如，在一座大山上的某处位置，由于不知道怎么下山，于是决定走一步算一步，也就是在每走到一个位置的时候，求解当前位置的梯度，沿着梯度的负方向，也就是当前最陡峭的位置向下走一步，然后继续求解当前位置梯度，向这一步所在位置沿着最陡峭、最易下山的位置走一步。这样一步步地走下去，一直走到觉得已经到了山脚。当然，这样走下去有可能不能走到山脚，而是到了某个局部的山峰低处。

从上面的解释可以看出，梯度下降不一定能够找到全局的最优解，有可能是一个局部最优解。当然，如果损失函数是凸函数，梯度下降法得到的解就一定是全局最优解。关于梯度下降法的直观理解也可从图 3-1 分析。

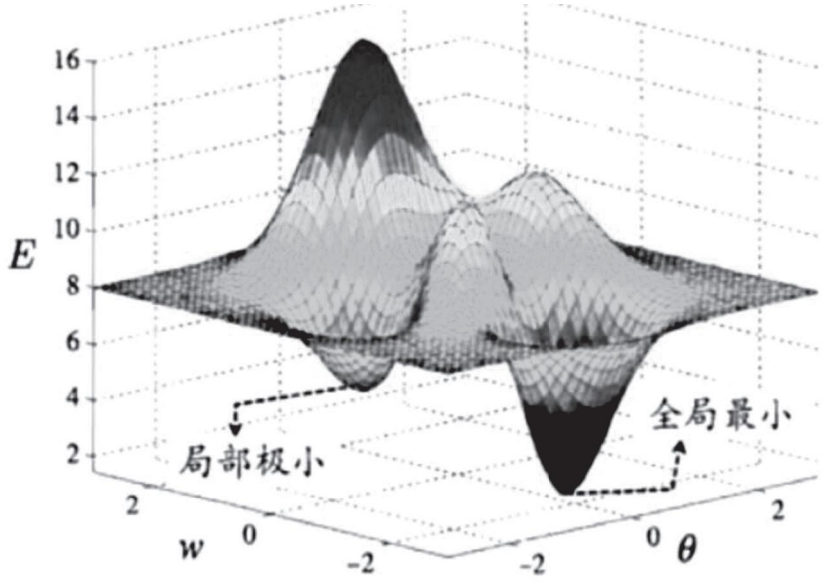



图 3-1　梯度下降法示意图

## 子任务 3.3.2 梯度下降的相关概念

在详细了解梯度下降的算法之前，先理解如下相关的一些概念。

（1）步长（Learning Rate）：决定了在梯度下降迭代的过程中每一步沿梯度负方向前进

的长度。用上面下山的例子，步长就是在当前这一步所在位置沿着最陡峭、最易下山的位置走的那一步的长度。

笔记

（2）特征（Feature）：指的是样本中的输入部分，如 2 个单特征的样本 $(x^{(0)},y^{(0)})$、$(x^{(1)},y^{(1)})$，则第一个样本特征为 $x^{(0)}$，第一个样本输出为 $y^{(0)}$。

（3）假设函数（Hypothesis Function）：在监督学习中，为了拟合输入样本而使用的假设函数，记为 $h_\theta(x)$。例如，对于单个特征的 $m$ 个样本 $(x^{(i)},y^{(i)})$，可以采用式（3.8）所示的拟合函数。

（4）损失函数（Loss Function）：为了评估模型拟合的好坏，通常用损失函数度量拟合的程度。损失函数极小化，意味着拟合程度最好，对应的模型参数即最优参数。在线性回归中，损失函数通常为样本输出和假设函数的差取平方。例如，对于 $m$ 个样本 $(i=1,2,\cdots,m)$，采用线性回归，可采用式（3.9）所示的损失函数。

在了解这些基本概念的基础上，下面以上文所述线性回归的一般模型的矩阵表示为基础介绍梯度下降法的矩阵法解法。

## 子任务 3.3.3　梯度下降法的矩阵方式描述

（1）先决条件：模型假设函数如式（3.4）、损失函数如式（3.6）所示。其中 $\boldsymbol{Y}$ 是样本的输出向量，维度为 $m\times1$。

（2）算法相关参数初始化：$\boldsymbol{\theta}$ 向量、算法终止距离 、步长 ，在没有任何先验知识的时候，可以将所有的 $\boldsymbol{\theta}$ 初始化为 0，将步长初始化为 1。在调优的时候再优化。

（3）算法过程：

① 确定当前位置的损失函数的梯度，对于 向量，其梯度表达式如下：

$$\frac{\partial}{\partial\boldsymbol{\theta}}\boldsymbol{J}(\boldsymbol{\theta})=\boldsymbol{X}'(\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{Y}) \tag{3.13}$$

②用步长乘以损失函数的梯度，得到当前位置下降的距离，即 $\alpha\frac{\partial}{\partial\theta_i}\boldsymbol{J}(\theta_0,\theta_1,\cdots,\theta_n)$ 对应前面登山例子中的某一步。

③ 确定 $\boldsymbol{\theta}$ 向量里的每个值，梯度下降的距离都小于 $\varepsilon$，如果小于 $\varepsilon$，则算法终止，当前 向量即最终结果，否则进入步骤④。

④ 更新 $\boldsymbol{\theta}$ 向量，其更新表达式如式（3.14）所示。更新完毕后继续转入步骤①。

$$\boldsymbol{\theta}=\boldsymbol{\theta}-\alpha\frac{\partial}{\partial\boldsymbol{\theta}}J(\boldsymbol{\theta})=\boldsymbol{\theta}-\alpha\boldsymbol{X}'(\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{Y}) \tag{3.14}$$

## 子任务 3.3.4　梯度下降的算法调优

使用梯度下降法时，需要进行调优。哪些地方需要调优呢？

（1）算法的步长选择。在前面的算法描述中提到取步长为 1，但实际上取值取决于数据样本，可以多取一些值。从大到小，分别运行算法，看看迭代效果。如果损失函数在变小，说明取值有效；否则要增大步长。前面说了，步长太大会导致迭代过快，甚至有可能错过最优解；步长太小，迭代速度太慢，很长时间算法都不能结束。所以，算法的步长需

笔记

要多次运行后，才能得到一个较优的值。

（2）算法参数的初始值选择。初始值不同，获得的最小值也有可能不同，因此梯度下降求得的只是局部最小值；当然，如果损失函数是凸函数，则一定是最优解。由于有局部最优解的风险，因此需要用不同的初始值运行算法，关键是求取损失函数的最小值，选择损失函数最小化的初值。

（3）归一化。由于样本不同特征的取值范围不一样可能导致迭代很慢，为了减少特征取值的影响，可以对特征数据归一化，也就是对于每个特征 $x$，求出它的期望 $x$ 和标准差 $\mathrm{std}(x)$，然后转化为：

$$\frac{x-\bar{x}}{\mathrm{std}(x)} \tag{3.15}$$

这样，特征的新期望为 0，新方差为 1，迭代速度大大加快。

## 子任务 3.3.5 梯度下降法和其他无约束优化算法的比较

在机器学习中的无约束优化算法，除了梯度下降以外，还有前面提到的最小二乘法，此外还有牛顿法和拟牛顿法。

梯度下降法和最小二乘法相比，梯度下降法需要选择步长，而最小二乘法不需要。梯度下降法是迭代求解，最小二乘法是计算解析解。如果样本量不算很大，且存在解析解，最小二乘法比梯度下降法有优势，计算速度很快。但是，如果样本量很大，用最小二乘法由于需要求一个超级大的逆矩阵，这时就很难或者很慢才能求解解析解了，使用迭代的梯度下降法比较有优势。

梯度下降法和牛顿法 / 拟牛顿法相比，两者都是迭代求解，不过梯度下降法是梯度求解，而牛顿法 / 拟牛顿法是用二阶的海森矩阵的逆矩阵或伪逆矩阵求解。相对而言，使用牛顿法 / 拟牛顿法收敛更快，但是每次迭代的时间比梯度下降法长。

# 任务 3.4 多项式回归模型

回到开始的式（3.2）所示的线性模型，如果这里不仅仅是 $x$ 的一次方，比如增加为二次方，那么模型就变成了多项式回归。这里写一个只有两个特征的 $p$ 次方多项式回归的模型：

$$\boldsymbol{h}_{\theta}(x_1,x_2)=\theta_0+\theta_1x_1+\theta_2x_2+\theta_3x_1^2+\theta_4x_2^2+\theta_5x_1x_2 \tag{3.16}$$

令 $x_0=1$、$x_1=x_1$、$x_2=x_2$、$x_3=x_1^2$、$x_4=x_2^2$、$x_5=x_1x_2$，则式（3.16）可表示为

$$\boldsymbol{h}_{\theta}(x_1,x_2)=\theta_0+\theta_1x_1+\theta_2x_2+\theta_3x_3+\theta_4x_4+\theta_5x_5 \tag{3.17}$$

可以发现，式（3.17）又重新回到线性回归。这是一个五元线性回归，可以用线性回归的方法完成算法。对于每个二元样本特征 $(x_1,x_2)$，可以得到一个五元样本特征 $(x_1,x_2,x_1^2,x_2^2,x_1x_2)$，通过这个改进的五元样本特征，就重新把不是线性回归的函数变回线性回归。

笔记

## 任务 3.5　广义线性回归

在多项式回归模型中，我们对样本特征端做了推广，这里对输出 $\boldsymbol{Y}$ 做推广。例如，有输出 $\boldsymbol{Y}$ 不满足和 $\boldsymbol{X}$ 的线性关系，但是 $\ln\boldsymbol{Y}$ 和 $\boldsymbol{X}$ 满足线性关系，模型函数如下：

$$\ln \boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\theta} \tag{3.18}$$

这样，对于每个样本的输入 $\boldsymbol{X}$，我们用 $\ln\boldsymbol{Y}$ 去对应，从而仍然可以用线性回归的算法处理这个问题。把 $\ln\boldsymbol{Y}$ 一般化，假设这个函数是单调可微函数 $g(\cdot)$，则一般化的广义线性回归形式是：

$$g(\boldsymbol{Y}) = \boldsymbol{X}\boldsymbol{\theta} \tag{3.19}$$

式中：$g(\cdot)$ 通常称为联系函数。

## 任务 3.6　线性回归应用示例

### 子任务 3.6.1 数据准备

$$\min Q(\hat{a},\hat{b}) = \sum_{i=1}^{n}[y_i - E(y_i)]^2 = \sum_{i=1}^{n}(y_i - \hat{a}x_i - \hat{b})^2 \tag{3.20}$$

为使上述离差平方和达到极小，对 $a$、$b$ 求偏导数，并令其等于 0 求取其估计值。

随着经济水平的提高，越来越多的人开始重视自己的身体，如肥胖等健康问题。在医学上，肥胖一般用体重指数反映，肥胖的人更容易得高血压。下面利用线性回归预测肥胖和高血压的具体关系，并给出如何预测自己是否具有高血压，其中血压预测数据如表 3-1 所示。

表 3-1　血压预测数据

| ID | 体重指数 | 血压 /mmHg | ID | 体重指数 | 血压 /mmHg |
|---|---|---|---|---|---|
| 1 | 20.9 | 123 | 8 | 21.4 | 126 |
| 2 | 21.5 | 123 | 9 | 21.4 | 124 |
| 3 | 19.6 | 123 | 10 | 25.3 | 129 |
| 4 | 26 | 130 | 11 | 22.4 | 124 |
| 5 | 16.6 | 119 | 12 | 26.1 | 133 |
| 6 | 25.9 | 131 | 13 | 23 | 129 |
| 7 | 21.6 | 127 | 14 | 16 | 118 |

利用上面的线性回归求解过程，列出目标函数（误差平方和），然后分别求偏导等于 0，从而求得：

$$a=1.32 \qquad b=96.8$$

在利用回归方法进行预测时需要进行模型显著性检验，计算得模型的平均残差和判定系数为：

残差为 1.17，判定系数为 0.90。

笔记

平均残差远小于血压的平均值 125.6，而且判定系数接近 1。因此，可以说该回归方程是显著的，可以很好地拟合数据集，并可以在此基础上预测未知数据：

$$y=1.32 \times 24 + 96.8 = 128.48$$

## 子任务 3.6.2 代码实现

```
rom numpy import *
import sys
import os
import matplotlib.pyplot as plt
def loadDataSet(fileName):
    X = []; Y = []
    fr = open(fileName)
    for line in fr.readlines():
        curLine = line.strip().split(' ')
        X.append(float(curLine[0])); Y.append(float(curLine[-1]))
    return X,Y
# 绘制图形
def plotscatter(Xmat,Ymat,a,b,plt):
    fig = plt.figure()
    ax = fig.add_subplot(111) # 绘制图形位置
    ax.scatter(Xmat,Ymat,c='blue',marker='o')  # 绘制散点图
    Xmat.sort() # 对 Xmat 各元素进行排序
    yhat = [a*float(xi)+b for xi in Xmat] # 计算预测值
    plt.plot(Xmat,yhat,'r') # 绘制回归线
    plt.show()
    return yhat
# 数据矩阵，分类标签
xArr,yArr = loadDataSet("regdataset.txt")
# 生成 X 坐标列
m = len(xArr)
Xmat = mat(ones((m,2)))
for i in xrange(m): Xmat[i,1] = xArr[i]
Ymat = mat(yArr).T # 转换为 y 列
xTx = Xmat.T*Xmat  # 矩阵左乘自身的转置
ws = []
if linalg.det(xTx) != 0.0:
    # 计算直线的斜率和截距
    # 矩阵正规方程组公式 :inv(X.T*X)*X.T*Y
```

```
        ws = xTx.I * (Xmat.T*Ymat)
    else:
        print "This matrix is singular, cannot do inverse"
        sys.exit(0)  # 退出程序
    print "ws:",ws
    yHat = plotscatter(Xmat[:,1],Ymat,ws[1,0],ws[0,0],plt)
    # 计算相关系数：
    print corrcoef(yHat,Ymat.T)
```

笔记

程序运行结果如图 3-2 所示。

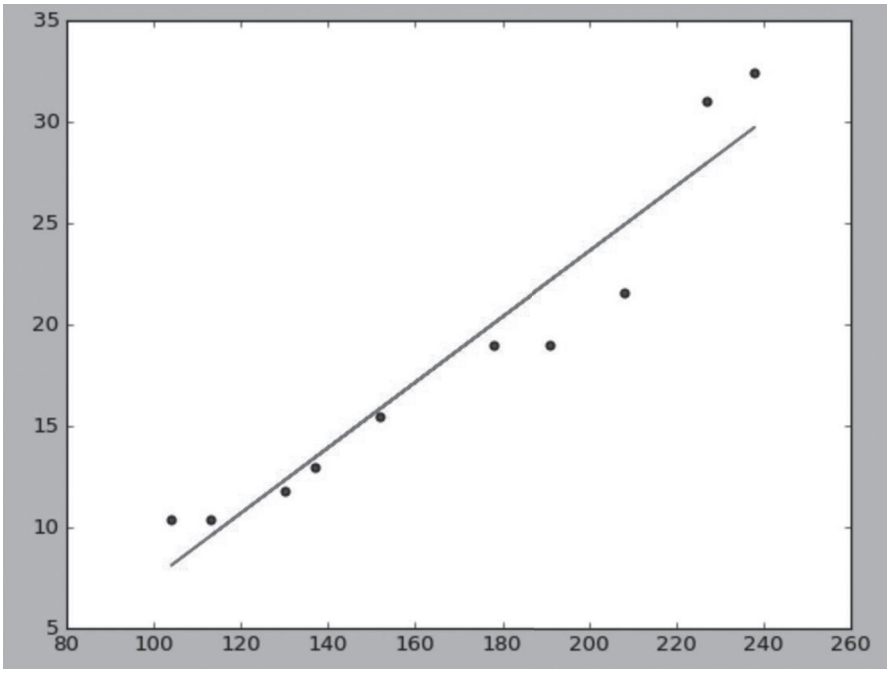



图 3-2　程序运行结果

# 项目 4 逻辑回归模型及应用

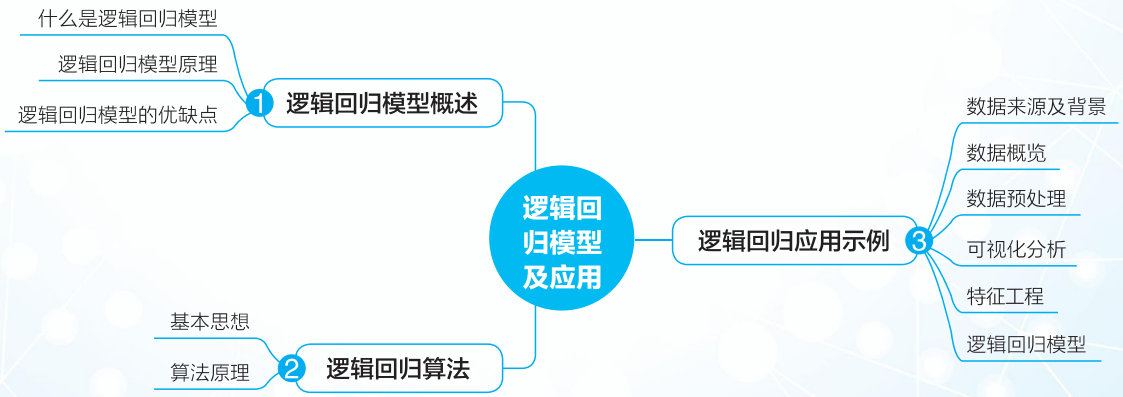
## 知识目标

1. 了解逻辑回归模型的基本概念与原理。
2. 理解逻辑回归算法的基本思想与算法流程。

## 能力目标

1. 掌握基于 Python 语言的逻辑回归算法实现。
2. 具有应用逻辑回归模型解决实际问题的能力。

## 知识导图



笔记

## 项目导入

逻辑回归，虽然这个算法从名字上看是回归算法，但其实际上是一个分类算法。逻辑斯蒂回归模型最早应用于种群生态学，但由于其优秀的性质，对于我们的分类问题，即已知许多特征 $x$，希望通过这些特征预测出 label，就是类别的标签，对于二分类问题，标签只有两个，这里记作 0 和 1（有的记作 +1 和 –1）。对于 $x$，我们希望用一个模型，最终综合起所有的特征 $x$，然后得到一个待判决的数值。要想判断具体属于哪一类，需要设定一个阈值，若大于这个阈值，则给一个类别；若小于这个阈值，则给另一个类别，就像最基本的高维空间用超平面分割两类一样。

# 任务 4.1 逻辑回归模型概述

逻辑（logistic）回归又称 logistic 回归分析，是一种广义的线性回归分析模型，常用于数据挖掘、疾病自动诊断、经济预测等领域。也就是说，逻辑回归是从线性回归模型推广而来的。

## 子任务 4.1.1 什么是逻辑回归模型

逻辑回归（Logistic Regression，LR）模型其实仅在线性回归的基础上套用了一个逻辑函数，就是由于这个逻辑函数，使得逻辑回归模型成为机器学习领域一颗耀眼的明星，更是计算广告学的核心。

“逻辑回归”也称作“评定模型”“分类评定模型”，是离散选择法模型之一。逻辑回归模型是最早的离散选择模型，也是目前应用最广的模型。此后，逻辑回归模型在心理学、社会学、经济学及交通领域得到广泛的应用，并衍生发展出其他离散选择模型，形成完整的离散选择模型体系，如 Probit 模型、NL（Nest Logit）模型、Mixed Logit 模型等。

## 子任务 4.1.2 逻辑回归模型原理

逻辑回归和线性回归的原理相似，可以简单描述为以下过程：

（1）找一个合适的预测函数，一般表示为 $h$ 函数，该函数就是我们需要找的分类函数，用来预测输入数据的判断结果。这个过程非常关键，需要对数据有一定的了解或分析，知道或者猜测预测函数的“大概”形式，如是线性函数，还是非线性函数。

（2）构造一个 Cost 函数（损失函数），该函数表示预测的输出（$h$）与训练数据类别（$y$）之间的偏差，可以是两者之间的差（$h-y$）或者是其他形式。综合考虑所有训练数据的“损失”，将 Cost 求和或者求平均，记为 $J(\theta)$ 函数，表示所有训练数据预测值与实际类别的偏差。

（3）显然，$J(\theta)$ 函数的值越小，表示预测函数越准确（即 $h$ 函数越准确），所以这一步需要做的是找到 $J(\theta)$ 函数的最小值。找函数的最小值有不同的方法，Logistic Regression 实现时用的是梯度下降法（Gradient Descent）。

## 子任务 4.1.3 逻辑回归模型的优缺点

笔记

1. 优点

（1）逻辑回归模型的预测结果是界于 0 和 1 之间的概率。

（2）逻辑回归模型可以适用于连续性和类别性自变量。

（2）逻辑回归模型容易使用和解释。

2. 缺点

（1）对模型中自变量多重共线性较为敏感，例如两个高度相关自变量同时放入模型，可能导致较弱的一个自变量回归符号不符合预期，符号被扭转。需要利用因子分析或者变量聚类分析等手段来选择代表性的自变量，以减少候选变量之间的相关性。

（2）预测结果呈“S”型，因此从 log(odds) 向概率转化的过程是非线性的。在两端随着 log(odds) 值的变化，概率变化很小，边际值太小，slope 太小，而中间概率的变化很大，很敏感。导致很多区间的变量变化对目标概率的影响没有区分度，无法确定阀值。

# 任务 4.2 逻辑回归算法

逻辑回归用于处理因变量为分类变量的回归问题，常见的是二分类或二项分布问题，也可以处理多分类问题，它实际上属于一种分类方法，如判断一封邮件是否为垃圾邮件。逻辑回归一般是提供样本和已知模型求回归参数。

## 子任务 4.2.1 基本思想

在前面讲述的回归模型中处理的因变量都是数值型区间变量，建立的模型描述是因变量的期望与自变量之间的线性关系。例如常见的线性回归模型：

$$h_\theta(x)=\theta_0+\theta_1x_1+\theta_2x_2+\cdots+\theta_nx_n \tag{4.1}$$

而在采用回归模型分析实际问题中研究的变量往往不全是区间变量，而是顺序变量或属性变量，如二项分布问题。通过分析年龄、性别、体质指数、平均血压、疾病指数等指标，判断一个人是否患糖尿病，$Y$=0 表示未患病，$Y$=1 表示患病，这里的响应变量是一个两点（0-1）分布变量，不能用 $h$ 函数连续的值预测因变量 $Y$（只能取 0 或 1）。

总之，线性回归模型通常处理因变量是连续变量的问题，如果因变量是定性变量，线性回归模型就不再适用了，须采用逻辑回归模型解决。

## 子任务 4.2.2 算法原理

逻辑回归模型是一种分类模型，用条件概率分布的形式表示为 $P(Y|X)$，这里随机变量 $X$ 的取值为 $n$ 维实数向量，例如 $x=(x(1),x(2),\cdots,x(n)),x=(x(1),x(2),\cdots,x(n))$，$Y$ 的取值为 0 或 1，即

$$p(Y=1|x)=\frac{\exp(wx+b)}{1+\exp(wx+b)} \tag{4.2}$$

笔记

$$p(Y=0|0)=\frac{1}{1+\exp(wx+b)} \tag{4.3}$$

或

$$\varphi(x)=\frac{1}{1+\exp^{-w\boldsymbol{T}x-b}} \tag{4.4}$$

假设有一个二分类问题，输出为 $y \in \{0, 1\}$，二线性回归模型 $z=w\boldsymbol{T}x+b$ 是一个实数值，我们希望有一个理想的阶跃函数帮我们实现 $z$ 值到 0/1 值的转化，于是找到 Sigmoid 函数代替：

$$g(z)=\frac{1}{1+e^{-z}} \tag{4.5}$$

有了 Sigmoid 函数之后，由于其值的取值范围为 [0,1]，就可以将其视为类 1 的后验概率估计 $p(y=1|X)$。说白了就是，如果有一个测试点 $x$，就可以把 Sigmoid 函数算出的结果当作该点 $x$ 属于类别 1 的概率大小。

于是，非常自然地，我们把 Sigmoid 函数计算得到的大于或等于 0.5 的值归为类别 1，小于 0.5 的值归为类别 0。

# 任务 4.3 逻辑回归应用示例

## 子任务 4.3.1 数据来源及背景

数据来源为 https://www.kaggle.com/jiangzuo/hr-comma-sep/version/1。

该数据集包含 14999 个样本以及 10 个特征，通过现有员工是否离职的数据，建立模型预测有可能离职的员工。

## 子任务 4.3.2 数据概览

### 1. 查看前 2 行和后 2 行数据

```
import pandas as pd
df = pd.read_csv(r'D:\Data\HR_comma_sep.csv')
pd.set_option('display.max_rows', 4)
df
```

结果如图 4-1 所示。10 个字段分别是：员工对公司的满意度、最新考核评估、项目数、平均每月工作时长、工作年限、是否出现工作事故、是否离职、过去 5 年是否升职、岗位、薪资水平。

| | satisfaction_level | last_evaluation | number_project | average_montly_hours | time_spend_company | Work_accident | left | promotion_last_5years | sales | salary |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.38 | 0.53 | 2 | 157 | 3 | 0 | 1 | 0 | sales | low |
| 1 | 0.80 | 0.86 | 5 | 262 | 6 | 0 | 1 | 0 | sales | medium |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 14997 | 0.11 | 0.96 | 6 | 280 | 4 | 0 | 1 | 0 | support | low |
| 14998 | 0.37 | 0.52 | 2 | 158 | 3 | 0 | 1 | 0 | support | low |

14999 rows × 10 columns

图 4-1　离职员工的 10 个特征

可以看到，除岗位以及薪资水平是字符型外，其余均是数值型。

### 2. 查看数据类型等信息

```
df.info ( )
<class>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
satisfaction_level                    14999 non-null float64
last_evaluation                       14999 non-null float64
number_project                        14999 non-null int64
average_montly_hours                  14999 non-null int64
time_spend_company                    14999 non-null int64
Work_accident                         14999 non-null int64
left                                  14999 non-null int64
promotion_last_5years                 14999 non-null int64
sales                                 14999 non-null object
salary                                14999 non-null object
dtypes: float64 ( 2 ), int64 ( 6 ), object ( 2 )
memory usage: 1.1+ MB</class>
```

前两个特征为浮点型，后两个特征为字符型，其余特征为整型，且均无缺失值。

### 3. 描述性统计

```
df.describe ( )
df.describe (include=['O']) .T
```

满意度：范围 0.09~1，中位数 0.640，均值 0.613；
最新考核评估：范围 0.36~1，中位数 0.720，均值 0.716；
项目数：范围 2~7 个，中位数 4，均值 3.8；
平均每月工作时长：范围 96~310 小时，中位数 200，均值 201；
工作年限：范围 2~10 年，中位数 3，均值 3.5；
工作中出现工作事故的占：14.46%；
已经离职的占：23.81%；
过去 5 年升职的占：2.13%。

如图 4-2 所示。

| | satisfaction_level | last_evaluation | number_project | average_montly_hours | time_spend_company | Work_accident | left | promotion_last_5years |
|---|---|---|---|---|---|---|---|---|
| count | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 | 14999.000000 |
| mean | 0.612834 | 0.716102 | 3.803054 | 201.050337 | 3.498233 | 0.144610 | 0.238083 | 0.021268 |
| std | 0.248631 | 0.171169 | 1.232592 | 49.943099 | 1.460136 | 0.351719 | 0.425924 | 0.144281 |
| min | 0.090000 | 0.360000 | 2.000000 | 96.000000 | 2.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.440000 | 0.560000 | 3.000000 | 156.000000 | 3.000000 | 0.000000 | 0.000000 | 0.000000 |
| 50% | 0.640000 | 0.720000 | 4.000000 | 200.000000 | 3.000000 | 0.000000 | 0.000000 | 0.000000 |
| 75% | 0.820000 | 0.870000 | 5.000000 | 245.000000 | 4.000000 | 0.000000 | 0.000000 | 0.000000 |
| max | 1.000000 | 1.000000 | 7.000000 | 310.000000 | 10.000000 | 1.000000 | 1.000000 | 1.000000 |

图 4-2　分类统计 14999 个样本

员工岗位有 10 种，其中最多的是销售，多达 4140 人。薪资水平共有 3 个等级，最多的是低等，多达 7316 人。

## 子任务 4.3.3 数据预处理

由于没有缺失值，因此不用处理缺失值。对于记录来说，其没有唯一标识的字段，因此会存在重复记录，这里采取不处理。

异常值：通过箱线图查看异常值。

```
import seaborn as sns
fig, ax = plt.subplots(1, 5, figsize=(12, 2))
sns.boxplot(x=df.columns[0], data=df, ax=ax[0])
sns.boxplot(x=df.columns[1], data=df, ax=ax[1])
sns.boxplot(x=df.columns[2], data=df, ax=ax[2])
sns.boxplot(x=df.columns[3], data=df, ax=ax[3])
sns.boxplot(x=df.columns[4], data=df, ax=ax[4])
```

除工作年限外，其他均无异常值。该异常值也反映了该公司员工以年轻人为主，如图 4-3 所示。

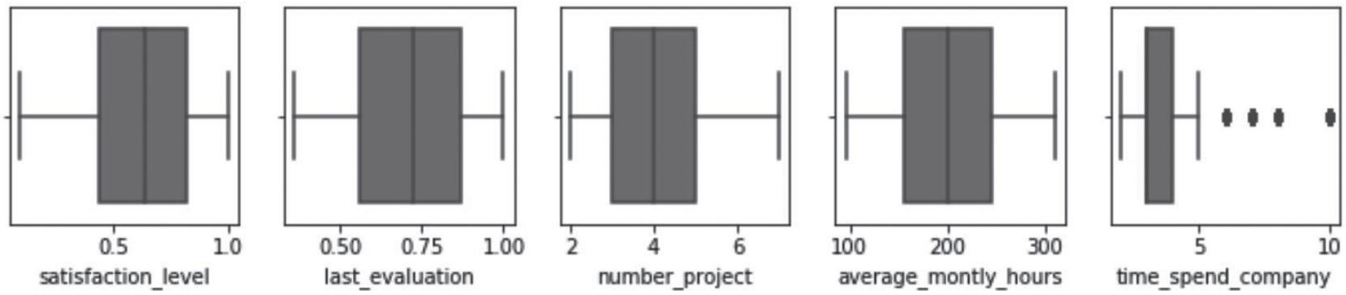



图 4-3　该箱线图反映了公司员工以年轻人为主

## 子任务 4.3.4 可视化分析

笔记

### 1. 人力资源总体情况

```
from pyecharts import Pie
attr = ["离职", "在职"]
v1 =[df.left.value_counts ( ) [1], df.left.value_counts ( ) [0]]
pie = Pie ("该公司人力资源总体情况", title_pos='center')
pie.add (
    "",
    attr,
    v1,
    radius=[35, 65],
    label_text_color=None,
    is_label_show=True,
    legend_orient="vertical",
    legend_pos="left",
)
pie.render ( )
```

离职 3571 人，占比 23.81%；在职 11428 人，占比 76.19%，如图 4-4 所示。

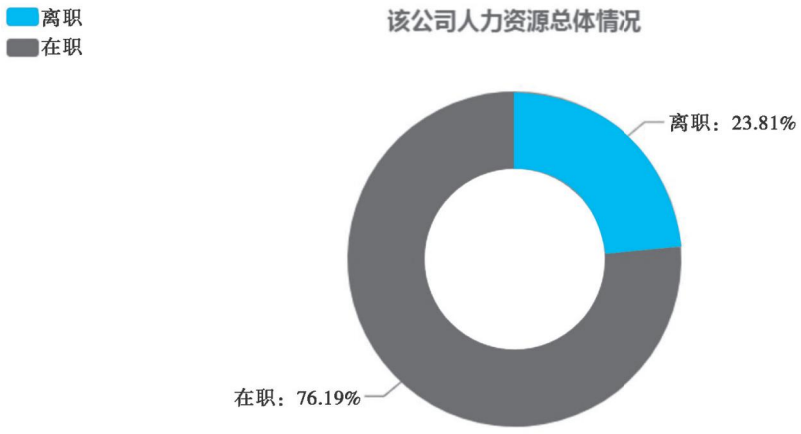



图 4-4　该公司人力资源总体情况

### 2. 对公司满意度与是否离职的关系

```
from pyecharts import Boxplot
# 字段重命名
df.columns=['satisfaction', 'evaluation', 'project', 'hours', 'years_work','work_accident', 'left', 'promotion',
'department', 'salary']
# 绘制箱线图
boxplot = Boxplot("对公司满意度与是否离职关系图", title_pos='center')
```

```
x_axis = ['在职', '离职']
y_axis = [df[df.left == 0].satisfaction.values, df[df.left == 1].satisfaction.values]
boxplot.add("", x_axis, boxplot.prepare_data(y_axis))
boxplot.render()
```

就中位数而言，离职人员对公司满意度相对较低，且离职人员对公司满意度整体波动较大。另外，离职人员中没有满意度为 1 的评价，如图 4-5 所示。

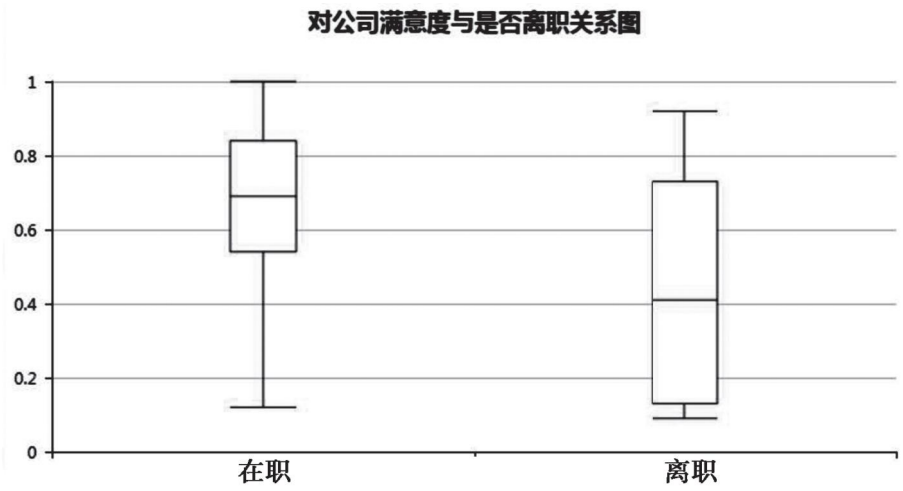



图 4-5　对公司满意度与是否离职的关系

### 3. 最新考核评估与是否离职的关系

```
boxplot = Boxplot("最新考核评估与是否离职关系图", title_pos='center')
x_axis = ['在职', '离职']
y_axis = [df[df.left == 0].evaluation.values, df[df.left == 1].evaluation.values]
boxplot.add("", x_axis, boxplot.prepare_data(y_axis))
boxplot.render()
```

就中位数而言，离职人员的最新考核评估相对较高，但其波动也大，如图 4-6 所示。

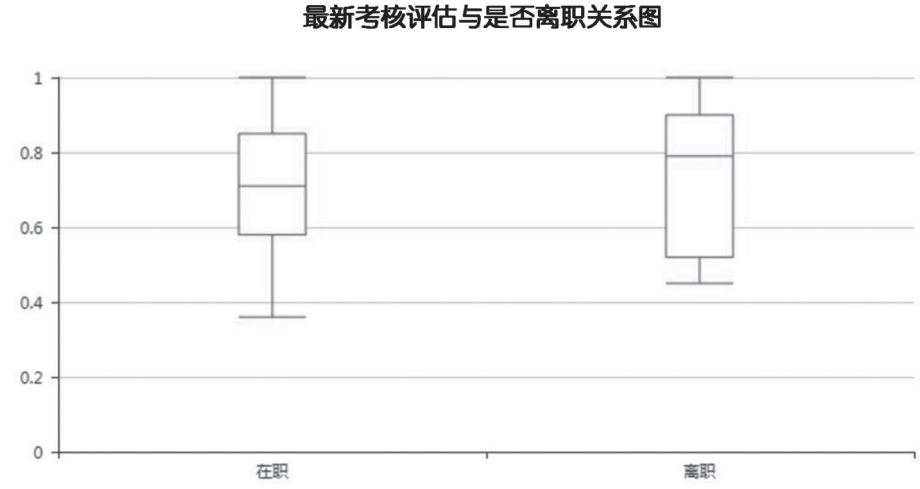



图 4-6　最新评估与是否离职的关系

### 4. 所参加项目与是否离职的关系

```
from pyecharts import Bar, Pie, Grid
# 按照项目数分组分别求离职人数和所有人数
project_left_1 = df[df.left == 1].groupby('project')['left'].count()
project_all = df.groupby('project')['left'].count()
# 分别计算离职人数和在职人数所占比例
project_left1_rate = project_left_1 / project_all
project_left0_rate = 1 - project_left1_rate
attr = project_left1_rate.index
bar = Bar("所参加项目数与是否离职关系图", title_pos='10%')
bar.add("离职", attr, project_left1_rate, is_stack=True)
bar.add("在　职", attr, project_left0_rate, is_stack=True, legend_pos="left", legend_
orient="vertical")
# 绘制圆环图
pie = Pie("各项目数所占百分比", title_pos='center')
pie.add("", project_all.index, project_all, radius=[35, 60], label_text_color=None, is_label_
show=True, legend_orient="vertical", legend_pos="67%")
grid = Grid(width=1200)
grid.add(bar, grid_right="67%")
grid.add(pie)
grid.render()
```

如图 4-7 所示，可以发现以下两点：

（1）离职人员所占比例随着项目数的增多而增大，2 个项目数是特例；

（2）离职人员比例较高的项目数 2、6、7 在总项目数中所占百分比相对较少。项目数为 2 的这部分人可能是工作能力不被认可，其离职人数也相对较高；项目数为 6、7 的这部分人工作能力较强，其在其他企业可能有更好的发展，自然离职比例也相对较高。

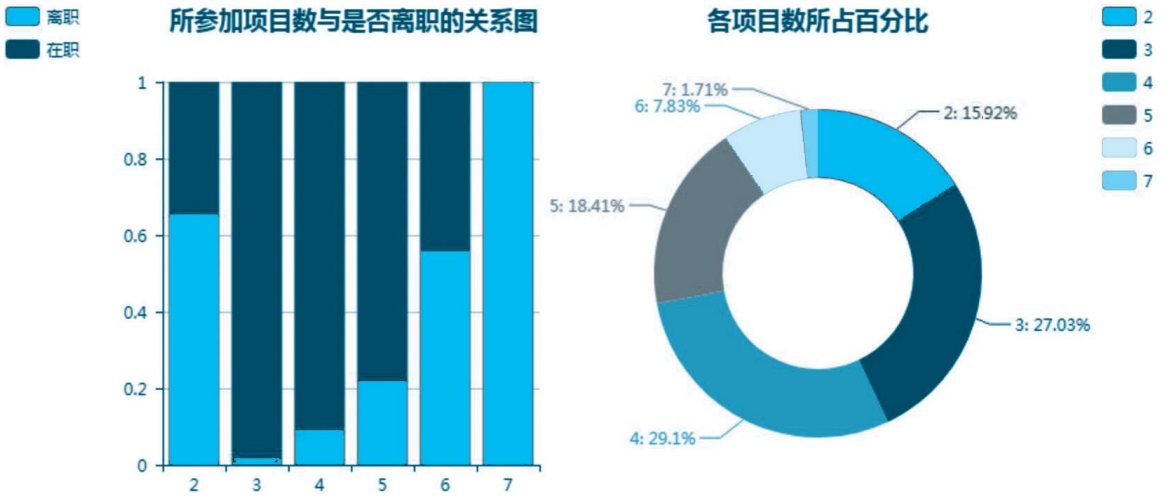



图 4-7　所参加项目与是否离职的关系，各项目数所占百分比

笔记

### 5. 平均每月工作时长与是否离职的关系

```
boxplot = Boxplot("平均每月工作时长与是否离职关系图", title_pos='center')
x_axis = ['在职', '离职']
y_axis = [df[df.left == 0].hours.values, df[df.left == 1].hours.values]
boxplot.add("", x_axis, boxplot.prepare_data(y_axis))
boxplot.render()
```

通过图 4-8 可以看到，离职人员的平均每月工作时长相对较长，每月按照 22 个工作日计算，每日工作时数的中位数为 10.18 小时，最大值为 14.09 小时。

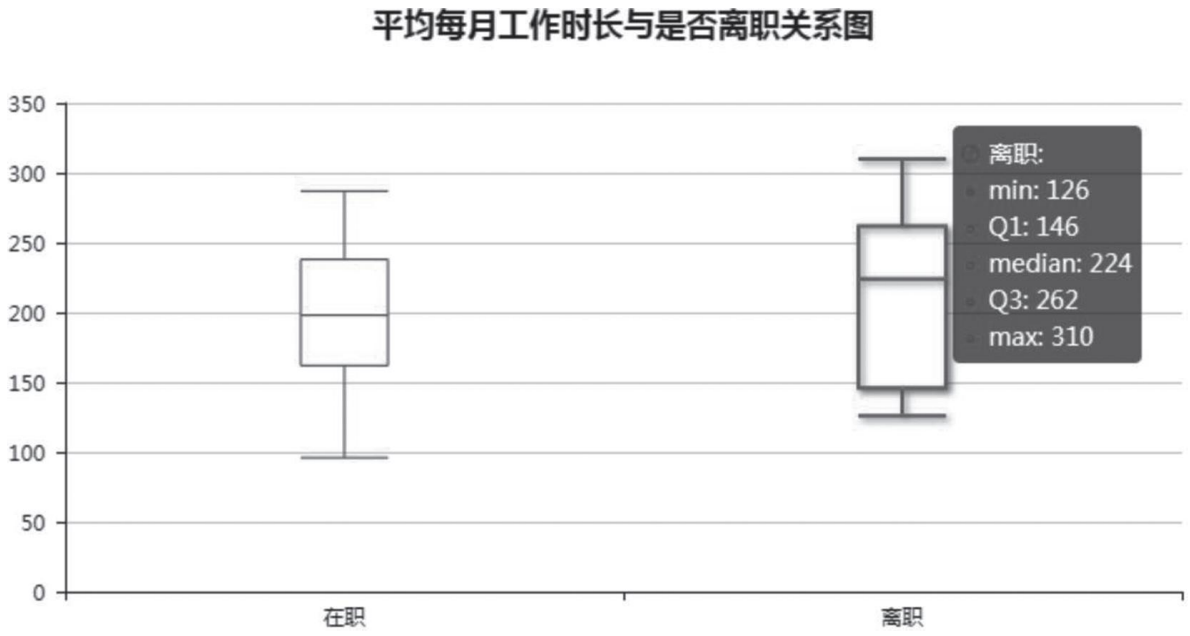



图 4-8　平均每月工作时长与是否离职的关系

### 6. 工作年限与是否离职的关系

```
from pyecharts import Bar, Pie, Grid
# 按照工作年限分别求离职人数和所有人数
years_left_0 = df[df.left == 0].groupby('years_work')['left'].count()
years_all = df.groupby('years_work')['left'].count()
# 分别计算离职人数和在职人数所占比例
years_left0_rate = years_left_0 / years_all
years_left1_rate = 1 - years_left0_rate
attr = years_all.index
bar = Bar("工作年限与是否离职的关系图", title_pos='10%')
bar.add("离职", attr, years_left1_rate, is_stack=True)
bar.add("在职", attr, years_left0_rate, is_stack=True, legend_pos="left",
```

```
    legend_orient="vertical")
    # 绘制圆环图
    pie = Pie("各工作年限所占百分比", title_pos='center')
    pie.add("", years_all.index, years_all, radius=[35, 60], label_text_color=None,
    is_label_show=True, legend_orient="vertical", legend_pos="67%")
    grid = Grid(width=1200)
    grid.add(bar, grid_right="67%")
    grid.add(pie)
    grid.render()
```

笔记

如图 4-9 所示，可以得出：

（1）在各工作年限中，离职人员较集中于 3~6 年，6 年以上相对稳定；

（2）企业中 3 年人数所占百分比最多，其次是 2 年，主要以年轻人为主。

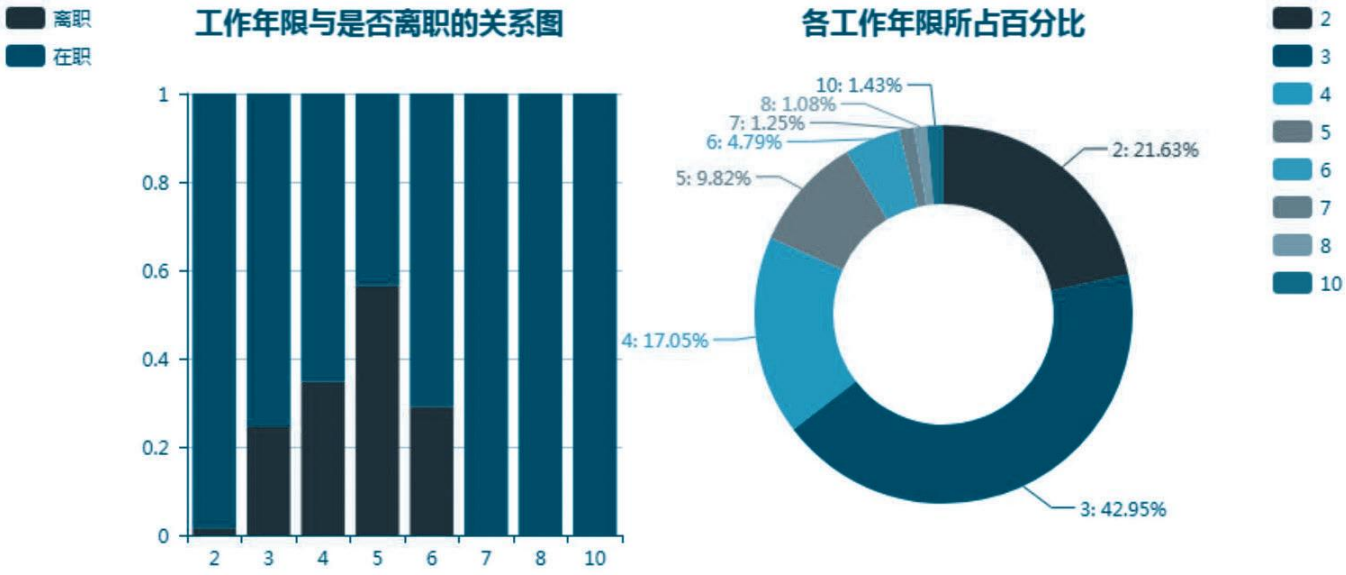



图 4-9　工作年限与是否离职的关系，各工作年限所占百分比

### 7. 是否发生工作事故与是否离职的关系

```
    from pyecharts import Bar
    accident_left = pd.crosstab(df.work_accident, df.left)
    attr = accident_left.index
    bar = Bar("是否发生工作事故与是否离职关系图", title_pos='center')
    bar.add("离职", attr, accident_left[1], is_stack=True)
    bar.add("在　职", attr, accident_left[0], is_stack=True, legend_pos="left",
legend_orient="vertical", is_label_show=True)
    bar.render()
```

笔记

如图 4-10 所示，可以看到少部分出现工作事故，且其中有较少部分人离职。

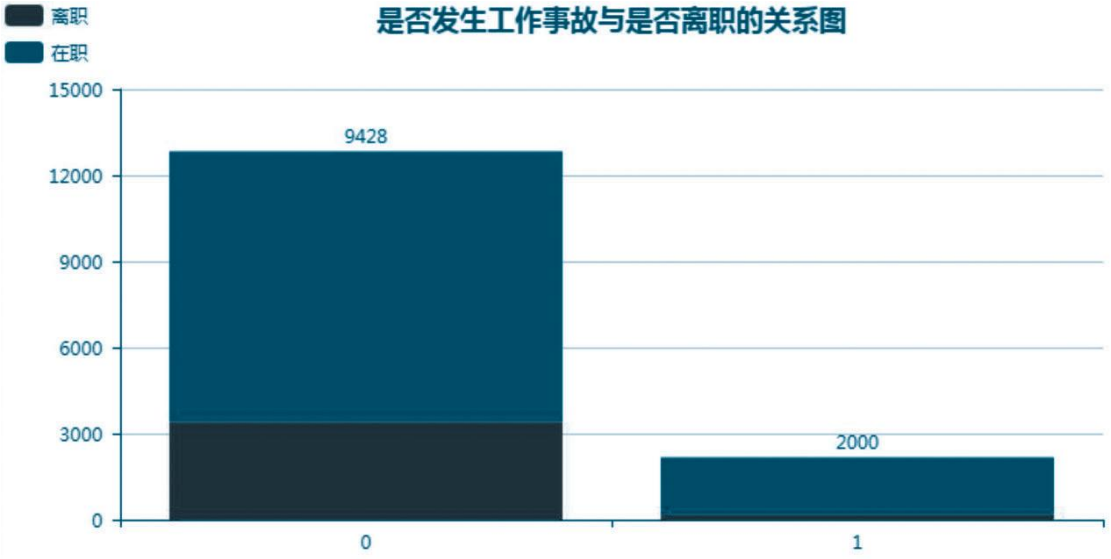



图 4-10　是否发生工作事故与是否离职的关系

### 8. 5 年内是否升职与是否离职的关系

```
    promotion_left = pd.crosstab(df.promotion, df.left)
    attr = promotion_left.index
    bar = Bar("5 年内是否升职与是否离职关系图 ", title_pos='center')
    bar.add(" 离职 ", attr, promotion_left[1], is_stack=True)
    bar.add(" 在　职 ", attr, promotion_left[0], is_stack=True, legend_pos="left" ,
legend_orient="vertical", is_label_show=True)
    bar.render()
```

如图 4-11 所示，5 年内多数人没有升职，离职率就相对较高。

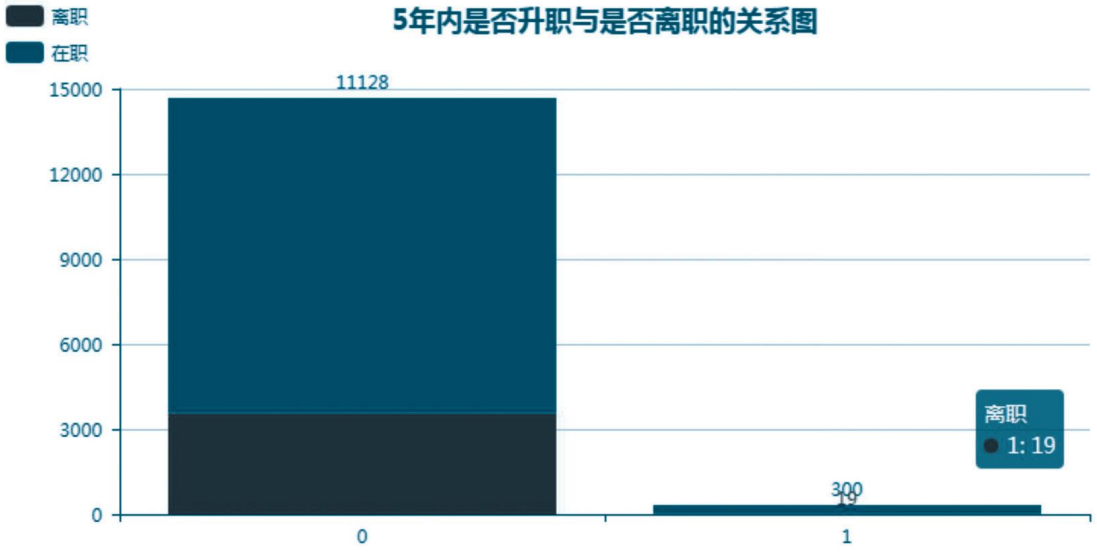



图 4-11　5 年内是否升职与是否离职的关系

### 9. 岗位与是否离职的关系

```
    # 分别计算各岗位离职人员比例和各岗位占总体百分比
    department_left_0 = df[df.left == 0].groupby('department')['left'].count()
    department_all = df.groupby('department')['left'].count()
    department_left0_rate = department_left_0 / department_all
    department_left1_rate = 1 - department_left0_rate
    attr = department_all.index
    bar = Bar("岗位与离职比例的关系图", title_top='40%')
    bar.add("离职", attr, department_left1_rate, is_stack=True)
    bar.add("在　职", attr, department_left0_rate, is_stack=True, is_datazoom_show=True, xaxis_
interval=0, xaxis_rotate=30, legend_top="45%", legend_pos="80%")
    # 绘制圆环图
    pie = Pie("各个岗位所占百分比", title_pos='left')
    pie.add("", department_all.index, department_all,center=[50, 23], radius=[18, 35], label_text_
color=None, is_label_show=True, legend_orient="vertical", legend_pos="80%", legend_top="4%")
    grid = Grid(width=1200, height=700)
    grid.add(bar, grid_top="50%", grid_bottom="25%")
    grid.add(pie)
    grid.render()
```

如图 4-12 所示，可以看出：

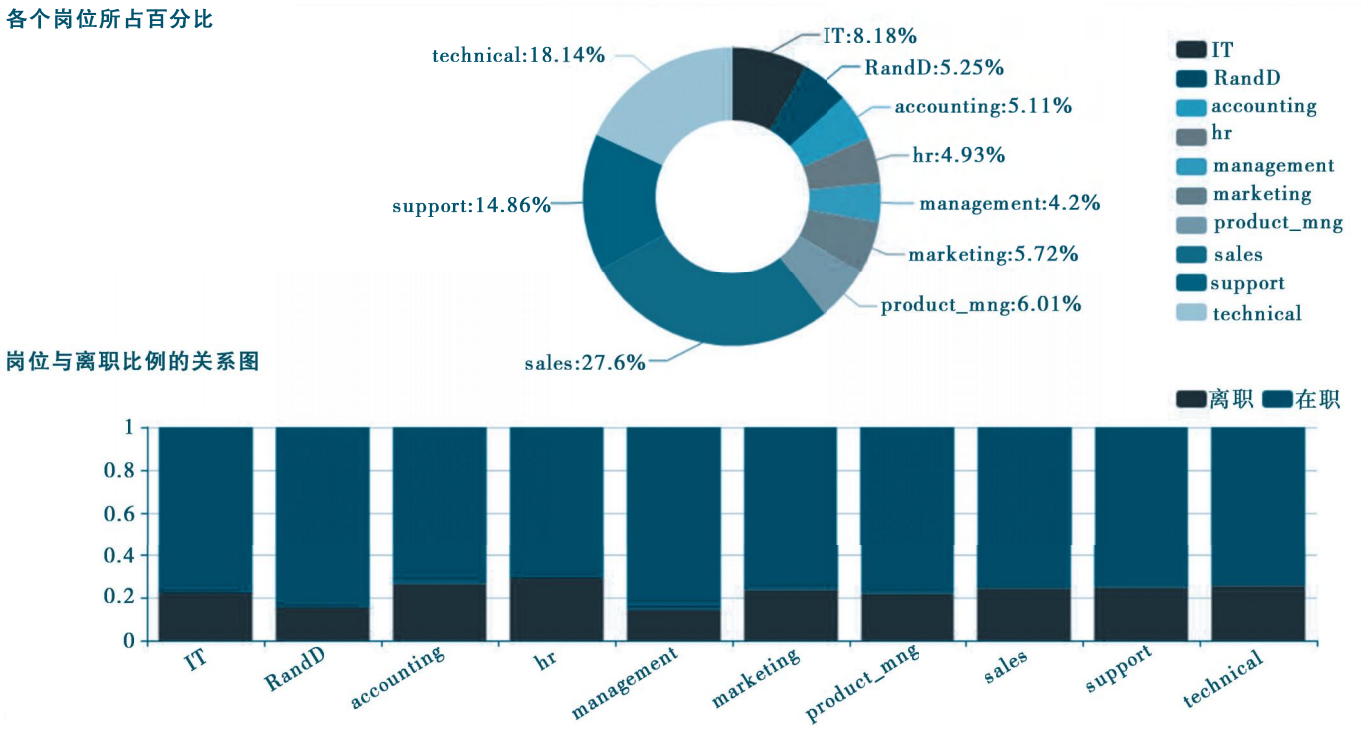



图 4-12　各个岗位所占百分比，岗位与离职比例的关系

笔记

（1）销售岗位所占百分比最多，达到 27.6%，最少是管理层，其所占百分比是 4.2%；

（2）令人意外的是 hr 岗位离职比例最大。

### 10. 薪资水平与是否离职的关系

```
from pyecharts import Bar
# 按照薪资水平分别求离职人数和所有人数
salary_left = pd.crosstab(df.salary, df.left).sort_values(0, ascending = False)
attr = salary_left.index
bar = Bar("薪资水平与是否离职关系图", title_pos='center')
bar.add("离职", attr, salary_left[1], is_stack=True)
bar.add("在　职", attr, salary_left[0], is_stack=True, legend_pos="left", legend_
orient="vertical", is_label_show=True)
bar.render()
```

薪资分为三个水平：低等、中等、高等；低等水平离职人数最多，所占比例也最大，而高等水平离职人数最少，如图 4-13 所示。

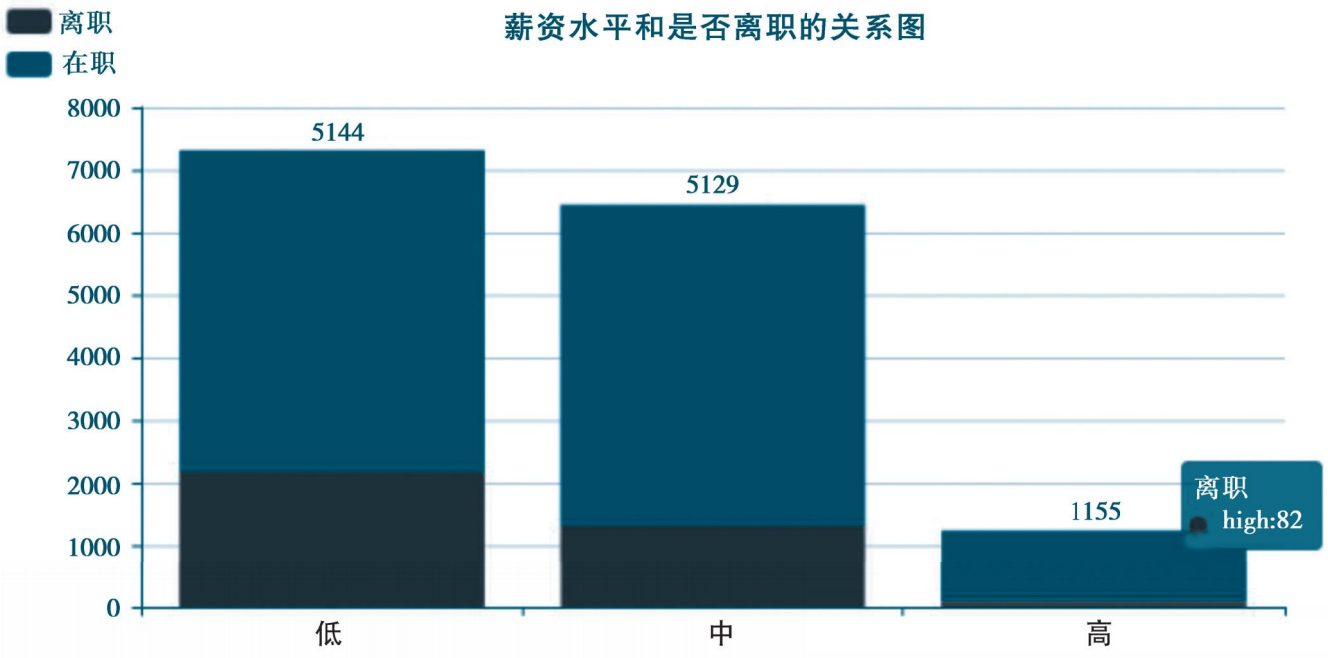



图 4-13　薪资水平与是否离职的关系

## 子任务 4.3.5 特征工程

### 1. 离散型数据处理

离散型数据可分为两种：一种是定序；一种是定类。

（1）定序。薪资水平含有顺序意义，因此将其字符型转化为数值型。

笔记

```
df['salary'] = df.salary.map({"low": 0, "medium": 1, "high": 2})
df.salary.unique()

array([0, 1, 2], dtype=int64)
```

（2）定类。岗位是定类型变量，对其进行 one-hot 编码，这里直接利用 pandas 的 get_dummies（ ）方法。

```
df_one_hot = pd.get_dummies(df, prefix="dep")
df_one_hot.shape

(14999, 19)
```

### 2. 连续型数据处理

逻辑回归模型能够适应连续型变量，因此可以不用进行离散化处理，又由于多个特征之间差异较大会造成梯度下降算法收敛速度变慢，故进行归一化处理。

```
# 采用 max-min 归一化方法
hours = df_one_hot['hours']
df_one_hot['hours'] = df_one_hot.hours.apply(lambda x: (x-hours.min()) / (hours.max()-hours.min()))
```

### 3. 相关系数

两个变量均是连续型且具有线性关系，则可以使用皮尔逊（Pearson）相关系数，否则使用斯皮尔曼（Spearman）相关系数，这里采用斯皮尔曼相关系数。

计算相关系数。

```
correlation=df_one_hot.corr(method = "spearman")
plt.figure(figsize=(18, 10))
```

绘制热力图，如图 4-14 所示。

```
sns.heatmap(correlation, linewidths=0.2, vmax=1, vmin=-1, linecolor='w',fmt='.2f', annot=True,annot_kws={'size':10},square=True)
```

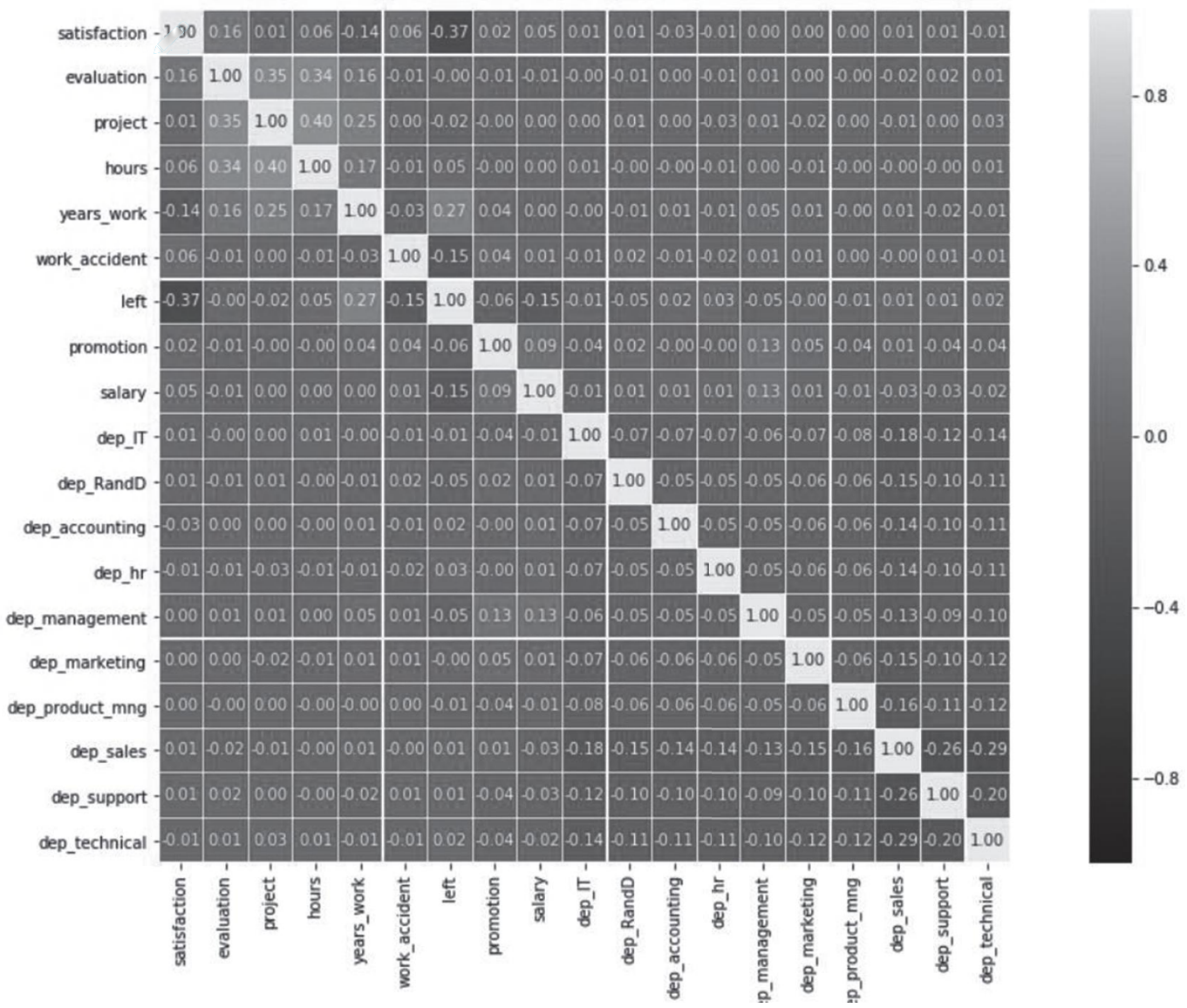



图 4-14　绘制热力图

## 子任务 4.3.6 逻辑回归模型

### 1. 划分数据集

```
    from sklearn.model_selection import train_test_split
    # 划分训练集和测试集
    X = df_one_hot.drop(['left'], axis=1)
    y = df_one_hot['left']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=1)
```

笔记

### 2. 训练模型

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
print(LR.fit(X_train, y_train))
print("训练集准确率：", LR.score(X_train, y_train))
print("测试集准确率：", LR.score(X_test, y_test))

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100, multi_class='warn', n_jobs=None, penalty='l2',
random_state=None, solver='warn', tol=0.0001, verbose=0, warm_start=False)
训练集准确率：0.7978998249854155
测试集准确率：0.7966666666666666
```

参考官方文档说明，参数 C 是正则化项参数的倒数，C 的数值越小，惩罚的力度越大 . penalty 可选 L1、L2 正则化项，默认是 L2 正则化。

参数 solver 可选 {'newton-cg'，'lbfgs'，'liblinear'，'sag'，'saga'} 这 5 个优化算法。

newton-cg、lbfgs 是拟牛顿法，liblinear 是坐标轴下降法，sag、saga 是随机梯度下降法，saga 适用于 L1 和 L2 正则化项，而 sag 只用于 L2 正则化项。

```
# 指定随机梯度下降优化算法
LR = LogisticRegression(solver='saga')
print(LR.fit(X_train, y_train))
print("训练集准确率：", LR.score(X_train, y_train))
print("测试集准确率：", LR.score(X_test, y_test))

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100, multi_class='warn', n_jobs=None, penalty='l2',
random_state=None, solver='saga', tol=0.0001, verbose=0, warm_start=False)
训练集准确率：0.7980665055421285
测试集准确率：0.7973333333333333
```

选择随机梯度下降法后，训练集准确率和测试集准确率都略有提升。

### 3. 调参

```
# 用准确率进行 10 折交叉验证选择合适的参数 C
from sklearn.linear_model import LogisticRegressionCV
Cs = 10**np.linspace(-10, 10, 400)
lr_cv = LogisticRegressionCV(Cs=Cs, cv=10, penalty='l2', solver='saga', max_
iter=10000, scoring='accuracy')
```

笔记

```
lr_cv.fit(X_train, y_train)
lr_cv.C_
array([25.52908068])
用该参数进行预测
LR = LogisticRegression(solver='saga', penalty='l2', C=25.52908068)
print("训练集准确率：", LR.score(X_train, y_train))
print("测试集准确率：", LR.score(X_test, y_test))
训练集准确率：0.7984832069339112
测试集准确率：0.798
```

训练集准确率和测试集准确率均有所提升，对于二分类问题，准确率有时不是很好的评估方法，这时就需要用到混淆矩阵。

### 4. 混淆矩阵

```
from sklearn import metrics
X_train_pred = LR.predict(X_train)
X_test_pred = LR.predict(X_test)
print('训练集混淆矩阵：')
print(metrics.confusion_matrix(y_train, X_train_pred))
print('测试集混淆矩阵：')
print(metrics.confusion_matrix(y_test, X_test_pred))

训练集混淆矩阵：
[[8494  647]
[1771 1087]]
测试集混淆矩阵：
[[2112  175]
[ 431  282]]
from sklearn.metrics import classification_report
print('训练集：')
print(classification_report(y_train, X_train_pred))
print('测试集：')
print(classification_report(y_test, X_test_pred))
训练集：
              precision    recall  f1-score   support
           0       0.83      0.93      0.88      9141
           1       0.63      0.38      0.47      2858
   micro avg       0.80      0.80      0.80     11999
   macro avg       0.73      0.65      0.67     11999
weighted avg       0.78      0.80      0.78     11999
```

测试集：

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.92 | 0.87 | 2287 |
| 1 | 0.62 | 0.40 | 0.48 | 713 |
| micro avg | 0.80 | 0.80 | 0.80 | 3000 |
| macro avg | 0.72 | 0.66 | 0.68 | 3000 |
| weighted avg | 0.78 | 0.80 | 0.78 | 3000 |

在训练集上有 0.83 的精准率和 0.93 的召回率，在测试集上有 0.83 的精准率和 0.92 的召回率。

笔记