第 1 章
金融智能的基本原理

章前导读

本章回顾人工智能发展的历史，并讨论金融智能中的常见场景以及背后使用的关键技术。结合国内外金融智能的发展现状，进一步探讨金融智能的发展方向以及遇到的关键挑战，介绍 Python 的基础知识和机器学习模型评估中的问题。

本章学习目标

首先，读者需要了解人工智能的概念，并理解人工智能发展过程遇到的挑战；其次，聚焦金融智能的方法与场景，鼓励读者思考金融智能应用和其他人工智能场景的区别和联系；然后为读者指出金融智能未来发展的方向；最后，读者需要了解 Python 的基础知识及如何进行机器学习模型评估。

1.1 人工智能概述与历史

1.1.1 人工智能概述

1. 概念与定义

随着互联网时代的到来，大量的数据得以记录，人类积累的数据井喷式增长。摩尔定律指出，集成电路上的晶体管数量每两年大约会增加一倍，这通常意味着计算能力也会呈指数级增长。数据和算力的持续积累，也为以深度学习为代表的智能算法在各个领域发挥

作用提供了支撑。在计算机视觉领域，图片识别精确度的提升促进了大批成功的商业应用诞生，包括智能安防和智能驾驶。在信息流推荐领域，相较于传统媒体，智能算法改变了信息分发的方式，通过理解用户的偏好推送更精确的内容。2022 年以来，以 ChatGPT 为代表的大语言模型，更是拉开了通用人工智能时代的大幕。

人工智能（artificial intelligence，AI）是计算机科学的一个分支，它是致力于创造能够执行通常需要人类智能才能完成的任务的智能机器或软件。人工智能作为一种使机器能够模仿、延伸并扩展人类的认知功能的工程，包括了学习、推理、自适应、感知和交互等多种能力。人工智能拓宽了人类的能力边界，极大地提升了生产效率。因此，理解人工智能的原理，有助于我们进一步拓宽人工智能应用的范围，从而更好地服务社会生活的需要。

2. 机器学习与深度学习

机器学习（machine learning，ML）最初来源于统计学方法。以线性模型中的最小二乘法举例，我们可以通过 X 来解释 Y 的变化。然而当 X 的观测值数量小于 X 的维度时，又会遭遇“维度的诅咒”，模型在数值上没办法求解。在这个时候，我们需要借助于以 LASSO 为代表的线性机器学习方法，加入正则化的惩罚项，从而实现模型的求解。然而很多时候数据之间也存在着非线性特征和交互效应，因此我们可以进一步借助包括随机森林、梯度提升树等方法进一步改进模型的预测效果。

神经网络算法是机器学习模型的代表性算法，它能够更好地从数据中学习到复杂模式，在样本外也表现出较好的泛化能力。然而传统的神经网络算法随着网络层数的增加，可能会出现梯度消失的情况，从而限制了模型的预测效果。以深度学习（deep learning，DL）为代表的机器学习算法，通过采用特定的方法增加网络层数，可以更好地使用多层神经网络来模拟人类大脑处理和解析数据。

常见的深度学习算法包括卷积神经网络（convolutional neural networks，CNNs）和循环神经网络（recurrent neural networks，RNNs）。其中卷积神经网络被广泛应用于计算机视觉中，以斯坦福大学李飞飞教授举办的 ImageNet 竞赛为例，表现最好的深度学习算法识别物体的准确率已经超过了人眼识别的准确率。循环神经网络也被广泛应用于文本生成、机器翻译等领域。同时近些年来，以 Transformer 为主要架构的大语言模型进一步拓展了深度学习算法的应用。深度学习算法在工程上的成功应用，深刻改变了相关行业，带来的潜在影响和冲击也广泛引起社会公众、产业界和政府部门的讨论。

3. 算法、算力和数据

算法、算力和数据是人工智能发展的三个关键因素。在人工智能发展的过程中，提升关键要素的投入对于推动该领域的创新和进步具有重要价值。这三个因素相互依赖，共同驱动了人工智能的发展。强大的算力支持更复杂的算法，而算法的改进又能更有效地利用数据。算法、算力和数据构成了一个相互促进的框架，共同推动了技术的不断进步和应用的不断拓展。

算法是人工智能的基础，包括从基础的统计方法到复杂的神经网络和优化技术。人工智能发展的历史也见证了算法的不断创新，特别是一系列深度学习算法的出现将人工智能应用的发展带上了一个新的高度。随着算法变得更加复杂，尤其是深度学习的兴起，对计算资源的需求显著增加。图形处理器（graphics processing unit，GPU）的发展、云计算的普及以及专用硬件（如 TPU）的出现极大地推动了人工智能的计算能力。数据是训练和优化人工智能模型的关键。有效地收集、清洗和处理数据是实现有效人工智能应用的基础。

1.1.2 人工智能发展历史

人工智能（AI）的发展经历了多个历史阶段，每个阶段都具有其独特的特点和技术突破。AI 的概念可以追溯到 20 世纪，最初与研究如何使机器模仿人类的思维过程有关。在早期探索阶段（1950s—1970s），随着计算机科学的诞生，AI 作为一个学术领域开始形成，主要研究集中在问题解决和逻辑推理等理论方法上。1956 年的达特茅斯会议首次使用了“人工智能”这一术语。然而，AI 随后进入了第一次低谷期（AI winters，1970s—1980s），由于技术和算力的限制，导致对 AI 过高的期望难以实现，于是投入资金减少，公众和研究社区的兴趣下降。

而 1980 年代见证了专家系统的兴起，AI 在金融、医疗和工程等领域找到了实际应用，由此 AI 进入了复兴和专家系统阶段（1980s—1990s）。随着 1990 年代的到来，互联网的普及带来了大量数据，为机器学习提供了有力支撑。同时，以支持向量机和随机森林为代表的算法的发展也推动了机器学习的进步。2010 年以来，AI 进入深度学习和大数据融合发展的阶段。GPU 硬件的进步极大地提高了深度学习模型的训练效率，同时以卷积神经网络为代表的深度学习算法在图像识别等领域大放光彩。

总体来看，每个阶段的 AI 发展都受到当时科技水平、经济状况和社会需求的影响，尽管关于 AI 的相关研究几经起落，但算法、算力和数据等关键要素的突破共同推动了 AI 的发展。

1.2 金融智能概述与场景

1.2.1 金融智能概述

1. 概念与定义

金融智能是融合金融专业知识和以人工智能为代表的先进技术的综合智能系统。它的核心在于通过分析大量的金融数据来提供深入的洞察、支持决策，并优化金融服务。金融智能的主要目标是提高金融行业的效率，提升客户体验，同时进行有效的风险控制。

金融智能的实践中广泛地使用了一系列关键技术，包括人工智能、区块链技术、云计

算和高性能计算、大数据分析等。这些技术共同作用，使得金融智能能够有效处理和分析金融市场的海量数据，并支持复杂的数据处理和模型训练。基于金融智能的应用，一方面可以更好地提升传统金融场景的效率，例如在传统的投资策略中引入机器学习算法，可以更好地提炼预测特征的信息，形成更加有效的投资策略；另一方面也可以基于客户的数据定制个性化的金融服务，典型的产品应用包括智能投顾、智慧银行等应用。金融智能的发展受到市场需求和技术进步的双重驱动。金融市场的复杂性和动态性要求更高效和精准的决策支持系统，而人工智能和大数据技术的快速发展则为金融智能提供了坚实的技术基础。

2. 金融智能的特殊性

金融智能作为人工智能在金融领域的应用，相较于其他领域，具有一系列显著不同的特点。金融智能需要更多地依赖金融专业知识，才能更好地发挥技术的优势。

首先，金融市场的高度复杂性和动态性是一个关键特点。金融市场受到诸如经济指标、市场情绪、政治事件等多种因素的影响，使得预测和决策变得极其复杂。此外，金融市场的不断变化要求金融智能系统能够实时适应新的市场条件。金融市场还具有自适应性的特征。以机器学习算法选股策略举例，如果某个策略可以获得超额收益，那么市场上的其他参与者也会追加资金模仿该策略，从而让市场变得更加难以预测，这个策略本身也会失效。

其次，数据的多样性和大量性也是金融智能的显著特征。金融数据既包括结构化数据（如价格、交易量），也包括非结构化数据（如新闻、社交媒体信息），这对计算能力和数据处理技术提出了更高的要求。著名的金融信息处理终端 Bloomberg 就基于金融领域的专有数据集，推出了名为 BloombergGPT 的金融大语言模型，可用于更好地处理金融相关的任务。

最后，金融智能的发展还必须考虑高风险和合规性要求。由于金融决策通常涉及巨大的经济利益，错误的决策可能导致重大损失；然而以神经网络为代表的智能算法的可解释性较差，难以有效地对输出结果进行归因。因此金融智能系统需要对风险高度敏感，审慎对待模型黑箱。此外，金融行业受到严格的法规和政策约束，确保金融智能符合相关合规要求也至关重要。

技术融合和创新驱动也是金融智能的关键特征，金融智能结合了人工智能、大数据、云计算等多种技术，不断推动金融产品和服务的创新。因此，金融智能的开发者和使用者在设计和实施解决方案时必须全面考虑到市场复杂性、数据多样性、风险和合规性等因素，以确保技术的有效性和安全性。

1.2.2 金融智能的常见场景

1. 投资交易

投资交易领域是金融智能一个关键应用场景。投资者使用人工智能（AI）技术来优化和自动化交易决策过程。投资交易通常指买卖金融资产（如股票、债券、期货等）的行为，

其中专业投资者依靠金融智能分析大量数据，进而识别交易机会。这一过程的目的在于使用 AI 技术提高交易的效率和效益，减少人为错误，并快速响应市场的动态变化。近年来，以智能投顾为代表的产品开始为个人投资者提供低门槛的个性化理财服务，通过一系列算法进行资产配置和投资组合管理，自动调整投资组合以优化收益与风险的平衡。

在投资交易中应用的技术主要包括机器学习、深度学习和自然语言处理（NLP）。传统的机器学习算法利用历史数据和实时市场数据训练模型，预测市场走势和识别交易机会。深度学习则借助卷积神经网络（CNN）和循环神经网络（RNN）等算法处理复杂的市场数据，识别模式和趋势。NLP 用于分析财经新闻、报告和社交媒体，以测度市场情绪，并分析潜在影响因素。上述技术手段的使用，进一步提升了投资交易策略的有效性和可靠性，也可以提高市场价格发现和资产定价的效率。

在投资交易领域，金融智能面临的挑战也十分明显。美国知名对冲基金 AQR 在名为“Can machine learn finance”的研究报告中总结了以机器学习为代表的智能算法在金融领域应用面临的挑战。首先是数据规模和质量问题，金融市场的核心任务——收益预测，通常是一个小数据问题。尽管有许多潜在的预测变量，但真正独立的观察数据（如每日或每月的收益数据）非常有限，这使得模型的训练数据不足。金融数据的信噪比通常较低。市场上的大量噪声和随机性事件，使得提取有效的信号变得困难。市场中的预测信号（如风险溢价）往往微弱且难以捕捉。其次，金融市场具备动态和自适应的特征。金融市场是动态变化的，投资者的行为和市场条件会随着时间改变。这意味着某个时期有效的模型或信号，可能在另一时期失效。市场中的竞争使得任何有效的预测信息很快被市场参与者利用，导致市场价格迅速调整，减少了预测信号的有效性。此外，金融领域的数据不仅包括结构化的数据，还包括大量的非结构化数据，如新闻文本、社交媒体内容和图像数据。这些数据的处理和分析需要新的技术和方法。许多新兴的非结构化数据源，如社交媒体数据，其历史记录相对较短，难以进行长期的回溯测试和策略验证。最后，许多机器学习模型（如深度神经网络）被视为黑箱模型，难以解释其内部工作机制。在投资交易领域，理解模型的决策逻辑对于风险管理和合规至关重要，因此需要平衡模型的预测能力和可解释性。

尽管机器学习在投资交易中面临诸多挑战，但通过结合经济理论和人类专业知识，可以逐步克服这些困难，实现更加精准和有效的金融预测和决策。未来发展趋势可能包括融合多源数据、运用增强学习在模拟环境中训练交易策略，以及提高算法的透明度和可解释性以增加投资者的信任。

2. 金融风险管理

金融智能在风险管理领域的应用是其最重要的功能之一，主要涉及使用先进技术来识别、评估、监控和缓解金融风险。风险管理的核心在于识别和分析潜在的金融风险，并采取措施来减轻或控制这些风险。这一过程的目的是在保持收益的同时，降低金融活动中潜在的损失。金融智能在此领域的应用包括使用数据分析技术对历史数据进行分析以识别风

险模式和趋势，建立预测模型来预测市场动态和各类风险，以及利用 NLP 分析非结构化数据以深入了解市场风险。

金融智能在风险管理领域的主要应用包括市场风险管理、操作风险控制等。这些应用涉及监控和评估由市场波动性引起的风险，以及识别和减轻业务流程、系统故障或欺诈行为等内部风险。以传统的市场风险预测为例，机器学习模型可以用于预测市场的波动性。例如，GARCH 模型和 LSTM（长短期记忆网络）可以用来预测金融市场的波动率，这对风险管理至关重要。进一步，机器学习可以帮助改进 VaR 模型的估计，通过对历史数据和市场条件的深度学习，更准确地预测潜在损失的概率分布。

3. 信用评估

信用评估领域是金融智能一个重要的应用场景，主要涉及使用人工智能技术来评估借款人或投资者的信用风险。信用评估的过程包括分析个人或企业的财务状况、历史行为和其他相关因素，以预测他们未来偿还债务的能力和可能性。这一过程的目的是为金融机构提供关于贷款、信用卡发放和其他信用相关产品的决策支持。应用技术主要包括机器学习和数据挖掘、NLP，以及深度学习，这些技术共同助力构建模型以分析历史交易数据、支付记录、用户行为等，从而预测信用风险，并分析非结构化数据如贷款申请者的社交媒体和消费行为。

金融智能在信用评估领域的主要应用包括个人信用评分、企业信用分析和欺诈检测。网络借贷企业通常借助上述应用，评估个人和企业的信用状况，决定贷款额度和利率，同时也用于识别可能的欺诈行为，减少信用风险。然而，信用评估领域也面临着数据隐私和安全的挑战，需要确保模型的精确性和无偏见，并适应经济和市场条件的变化。未来发展趋势可能包括结合传统信用评估和替代数据源进行全面风险评估、进一步自动化决策流程，以及使用模拟环境测试和优化信用评估模型。总体来说，金融智能在信用评估领域的应用为金融机构提供了更有效、更准确的工具，有助于降低信用风险，同时提升决策的速度和质量。

4. 其他场景

金融智能还在多个领域发挥着重要作用，包括支付结算和互联网保险等应用。在支付结算方面，金融智能可以通过区块链技术提供去中心化的账本，使得跨境支付更加高效和安全。区块链技术能够实时记录和验证交易信息，减少了中间环节，从而降低了交易成本并加快了结算速度。例如，Ripple 的支付网络使用区块链技术，实现了快速、安全的跨境支付，解决了传统支付系统中的延迟和高费用问题。在互联网保险领域，金融智能同样发挥着重要作用。通过大数据分析和机器学习算法，保险公司可以更准确地评估风险、定价和检测欺诈行为。大数据技术可以收集和分析大量的用户行为数据，从而帮助保险公司更好地理解客户需求和风险偏好，提供个性化的保险产品和服务。

1.2.3 金融智能中的常用技术

1. 大数据与人工智能技术

在金融智能领域，大数据和人工智能（AI）技术的结合正在革新传统的金融服务和操作方式。大数据的角色在于数据收集和处理。金融机构利用大数据技术收集和存储海量的交易数据、客户行为数据和市场数据等，有效地处理这些数据以提取有价值的信息和洞察。大数据不仅包括传统的金融数据，还包括来自社交媒体、新闻报道、网络行为等的非结构化数据。而 AI 技术的应用包括机器学习、深度学习和自然语言处理。这些技术用于从大量数据中学习模式和趋势，处理复杂的数据集，以及分析金融报告和新闻文章，提取关键信息和情绪。

大数据与 AI 的融合使得金融机构能够进行数据驱动的决策，并提供实时的市场分析和反应。这种融合在风险管理、客户洞察和市场洞察等应用场景中发挥重要作用。例如，利用大数据分析风险因素，结合 AI 模型进行风险预测和控制，以及通过分析客户数据提供更个性化的金融服务和产品。然而，这种技术融合也带来了数据隐私和安全的挑战，以及整合不同数据来源并应用 AI 技术的技术挑战。因此，金融机构需要不断创新和适应这些技术的发展，以在竞争激烈的市场中保持领先，充分利用大数据和 AI 技术带来的前所未有的机遇。

2. 区块链技术

区块链技术在金融智能领域已经超越了它最初作为加密货币基础技术的用途，成为了一种具有广泛影响力的创新工具。区块链是一种分布式账本技术，以去中心化的方式记录和验证所有交易记录，其主要特点包括数据的不可篡改性、透明性和去中心化。在金融领域，区块链技术被应用于快速、安全且成本较低的跨境支付和资金转移服务，智能合约的自动执行和控制，以及优化供应链融资和管理。

区块链与金融智能的结合增强了金融交易的信任和安全性，减少了欺诈和错误的可能性，同时促进了金融智能创新。这种结合为金融产品创新提供了新的可能性，特别是与 AI 和大数据等技术的融合。然而，这一领域也面临着监管和合规的挑战，以及技术成熟度和可扩展性的需求。未来的发展趋势包括推动去中心化金融（DeFi）的发展和与物联网、云计算等更多技术的跨界融合，为金融领域带来更多创新。总体来说，区块链技术在金融智能领域的应用不断扩展，正在改变传统金融操作的方式，并为金融行业的未来发展开辟了新的道路。

3. 云计算

云计算在金融智能领域扮演着至关重要的角色，它为金融行业提供了强大的计算能力、存储资源和高效的数据处理能力。作为一种基于互联网的计算方式，云计算允许用户通过网络访问、管理和处理共享的计算资源，其主要特点包括可扩展性、按需服务、资源共享

和成本效率。在金融智能领域，云计算不仅提供了数据存储和处理能力，更凭借其灵活性与可扩展性，为金融机构的协作模式升级和业务创新生态的构建注入了强大动力。它为金融机构提供了支持复杂金融分析和决策过程的能力，同时根据金融市场的动态变化快速调整资源规模。

云计算与金融智能的融合进一步加速了 AI 计算，并支持了数据集成和高级数据分析。云平台为训练复杂的 AI 模型提供必要的计算资源，并能集成来自不同来源的数据，支持机器学习和其他智能算法。然而，使用云计算也面临着安全性和隐私保护的挑战，特别是在金融领域。此外，合规性问题是金融行业在采用云计算时必须考虑的一个重要方面。未来发展趋势可能包括采用混合云和多云策略，以及开发专门为云环境设计的金融应用和服务。总体来说，云计算为金融智能提供了一个高效、灵活且成本效益高的平台，成为现代金融技术不可或缺的组成部分。

1.3 金融智能实践与方向

1.3.1 金融智能发展现状

1. 国内现状

国内金融智能的发展呈现出快速增长的态势，受到了金融行业以及科技界的广泛关注和投资。科技与金融的深度融合，特别是国内科技巨头如阿里巴巴、腾讯和百度等公司在金融智能领域的积极参与，以及众多金融科技创业公司的崛起，共同推动了金融服务的数字化转型。这一发展得到了政府的支持，政府通过政策引导和资金投入为金融智能的发展提供了良好的环境。同时，市场需求的增长，尤其是消费者对个性化和便捷金融服务需求的日益增加，进一步推动了对金融智能技术的需求。

金融智能技术在移动支付、网络信贷、投资交易和风险管理等领域已经有了广泛应用，提高了交易效率和用户体验。在金融投资方面，资产管理行业涌现出一批依靠模型和数据决策的量化投资基金。截至 2023 年，量化私募基金净规模 8700 亿元。量化私募依靠领先的智能算法来分析市场数据，寻找投资机会，并构建投资组合。以幻方量化为例，该公司在 2016—2021 年搭建了“萤火一号”和“萤火二号”AI 计算集群，一方面服务于内部投资策略的研发，实现了显著的超额收益；另一方面为部分高校 AI 实验室免费提供科研协助和算力支持，赋能 AI 基础科学研究。

近年来，中国金融智能市场格局趋于稳定，行业规范化和高质量发展成为共识。头部金融机构（如国有六大行）的科技投入仍保持增长态势。例如，2022 年国有六大行科技投入同比增长 8.42%，显示出金融科技在提升金融服务质量和效率方面的持续投入。智能算力和大模型技术在金融业的广泛应用，加速了数据智能技术的发展，进一步促进了数据要素价值的释放。例如，农业银行推出了自主金融 AI 大模型 ChatABC，工商银行基于昇腾 AI

发布了金融行业通用大模型，这些技术的应用显著提升了金融业的智能化水平。

2. 全球现状

全球范围内，金融智能正在经历快速发展和广泛应用，极大地推动了金融行业的创新，并改变了传统金融业务的运作方式。从硅谷到伦敦，从新加坡到北京，世界各大金融中心都在积极发展金融科技，众多创业公司和传统金融机构正通过应用 AI、区块链和大数据等金融智能技术，推动金融服务的创新和优化。金融智能的应用场景多样化，涵盖了跨境支付和国际贸易的简化、个性化投资咨询服务的提供等方面。然而，受全球经济增速放缓、经贸摩擦、美联储加息等因素影响，全球金融科技产业结束了快速扩张阶段，进入增速放缓期。根据中国信通院发布的《中国金融科技生态白皮书（2023 年）》显示：2022 年全球金融科技投融资总额为 770 亿美元，较 2021 年下降 45%。全球区域发展也呈现出不均衡的态势。北美和欧洲的金融科技投融资规模和活跃度显著下降，而拉美和非洲等新兴市场表现出较大的增长潜力。例如，2023 年二季度，拉美地区的金融科技投融资规模较一季度增长 150%，非洲地区的金融财务类应用活跃用户规模同比增长 31%。

全球金融科技市场主体不断加快对前沿技术的探索，寻求新的场景突破，同时也高度重视新技术带来的风险。全球 85 家上市金融科技公司中，仅有 45% 的公司处于盈利状态，显示出金融科技企业普遍面临盈利挑战和现金流问题。此外，随着金融智能技术的广泛应用，合规性成为一个重要的挑战，各国政府和监管机构正在加强对金融科技的监管协作，同时数据安全和隐私保护也成为重要考量因素。例如，国际货币基金组织（IMF）正在开发一个全球央行数字货币平台，旨在实现国家之间的数字货币交易，提升不同经济体的互操作性。

1.3.2 金融智能遇到的挑战

金融智能的快速发展伴随着一系列挑战，这些挑战不仅涉及技术层面，还包括数据安全、隐私保护、监管协作等多方面。首先是技术可靠性，确保金融智能系统在高风险金融环境中做出准确、可靠的决策至关重要。技术失误可能导致重大金融损失和声誉损害。同时，防止训练数据偏见在 AI 模型中产生不公平或有歧视的决策也是一大挑战，否则可能引起客户不满，甚至面临法律诉讼。技术可靠性是赢得客户信任的基础，客户信任度低可能影响新技术的采纳率和市场扩张。其次是数据安全，金融机构在处理大量敏感金融数据时必须确保数据的安全性和客户隐私。数据泄露和隐私侵犯事件可能导致信任危机和法律责任。最后是跨境协作与监管挑战。金融科技的国际协作需求日益提升，但在具体实践过程中，部分技术由于应用不当或缺少监管，放大了金融行业的风险。全球范围内，金融科技跨境协作规则和互操作性平台建设的推进面临诸多挑战，各国的监管标准和政策差异给跨境协作带来了复杂性。上述挑战要求金融机构、技术提供商和监管机构共同努力，通过技术创新、政策制定和行业协作来解决。

1.3.3 金融智能发展的方向

金融智能的未来发展方向将聚焦于已有挑战，从技术、安全和公平、监管三个角度，不断提升金融智能的发展水平。

首先，从技术角度，需要确保金融智能系统在高风险环境中做出准确、可靠的决策。增强 AI 模型的可解释性，使得决策过程透明，便于监管和客户理解。行业从业者需要进一步开发和应用更加鲁棒的算法，增加模型训练数据的多样性，减少数据偏见。实施严格的测试和验证程序，确保模型在各种市场条件下的可靠性。此外，为了解决金融场景中的实时数据处理问题，金融智能应用可以利用云计算和边缘计算，并探索量子计算等前沿技术，提升计算速度和效率，确保数据的快速处理和响应。

其次，从安全和公平角度，金融智能需要在数据安全与隐私保护、防止数据偏见等方面进一步加强。加强对金融数据的保护，确保数据在传输、存储和处理过程中的安全性和隐私保护，并防止训练数据中的偏见在 AI 模型中导致不公平或歧视性的决策。制定公平性检测和调整算法，定期审查和更新训练数据，确保其代表性和公正性。建立多样化的数据源和训练集，避免单一数据源的偏见影响。

最后，从监管角度，金融智能系统要主动适应监管要求，保持合规的透明度。增强金融智能系统的透明度和问责机制，确保其在使用过程中的合法性和合规性。金融智能系统需要建立透明的决策记录和审计机制，确保每一步决策都可追溯和审查。制定明确的问责机制，对因技术失误导致的金融损失和违规行为进行追责。此外，在跨境监管方面，未来的发展方向在于推动金融科技跨境协作，建立统一的监管标准和平台，防止跨境监管套利，进而形成系统性风险。

1.4 Python 基础知识

Python 是一种简单易学、对初学者友好、功能强大的编程语言，它有高效率的高层数据结构，可简单而有效地实现面向对象编程。这些优点使得 Python 在许多领域都是一个理想的脚本语言，特别适用于一些需要快速开发的场景。作为较受欢迎的编程语言之一，它在金融投资、机器学习、人工智能、文本分析等领域常年位居编程语言排行榜榜首。

此外，Python 是免费的开源软件，任何人都可以使用它，也可编写自己的第三方库来拓展 Python 的功能，因此在机器学习领域，Python 已经有了很多通用的第三方拓展库，尤其是 Sklearn 等，其功能丰富、应用简单，受到机器学习研究者的喜爱。此外，还有 Numpy、Pandas、Matplotlib 等第三方库也是机器学习的常用库。

本节主要涵盖 Python 安装和 Python 入门，带领读者快速地准备好可用的 Python 环境，并对 Python 在金融智能中的应用有着直观的理解和认识。

1.4.1 Python 安装

在机器学习、金融数据分析等领域的应用中，只安装 Python 并不够，还需要安装功能丰富的第三方库来搭建所需要的环境。针对机器学习，通常会使用 Numpy、Pandas、Matplotlib 等第三方库。Anaconda 实现了对 Python 及常用库的封装，对于数据分析、数据可视化以及机器学习等，使用 Anaconda 提供的封装，可以让环境的配置更加方便。

1. 安装 Anaconda

Anaconda 是一个用于科学计算的开源 Python 版本，用于计算科学（数据分析、机器学习、金融大数据处理和预测分析），支持 Linux、Mac、Windows 系统，提供了软件包管理和环境管理的功能，可以很方便地解决多版本 Python 并存、切换以及各种第三方包的安装问题。它利用 Conda 进行库（package）和环境（environment）的管理，并且已经包含了 Python 和相关的配套工具。本节会介绍 Python 的安装与使用，通常安装 Anaconda 后无须再额外安装 Python。

可以从 Anaconda 官方网站选择适合自己计算机设备的 Anaconda 版本进行下载安装，下载页面如图 1-1 所示。

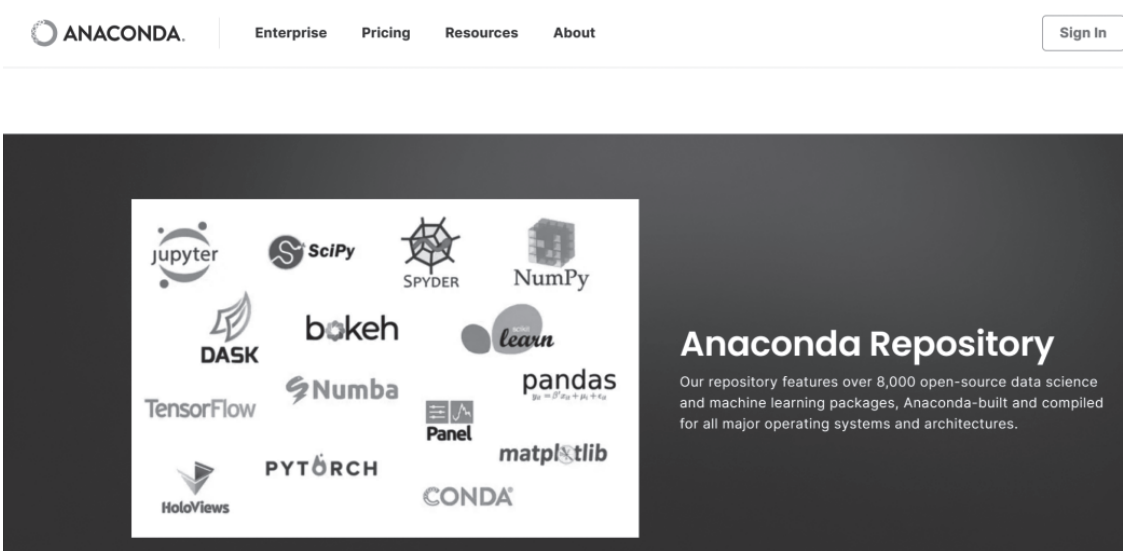


图 1-1　Anaconda 下载页面

针对下载好的 Anaconda，跟随安装向导安装即可，安装后打开 Anaconda Navigator，可以发现如图 1-2 所示的应用界面。

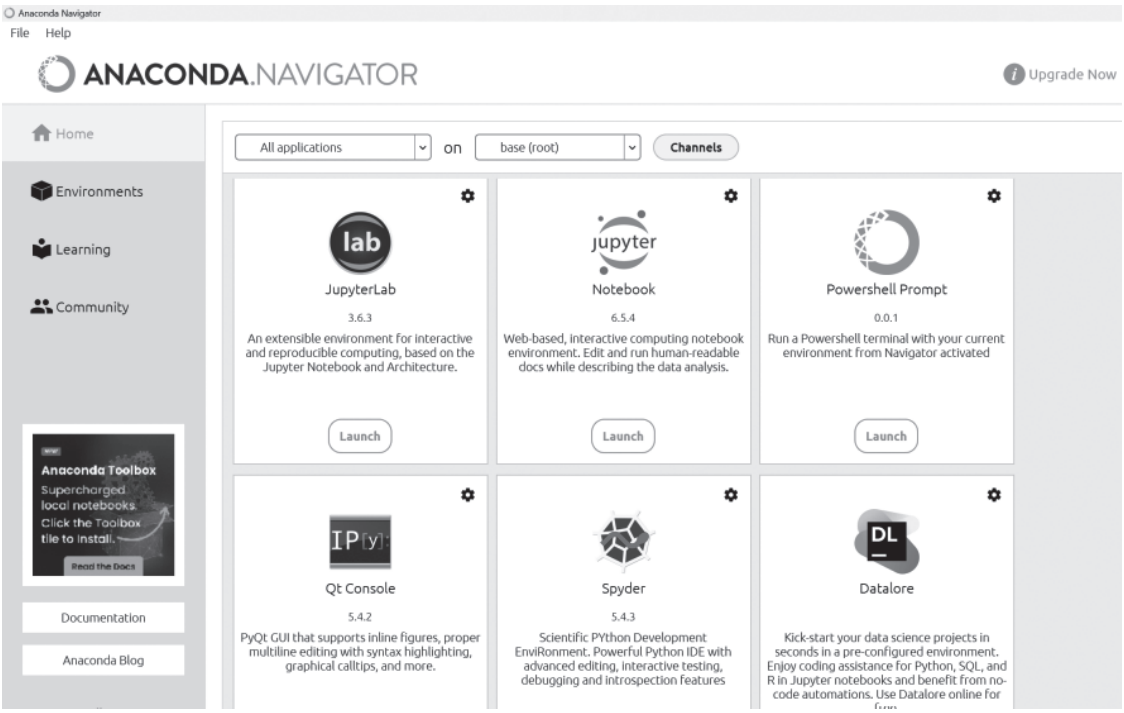


图 1-2　Anaconda 安装后的应用界面

该界面中常用来编写 Python 程序的应用有 Spyder、Jupter Notebook 和 JupyterLab 等。

2. 安装 Python 库

虽然在 Anaconda 中已经提前安装好了常用的 Python 库，但是在使用 Python 进行金融数据分析、机器学习时，还会遇到重新安装其他库的情况，下面就介绍通过 Conda 和 pip 安装 Python 库的方式。其中 Conda 是一个开源、跨平台和与语言无关的软件包管理和环境管理系统，通过 Conda 可安装、升级软件包。Conda 因 Python 而产生，但是它可以打包、分发任意语言编写的软件（例如 R 语言）和包含多语言的项目。Conda 已经包含在 Anaconda 中。pip 是一个以 Python 语言编写的软件库管理系统，它可以安装和管理 Python 库。

通过 Conda 命令安装，通常使用如下命令：

```
Conda install 库的名称
```

通过 pip 命令安装，通常使用如下命令：

```
pip install 库的名称
```

1.4.2　Python 常用数据类型、条件、循环与函数

本节将会介绍 Python 中的基础知识，帮助读者快速入门 Python，主要介绍如何使用 Python 中的列表、元组、字典与集合等数据结构，Python 中的条件判断、循环语句以及函数等内容。

1. 列表

列表（list）是 Python 中最基本的数据类型之一，是一种有序的集合，列表中的每个元素都会有一个数字作为它的索引，第一个索引是 0，第二个索引是 1，以此类推。列表可以通过索引获取列表中的元素。

Python 生成一个列表可以通过 list()函数或者中括号 [] 来完成。例如：生成包含 5 个元素的列表 A 的程序如下所示，同时列表的长度可以使用 len()函数进行计算，生成的列表 A 长度为 5。

```
...
Ln[1]: ## 生成一个列表
   A = [1,2,3,4,5]
   A
Out[1]: [1,2,3,4,5]
Ln[2]: ## 计算列表中元素的数量
   len (A)
Out[2]: 5
...
```

生成一个列表后，可以通过索引获取列表中的元素，从前往后的索引是从 0 开始的，而从后往前的索引是从 –1 开始。此外，获取列表中的一个范围内的元素，也可以通过切片索引来完成。例如，使用切片“0:2”，表示要获取索引从 0 开始，到达索引为 2 的元素结束，并且不包含索引位置为 2 的元素。例如下面的程序：

```
...
Ln[3]: ## 从前往后时，索引从 0 开始
   A[2]
Out[3]: 3
Ln[4]: ## 从后往前时，索引从 -1 开始
   A[-2]
Out[4]: 4
Ln[5]: ## 获取列表中的一段
   print (A[0:2]) # 输出的结果不包含索引 2 所表示的元素
   print (A[1:-1]) # 输出的结果不包含索引 -1 所表示的元素
Out[5]: [1,2]
[2,3,4]
...
```

针对一个已经生成的列表，可以通过 append()函数在其后面添加新的元素，并且元素的数据形式可以多种多样，数字、字符串，甚至新的列表也可以。此外，在指定位置插入新的内容也可以使用 insert()函数，该函数的第一个参数为内容插入的位置，第二个参数为要插入的内容。删除列表中的末尾元素可以通过列表的 pop()函数，该函数会每次删除列表中的最后一个元素。此外，还可以通过 del 删除列表中指定位置的元素。

2. 元组

元组（tuple）和列表非常类似，也是 Python 中最常使用的序列，但是元组一旦初始化

就不能修改。建立元组可以使用小括号 ()或者 tuple()函数。在使用小括号时，只有 1 个元素的元组在定义时必须在第一个元素后面加一个逗号。针对生成的元组同时可以使用 len()函数进行计算。表示元组的程序如下。

```
...
Ln[6]## 初始化一个元组
    B = (1,2,3,4,5)
    print (B)
    ## 输出元组中元素的个数
    print (len (B))
Out[6]: (1,2,3,4,5)
5
...
```

和列表一样，针对元组中的元素，同样可以使用索引获取元素，也可以通过加号“+”将多个元组进行拼接。

3. 字典

字典也是 Python 最重要的数据类型之一，其中字典的每个元素的键值（key: value）对用冒号“:”分割，键值对之间用逗号“,”分割，整个字典包括在大括号 {} 中，计算字典中键值对的数量可以使用 len()函数。例如：初始化字典 A 可以使用下面的方式。

```
...
Ln[7] ## 初始化一个字典
    A = {“A”:0,“B”:1,“C”:2,“D”:3}
    print (“A”,A)
## 计算字典中元素的数量
    print (len (A))
Out[7]: A: {‘A’:0,‘B’:1,‘C’:2,‘D’:3}
4
...
```

字典 A 中，可以通过 keys()函数查看字典的键，通过 values()函数查看字典的值，并且可以通过字典的键获取对应的值。字典的 pop()函数可以利用字典中的键，删除对应的键值对。

4. 集合

集合（set）是一个无序的不重复元素序列。可以使用大括号 {} 或者 set()函数创建集合。注意创建一个空集合必须用 set()函数而不是 {}，因为 {} 是用来创建一个空字典的。程序如下。

```
...
Ln[8] ## 创建一个集合
    A = {“A”,“B”,“C”,1,2,3}
    print (A)
## 集合元素的数量
print (len (A))
```

```
Out[8]: {'A','B','C',1,2,3}
    6
    ...
```

集合之间也可以相互运算，例如：集合的差集可以使用“-”或者 difference()函数；集合的并集可以使用“|”或者 union()函数；集合的交集可以使用“&”或者 intersection()函数；集合的并集减去交集可以使用“^”或者 symmetric_difference()函数。

5. 字符串

字符串也是 Python 中最常用的数据类型。可以使用引号来创建字符串。字符串的基础使用方式和列表很相似，例如：可以通过索引进行字符串内容的提取，通过 len()函数计算字符串的长度，通过“+”号拼接字符串。程序如下。

```
...
Ln[9]: ## 创建一个字符串变量 A
    A =“人工智能在金融领域应用”
    print (A)
Out[9]: 人工智能在金融领域应用
    11
    ...
```

除了上述的字符串基本操作之外，还可以通过 find()函数查找字符串中的子串；通过 join()函数拼接字符串；通过 split()函数拆分字符串；通过 replace()函数将指定内容进行替换。

6. 条件判断语句

条件判断语句是通过一条或者多条语句的执行结果是否为真（True 或者 False）来决定执行的代码块，是 Python 中的基础内容之一。常用的判断语句是 if 语句。

针对 if else 语句，其常用的结构为

```
If 判断条件：
    执行语句 1…
else:
    执行语句 2…
```

即如果满足判断条件，则执行语句 1，否则执行语句 2。程序如下。

```
...
Ln[10]: ## if else 语句
  A = 21
  If A % 2 ==0:
     print (“A 是偶数”)
  else:
     print (“A 是奇数”)
Out[10]: A 是奇数
...
```

7. 循环语句

循环语句也是 Python 最常用的语法之一，其中 for 循环是要重复执行语句，while 循环则是在给定的判断条件为真时执行循环，否则退出循环。例如使用 for 循环计算 0 ～ 100 的累加和，可以使用下面的程序，在程序中会依次从 0 ～ 100 中取出一个数进行相加。

```
...
# 初始化累加和变量
sum = 0
# 使用 for 循环遍历 0 到 100（包括 100）
for i in range（101）:
    sum += i
# 打印累加和
print（"0 到 100 的累加和是 :", sum）
...
```

8. 函数

函数是已经组织好的、可重复使用的、实现单一功能的代码段。函数能提高应用程序的模块性，增强代码的重复利用率。Python 提供了许多内建函数，例如 print()、len()等。

Python 也可以定义自己新的函数，其中定义函数的程序结构如下：

```
...
def functionname（parameters）:
function_suite # 函数的内容
return expression # 函数的输出
...
```

其中，functionname 表示函数的名称，parameters 可以指定函数需要传入的参数。

Python 中的 lambda 函数也叫匿名函数，即没有具体名称的函数，它可以快速定义单行函数，完成一些简单的计算功能。lambda 函数（表达式）中，冒号左边是参数，可以有多个，需要用逗号分隔；冒号右边是函数的计算主体，会返回其计算结果。可以使用下面的方式定义 lambda 函数。

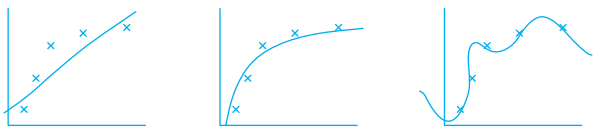
```
...
Ln[]## lambda 函数，一个参数
f = lambda x: x**2 + 1
f（2）
Out[]: 5
Ln[]: ##lambda 函数，多个参数
f = lambda x,y,z:（x+y）*z+x*y*z
f（1,2,3）
Out: 15
...
```

1.5 机器学习模型评估

1.5.1 过拟合与欠拟合

在构建机器学习模型时，核心目标是从有限的训练数据中揭示数据特性与预测目标之间的真实联系，从而使模型能在未知的测试数据上展现良好的预测能力。实现这一目标的关键是使模型掌握样本数据中的普遍规律，而非仅仅记忆训练集中的具体细节。若模型在学习过程中过度适应训练集，记住过多细节并展现出色，可能会导致模型在新数据集上预测性能不佳。

图 1-3 展示了欠拟合、好的拟合和过拟合的情形。设我们的目标是找到 x 与 y 之间的最佳函数关系。在图 1-3（a）中，简单线性关系被用以描述 x 与 y 的联系，但显然这种关系过于简化，导致欠拟合现象。图 1-3（b）使用二次抛物线关系来描述这两者之间的联系，虽不完美，但较好地描绘了 x 与 y 之间的关系。图 1-3（c）中，使用高次函数曲线关系描述 x 与 y 的关系，虽然看似完美，但这种过于复杂的模型可能导致对新样本的预测失效。这与奥卡姆剃刀（Occam's Razor）定律的观点相呼应，即在不必要的情况下不增加额外的复杂度，遵循“简约有效”原则。



（a）欠拟合　（b）好的拟合　（c）过拟合

图 1-3　回归问题中三种拟合状态

过拟合问题发生时，模型会错误地将训练数据中的噪声视为重要信息，导致在非训练数据上的预测性能下降。这种情况下，模型在训练集上的表现可能很好，但在测试集上表现不佳。过拟合问题通常伴随着选择更灵活或复杂的模型时出现，如决策树和神经网络等。相比之下，欠拟合是指模型对训练数据和新数据都无法进行有效拟合的情况。

解决过拟合问题的常用方法如下。

（1）增加更多数据：更多的数据可以帮助模型更好地逼近真实数据分布，改善预测结果。

（2）降低模型复杂度：通过正则化、剪枝等技术降低模型复杂度。

（3）集成算法：通过结合多个模型来提高整体预测性能。

（4）主动丢失信息：如在神经网络中使用丢弃（dropout）技术，或在随机森林中随机选择特征。

（5）使用交叉验证：确保模型在不同数据子集上的有效性和泛化能力。

相对于过拟合，欠拟合问题更容易解决，通常只需增加模型复杂度。但如果在使用复杂模型时仍出现欠拟合问题，可能需要检查数据的特征工程是否合理，或考虑引入新的、具有解释能力的特征变量。

1.5.2 偏差和方差的权衡

为了更深入地理解过拟合问题，可以通过对训练集误差进行偏差（bias）和方差（variance）的分解来分析。假设存在一个真实的函数$f(.)$描述了y与x之间的关系，即$y=f(x)$。在机器学习中，目的是根据现有的训练集数据$\mathcal{T}$，找到一个最优的模型$\hat{f}_{\mathcal{T}}(.)$，使其尽可能接近真实的函数$f(.)$。通常，这个模型$\hat{f}_{\mathcal{T}}(.)$与真实函数$f(.)$之间存在偏离，这种偏离可以分解为三部分：偏差、方差和不可避免的误差。

（1）偏差：由于模型选择引起的误差。例如，如果选择使用线性模型来拟合数据，那么就不可能获得最优模型，因为x和y之间的真正关系并非线性。

（2）方差：由于随机选择训练集数据引起的偏差。不同的训练集可能会导致学习到的模型有所不同，这种变化反映了模型的方差。

（3）不可避免的误差（inherent error, 内在误差）：由于训练集与总体样本之间存在的差异造成的。在机器学习中，不可能获取所有样本，因此基于有偏的训练集训练出来的模型，即使是最优模型，也无法完全等同于真实的函数。

简单来说，偏差通常与模型的复杂度负相关，而方差与模型的复杂度正相关。选择一个过于简单的模型可能导致高偏差（欠拟合），而选择一个过于复杂的模型可能导致高方差（过拟合）。理想的模型是在偏差和方差之间找到一个平衡点，以实现最佳的泛化性能。为了更严谨地理解这三种误差，可以从数学的角度对上述问题进行定义和证明。

1. 模型训练误差

在回归问题中，一般使用均方误差（Mean Squared Error，MSE）来定义模型的好坏。它定义了训练好的模型$\hat{f}_{\mathcal{T}}(.)$预测结果与真实函数$f(.)$之间的欧氏距离。即训练集的MSE由以下公式决定：

$$\mathrm{MSE}(x)=\mathbb{E}_{\mathcal{T}}\left[\left(\hat{f}_{\mathcal{T}}(x)-f(x)\right)^2\right] \tag{1.1}$$

式中，$\mathbb{E}_{\mathcal{T}}$表示来自不同训练集的平均值（期望值）。定义不同训练集预测结果的平均值为$\mu(x)$，即$\mu(x)=\mathbb{E}_{\mathcal{T}}\left[\hat{f}_{\mathcal{T}}(x)\right]$。

（1）偏差的定义。

偏差是人为选取模型带来的误差，它描述了训练集拟合出来的模型的预测结果的平均值与样本真实结果平均值的距离，数学上等于使用模型在训练集上预测结果的平均值减去真实样本的平均值：

$$\text{Bias}(x)=\mathbb{E}_T\left[\hat{f}_T(x)-f(x)\right]=\mu(x)-f(x) \tag{1.2}$$

（2）方差的定义。

方差是随机不同的训练集带来的偏离误差，它可以用统计学中方差的概念进行度量，即方差等于平均训练集预测结果与训练集预测结果的平均值之差的平方，用公式可以表示为

$$\begin{aligned}\text{Var}(x)&=\mathbb{E}_T\left[\left(\hat{f}_T(x)-\mu(x)\right)^2\right]\\&=\mathbb{E}_T\left[\left(\hat{f}_T(x)^2-2\hat{f}_T(x)\mu(x)+\mu(x)^2\right)\right]\\&=\mathbb{E}_T\left[\hat{f}_T(x)^2\right]-\mathbb{E}_T\left[2\hat{f}_T(x)\mu(x)\right]+\mathbb{E}_T\left[\mu(x)^2\right]\\&=\mathbb{E}_T\left[\hat{f}_T(x)^2\right]-\mu(x)^2\end{aligned} \tag{1.3}$$

（3）训练误差的拆解证明。

定义好偏差和方差之后，可以很容易地将训练误差进行拆解，证明过程如下：

$$\begin{aligned}\text{MSE}(x)&=\mathbb{E}_T\left[\left(\hat{f}_T(x)-f(x)\right)^2\right]\\&=\mathbb{E}_T\left[\left(\hat{f}_T(x)^2-2\hat{f}_T(x)f(x)+f(x)^2\right)\right]\\&=\mathbb{E}_T\left[\hat{f}_T(x)^2\right]-2\mu(x)f(x)+f(x)^2\\&=\mathbb{E}_T\left[\hat{f}_T(x)^2\right]-\mu(x)^2+\mu(x)^2-2\mu(x)f(x)+f(x)^2\\&=\mathbb{E}_T\left[\hat{f}_T(x)^2\right]-\mu(x)^2+\left(\mu(x)-f(x)\right)^2\\&=\text{Var}(x)+\text{Bias}(x)^2\end{aligned} \tag{1.4}$$

2. 偏差与方差权衡说明

从上述分析可知，在机器学习中，任何训练模型的误差都可以被拆分为偏差和方差。我们将使用射击的比喻来说明训练模型的目标（参见图 1-4）。

高偏差和高方差（图 1-4（d））：这种情况类似于一个射击能力很差的运动员，既无法打中靶心，又没有稳定性。在机器学习中，这对应于一个在各种样本上都表现差劲，且结果差异很大的模型。这通常意味着模型与数据集不匹配，可能需要更换模型或加强特征工程。

高方差和低偏差（图 1-4（c））：这种情况相当于射击能力尚可的运动员，偶尔能打中靶心，但通常偏差很大。在机器学习中，这对应于一个在某些样本上表现良好，但在其他样本上表现糟糕的模型。这通常是过拟合的表现，可能因为模型过于复杂。

低方差和高偏差（图 1-4（b））：这像是射击能力一般但成绩稳定的运动员，虽然不能打中靶心，却很一致。在机器学习中，这对应于一个在所有样本上表现一般，但预测结果

相对稳定的模型。这是欠拟合的特征，通常是因为模型过于简单。

低偏差和低方差（图 1-4（a））：这是理想的状态，类似于射击技巧高超的运动员，不仅每次都能打中靶心，而且非常稳定。在机器学习中，这是我们追求的目标：一个在各种样本上都表现出色且稳定的模型。

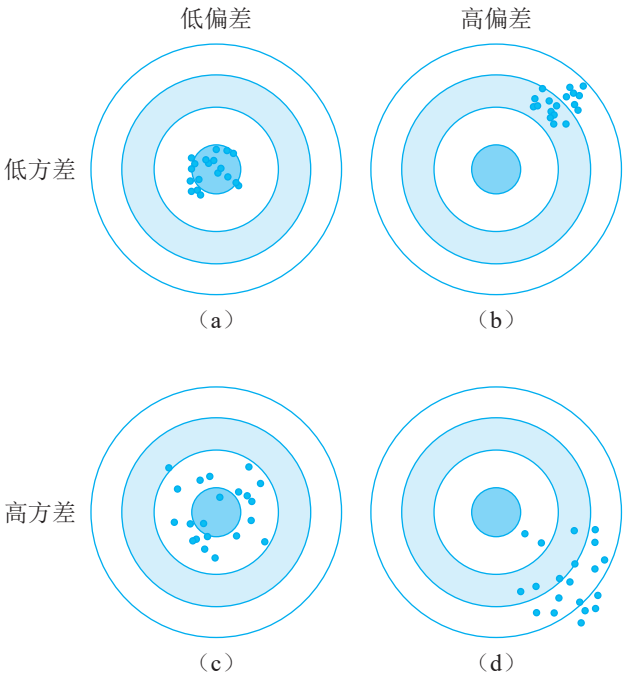


图 1-4　训练模型的目标

总体来说，机器学习的目标是找到一个既能准确预测（低偏差）又具有泛化能力（低方差）的模型。这就像找到一个既能准确瞄准靶心又能保持一致性的射击运动员一样。

3. 模型复杂度与方差和偏差

模型的复杂度对其误差也有重要影响，图 1-5 展示了模型复杂度与训练模型的总误差、偏差和方差之间的关系。

总误差：在使用的模型越复杂时，总误差通常会先下降然后上升。这是因为初始的复杂度增加有助于更好地拟合数据，但超过某个点后，复杂度的增加会导致模型对训练数据的过度拟合，从而增加在测试数据上的误差。

偏差：随着模型复杂度的增加，偏差会持续下降。简单模型（如线性模型）可能无法捕获数据的所有真实特征，从而导致较高的偏差。随着模型变得更复杂（如灵活的神经网络），它们能更好地逼近真实数据结构，因此偏差会降低。

方差：与偏差相反，随着模型复杂度的增加，方差会上升。复杂模型更容易对训练数据中的随机波动（噪声）做出反应，从而导致在不同的数据集上预测结果出现较大波动，即方差增大。

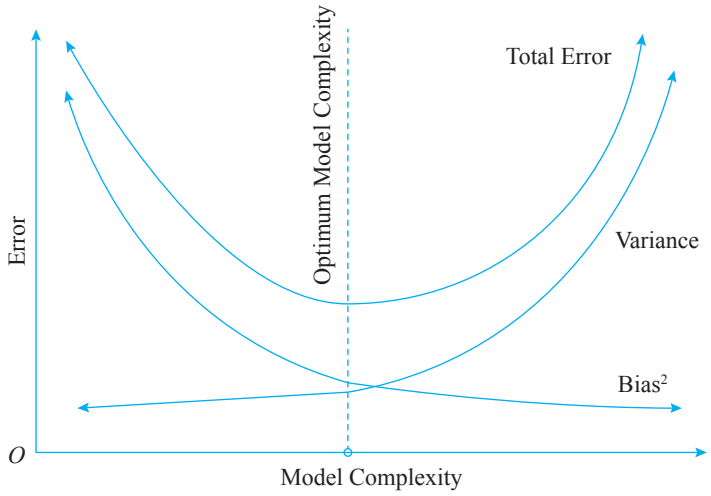


图 1-5　模型复杂度与总误差、偏差和方差的关系

最优模型的选择并非仅仅追求最小的偏差或最小的方差，而是两者的最佳平衡点。在这个平衡点上，模型的总误差最小，模型能够有效地泛化到新数据上。这就意味着，理想的模型既不会过于简单（以避免高偏差），也不会过于复杂（以避免高方差）。这种平衡在机器学习中被称为“偏差 - 方差权衡”（bias-variance tradeoff）。

1.5.3　回归问题机器学习模型的评价指标

我们已详细讨论了基于模型训练误差选择模型的方法。然而，在模型选择过程中，除了考虑训练误差，我们还常使用其他多种指标来评估模型的性能。在处理回归问题时，我们通常关注连续变量的预测。评价回归模型性能的常用指标包括均方误差（MSE）、平均绝对误差（MAE）和决定系数。假设回归问题中的实际标签值为 y，模型的预测值为 $\hat{y}$，样本数量为 n，标签真实值的平均值为 $\bar{y}$。以下是不同回归模型评价指标的详细说明。

1. MSE

MSE 是评估回归模型中最常用的指标，它通过平方项来量化预测值与实际值之间的差异，也被称为 L2 损失函数。其计算公式如下：

$$\mathrm{MSE}(y,\hat{y})=\frac{1}{n}\sum_{i=0}^{n-1}(y_i-\hat{y}_i)^2 \tag{1.5}$$

2. MAE

MAE 通过绝对值来量化预测值与实际值之间的差异，这也是 L1 损失函数的一种形式。其计算公式如下：

$$\mathrm{MAE}(y,\hat{y})=\frac{1}{n}\sum_{i=0}^{n-1}\left|y_i-\hat{y}_i\right| \tag{1.6}$$

3. 决定系数

决定系数（R^2）表示模型解释实际预测标签方差的比例。其计算公式见式（1.7）。当 R^2 值为 1 时，表示预测值与实际值完全一致。值得注意的是，R^2 值也可能小于 0，这表明模型的预测误差（分子）相比预测值与样本平均值之间的差异（分母）更大。当预测模型结果极差，甚至不如样本均值时，此指标可能呈现负值。

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{1.7}$$

4. Python 实现

可以直接调用 Sklearn 库中的以下函数来实现以上评价指标的计算。

```
...python
sklearn.metrics.mean_squared_error (y_true, y_pred)
sklearn.metrics.mean_absolute_error (y_true, y_pred)
sklearn.metrics.r2_score (y_true, y_pred)
...
```

参数解析如下。

y_true：数据集标签的真实值；

y_pred：数据集标签的预测值。

1.5.4 机器学习的超参数调校

在机器学习模型构建过程中，通常将现有数据样本分为三部分（图 1-6）：训练集、验证集和测试集。训练集用于初步拟合数据和构建模型；验证集相当于一个模拟测试集，主要用于调节模型的超参数（hyperparameters）；测试集用于最终评价各模型在未见样本上的表现。

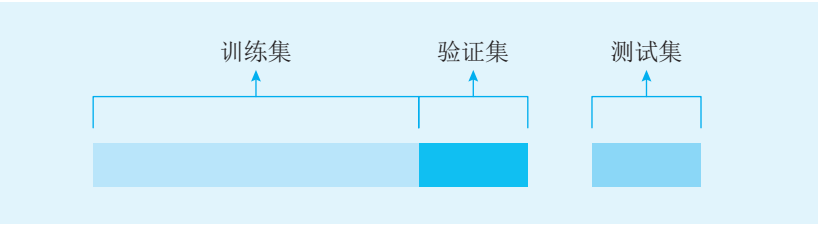


图 1-6　数据样本划分

想象你是一位音乐学校的钢琴老师，目前你需要在所有学生中挑选一位代表学校参加全国钢琴大赛，争取荣誉。你有过去 20 年全国钢琴大赛的曲目和评审标准。首先，你会让所有学生学习这 20 年中的前 10 年部分的比赛曲目，然后观察学生们对这些曲目的掌握程度。你会特别关注那些课堂表现出色，对这 10 年曲目反应敏捷的学生。接着，在这些学生

中进行额外的 5 次测试，使用之前没教过的另外 10 年的曲目。你不仅会评估学生对新曲目的掌握情况，还会对这些曲目进行讲解，以促进学生的进步，同时也可以筛选出那些只会死记硬背，不能灵活运用知识的学生。最后，基于最后 10 年的曲目，举办 5 场模拟比赛，从中选出在这些模拟比赛中平均成绩最好的学生，代表学校参加全国钢琴大赛。

这个过程与挑选正确的机器学习模型非常相似。为了增强模型对新样本的表现，提高其泛化能力，降低泛化误差，可以将所有数据分为三部分：1/2 作为训练集用于训练和拟合模型（类似前 10 年的钢琴曲目）；1/4 作为验证集用于调整超参数（类似额外的 5 次测试）；1/4 作为测试集，用于比较不同模型的泛化误差（类似基于最后 10 年曲目的模拟比赛）。通常，训练集、验证集和测试集应互不相交，即每个集合的样本都是独立的。这就像假设全国钢琴大赛不会重复以前的曲目一样，希望学生们在模拟比赛中也不会遇到之前教学中使用的曲目。

根据样本特点的不同，验证集的设置方法一般有三种：固定交叉验证、*k* 折交叉验证、时间序列交叉验证等。

1. 固定交叉验证

固定交叉验证（holdout cross-validation）是一种在机器学习中常用的验证方法。这种方法涉及将数据集分为两部分：训练集和验证集。训练集用于模型的训练，而验证集用于评估模型性能。这种方法的一个关键特点是验证集在整个训练过程中保持不变。

在固定交叉验证中，重要的是要确保训练集和验证集的样本代表性，避免出现样本分布不均匀的情况。例如，如果训练集包含的都是容易预测的样本，而验证集包含的都是难以预测的样本，这可能会导致对模型性能的不准确评估。

Python 代码如下。

```
...
python
sklearn.model_selection.train_test_split (*arrays, test_size=None, train_
size=None, random_state=None, shuffle=True, stratify=None)
...
```

在实现固定交叉验证时，可以使用 Python 的 sklearn 库中的 train_test_split() 函数。这个函数允许用户指定训练集和测试集的比例，并提供参数来控制数据切分的方式。

参数解析如下。

arrays: 待切割的数据集，可以是列表、NumPy 数组、Pandas 数据框等格式。

train_size: 训练集的比例或样本数量。

test_size: 测试集的比例或样本数量。

random_state: 随机数种子，用于确保实验的可重复性。

shuffle: 是否对数据进行随机打乱。

stratify: 是否按类别进行分层抽样。该参数在处理不平衡数据集时特别有用，可以确保

训练集和测试集在类别分布上保持一致。

使用 train_test_split() 函数可以有效地进行数据切分，但需要注意的是，由于这种函数只进行一次划分，其结果可能具有一定的偶然性。因此，在某些情况下，可能需要考虑使用其他方法，如 *k* 折交叉验证（*k*-fold cross-validation），来获取更稳健的模型性能评估。

2. *k* 折交叉验证

k 折交叉验证（图 1-7）是一种在机器学习中广泛使用的模型验证方法，用于评估模型的泛化能力。这种方法特别有效，因为它能够克服单次划分（如固定交叉验证）可能带来的样本选择偏差。

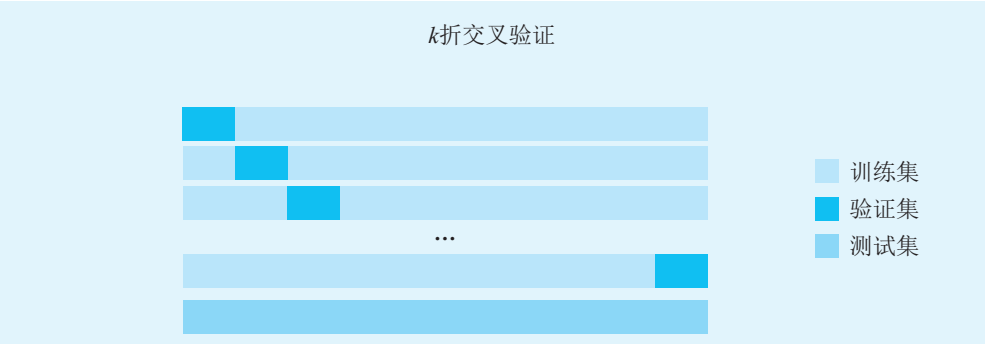


图 1-7　*k* 折交叉验证

以 5 折交叉验证为例，整个过程包括以下步骤。

（1）数据划分：将整个训练数据集等分为 5 个互斥的子集（即“折”）。

（2）循环训练与测试：对于每一组超参数设置，进行 5 次训练和测试。在每一次中，选择其中 1 折作为测试集，剩余的 4 折作为训练集。这个过程会循环 5 次，每次都选择不同的 1 折作为测试集。

（3）评估平均得分：完成所有 5 次训练和测试后，计算这 5 次测试的结果（通常是模型预测的得分或误差）的平均值。这个平均值被用来评估模型的性能。

k 折交叉验证的优势在于它允许每个样本点都有机会作为测试集的一部分，从而确保了所有样本在模型训练和评估中的使用。这种方法减少了对特定训练集和测试集划分的依赖，提高了评估结果的稳定性和可靠性。

此外，通过改变 *k* 的值，可以在训练集大小和验证次数之间找到平衡。较小的 *k* 值意味着更大的训练集和较少的验证循环，可能导致评估结果的方差增大；而较大的 *k* 值意味着更小的训练集和更多的验证循环，可能导致模型训练时间增长。常见的选择是 *k*=5 或 *k*=10，但最佳选择取决于具体的数据集和问题。

Python 代码如下。

```
...
python
sklearn.model_selection.cross_val_score (estimator, X, y=None, *,
```

```
groups=None, scoring=None, cv=None, n_jobs=None, verbose=0, fit_params=None,
pre_dispatch='2*n_jobs', error_score=nan)
    ...
```

在 Python 的 Sklearn 库中，cross_val_score() 函数是实现 k 折交叉验证的核心工具。它主要用于评估给定机器学习模型的性能。以下是该函数的一些核心参数及其解释。

estimator: 用于训练的机器学习模型。可以是任何符合 Sklearn 规范的模型，比如线性回归、决策树、支持向量机等。

X: 特征数据集。这是模型用来进行预测的数据，通常是一个二维数组或类似结构的数据。

y: 目标变量或预测标签。这是模型尝试预测的变量，通常是一个一维数组。

scoring: 评估模型性能的指标。例如，可以使用 MSE、R^2 等。这个参数定义了如何衡量模型的好坏。

cv: 指定进行 k 折交叉验证的次数。例如，cv=5 表示采用 5 折交叉验证。

n_jobs: 指定并行运行的作业数。例如，n_jobs=–1 表示会使用所有可用的 CPU 核心。

verbose: 控制输出的详细程度。verbose=0 意味着不输出任何训练过程的信息，值越高表示输出的信息越详细。

fit_params: 传递给估计器的其他参数。是在训练模型时使用的额外参数。

pre_dispatch: 控制并行执行的任务数量。在 n_jobs 参数大于 1 的情况下，这个参数可以防止因大量并行任务而导致的内存溢出。例如，pre_dispatch='2*n_jobs' 表示最多允许有 2n_jobs 个任务同时运行。

error_score: 如果模型拟合过程中发生错误，该参数指定返回的错误分数。默认情况下，它设置为 NaN。

通过调整这些参数，可以灵活地使用 cross_val_score() 函数来评估不同模型的性能，从而找到最适合特定数据集的模型。这对于确保模型具有良好的泛化能力至关重要。

3. 超参数的选取：网格搜索与随机搜索

超参数的选择对于机器学习模型的性能至关重要，有时甚至比选择正确的算法更加关键。超参数调整应考虑模型原理、数据量和模型训练程度等因素。在机器学习中，超参数调整通常采用两种主要方法：网格搜索（Grid Search）和随机搜索（Random Search）。

1）网格搜索

（1）网格搜索通过列举所有可能的超参数组合来形成一个“网格”。

（2）它对这个网格中的每一组超参数进行测试，通常结合交叉验证来评估每组超参数的性能。这种方法虽然全面，但也可能非常耗时，尤其是当超参数的数量较多或者每个超参数的可能取值较多时。

（3）最终，从测试中选择表现最优的超参数组合。

2）随机搜索

（1）随机搜索不是测试超参数空间中的所有可能组合，而是从设定的超参数范围中随机选择组合。

（2）这种方法通常比网格搜索更快，尤其是在超参数空间很大的情况下。

（3）它通过随机采样超参数组合，可以在更大的范围内探索可能的超参数值。

（4）虽然随机搜索可能错过某些优秀的超参数组合，但它通常能在合理的时间内找到一个不错的解决方案。

网格搜索因其全面性而在小型到中型数据集上非常有效，而随机搜索在处理大型数据集或者超参数空间非常广泛时更为高效。实际应用中，可以根据数据集的大小和复杂度、可用的计算资源以及对模型性能的要求来选择合适的方法。在某些情况下，甚至可以结合这两种方法，先用随机搜索快速缩小搜索范围，然后用网格搜索细化搜索。

网格搜索 Python 代码如下。

```
...python
sklearn.model_selection.GridSearchCV (estimator, param_grid, *, scoring=None, n_jobs=None, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', error_score=nan, return_train_score=False
...
```

在机器学习中，GridSearchCV() 函数是一个强大的工具，用于自动化地搜索最佳的超参数组合。以下是该函数的核心参数及其解释。

estimator: 想要优化的机器学习模型。例如，可以是一个线性回归模型、随机森林模型等。

param_grid: 一个字典或字典列表，其中包含了想要尝试的超参数及其可能的取值。GridSearchCV() 函数会尝试所有这些超参数组合，以找到最佳组合。

scoring: 评估模型性能的指标。例如，可以使用 MSE、R^2 等来评估回归模型，或者用准确率（accuracy）、AUC 等来评估分类模型。

n_jobs: 指定并行运行的作业数。如果设置为 –1，则表示使用所有可用的 CPU 核心。

refit: 通常设置为 True，表示一旦找到最佳的超参数组合，使用这些参数重新拟合整个数据集。

cv: 指定进行 k 折交叉验证的次数。例如，cv=5 意味着使用 5 折交叉验证。

verbose: 控制输出的详细程度。当设置为 0 时，表示不会输出任何训练过程的信息。

pre_dispatch: 控制并行执行的任务数量。有助于避免在计算资源有限的情况下发生内存溢出。

return_train_score: 如果设置为 True，将返回每一组参数在训练集上的评分。这有助于识别模型是否出现了过拟合或欠拟合。

GridSearchCV() 函数尤其适用于超参数数量较少的情况，因为随着超参数数量的增加，需要尝试的组合数量会呈指数级增长。在超参数数量较多或其中一些超参数取值为连续变量时，RandomizedSearchCV() 函数通常是一个更好的选择。RandomizedSearchCV() 函数通过在超参数的分布范围内随机选择参数组合，而不是尝试每个可能的组合，这样可以在更短的时间内探索更广泛的参数空间。然而，随机性意味着可能无法找到绝对最佳的参数组合，特别是当尝试的次数较少时。

随机搜索 Python 代码如下。

```
...
sklearn.model_selection.RandomizedSearchCV (estimator, param_
distributions, *, n_iter=10, scoring=None, n_jobs=None, refit=True, cv=None,
verbose=0, pre_dispatch='2*n_jobs', random_state=None, error_score=nan,
return_train_score=False)
...
```

RandomizedSearchCV() 函数是 Sklearn 库中另一个用于超参数调优的工具，它通过在指定的参数空间内随机采样来寻找最佳的超参数组合。以下是该函数的核心参数及其解释。

estimator: 用于训练的机器学习模型，例如线性回归、支持向量机等。

param_distributions: 定义超参数取值范围的字典。与 GridSearchCV() 函数不同，这里不是列出所有可能的组合，而是定义每个超参数的概率分布。

n_iter: 随机搜索的迭代次数，即随机抽取超参数组合的次数。

scoring: 评估模型性能的指标。如 MSE、R^2 等。

cv: 指定进行 k 折交叉验证的次数。例如 5 折或 10 折交叉验证。

n_jobs: 指定并行运行的作业数。–1 表示使用所有可用的 CPU 核心。

refit: 默认为 True。表示找到最佳超参数后，用这些参数重新训练整个数据集。

verbose: 控制输出的详细程度。设置为 0 表示不输出任何训练过程信息。

pre_dispatch: 控制并行执行的任务数量。用于防止内存溢出。

return_train_score: 是否返回训练分数。有助于判断模型是否出现过拟合或欠拟合。

random_state: 随机数种子。用于保证实验可重复。

RandomizedSearchCV() 函数的主要优势在于它的高效性，特别是当超参数空间很大时。通过随机采样，它能够快速探索广泛的参数空间，而不需要像 GridSearchCV() 函数那样尝试每一种可能的组合。这使得 RandomizedSearchCV() 函数在有限的时间内更有可能找到一个不错的解决方案。然而，由于其随机性，可能需要较多的迭代次数才能接近最佳参数组合，尤其是在超参数空间极为庞大的情况下。

1.6 本章小结

人工智能技术被广泛应用于投资交易、风险管理、信用评估、互联网保险等金融场景中，提升了金融行业的效率，也引发了一系列挑战，包括模型失效、算法歧视等问题。金融智能的未来发展，需要市场需求和技术演进的共同推动。

关键名词

人工智能、深度学习、机器学习、金融智能

复习思考题

（1）随着人工智能进入大语言模型时代，金融智能将面临哪些变化和挑战。

（2）搜集国内金融智能领域的头部公司，并思考人工智能在它们的商业模式中扮演了什么样的角色。