

**机器学习项目实践报告**

**项目题目： 机器学习算法在网络入侵检测中的效果比较与分析**

**班 级： 计算机2202班**

**学 号： 20225797**

**姓 名： 张扬勇**

**指导老师： 陈喆**

**设计时间： 2025年7月8日**

**摘要**

本研究旨在探究机器学习算法在网络入侵检测中的效果比较与分析，通过比较不同机器学习算法在网络入侵检测任务上的性能表现，探讨其适用性和优劣势，为网络安全领域提供有效的技术支持。研究采用了包括神经网络、决策树、支持向量机、随机森林等多种常见机器学习算法，并针对网络入侵检测任务进行了模型训练和性能评估。在研究对象方面，项目使用了来自美国国家标准与技术研究院（National Institute of Standards and Technology）网络安全实验室（National Security Laboratory）的数据集NSL-KDD为研究对象，选取数据集中的200000条数据作为模型的训练数据对模型进行训练，20000条数据作为模型的测试数据对模型进行性能检验，并完成相应预测。

本项目采用了多种特征工程方法1、对数据进行清洗和填充缺失值处理，保证了数据的完整性和稳定性2、利用了时间特征和统计特征进行了数据增强，包括对长短期时间数据的处理、对连续特征的分桶以及统计等操作3、根据相关性生成了新的二元和三元交叉特征，进一步丰富了特征空间。

在模型选择方面，本项目选取了基于线性回归、逻辑回归、支持向量机、神经网络的方面理论的算法分别作为预测模型，并在最后运用基于集成学习的多学习器算法整合以上模型解决目标问题。项目对各个模型进行了多方面的比较与分析：1、对不同机器学习算法在网络入侵检测任务上的性能进行了对比，包括准确率、召回率、F1分数等指标；2、针对不同算法的特点和适用场景，分析了其优缺点和适用性；3、对于不同算法的模型训练时间和资源消耗也进行了比较，以全面评估其性能。

实验结果显示，在经过特征工程和超参数优化之后，随机森林模型和神经网络模型在交叉验证中达到了RMSE降低至0.1和准确率高于98%的预测效果，同时，对两种回归模型的模型训练时间进行记录，随机森林模型和神经网络模型的训练时间分别为106秒和1.032秒。最终，利用随机森林和神经网络搭建自定义学习器集成学习算法，在98.55s秒的运行时间，得出本项目实验中多种算法的最高准确率：0.9936，证明该算法的有效性

**关键词：**网络入侵检测、机器学习算法、特征工程、集成学习

**目录**

**[一、 研究背景](#_Toc164943951)** [5](#_Toc164943951)

**[1.1 项目背景](#_Toc164943952)** [5](#_Toc164943952)

**[1.2 与本专业的关系](#_Toc164943953)** [6](#_Toc164943953)

**[1.3 与本课程的关系](#_Toc164943954)** [6](#_Toc164943954)

**[二、 算法设计](#_Toc164943955)** [8](#_Toc164943955)

**[2.1 算法整体设计思路](#_Toc164943956)** [8](#_Toc164943956)

**[2.2 数据预处理](#_Toc164943957)** [8](#_Toc164943957)

**[2.2.1 检测异常值](#_Toc164943958)** [8](#_Toc164943958)

**[2.2.2 缺失值补充](#_Toc164943959)** [9](#_Toc164943959)

**[2.3.3 设计编码格式](#_Toc164943960)** [9](#_Toc164943960)

**[2.3.4 归一化和标准化](#_Toc164943961)** [9](#_Toc164943961)

**[2.3 数据统计和特征组合](#_Toc164943962)** [9](#_Toc164943962)

**[2.3.1 箱型统计](#_Toc164943963)** [9](#_Toc164943963)

**[2.3.2 关联性分析](#_Toc164943964)** [10](#_Toc164943964)

**[2.3.3 特征组合和舍弃](#_Toc164943965)** [10](#_Toc164943965)

**[2.4 机器学习模型](#_Toc164943966)** [10](#_Toc164943966)

**[2.4.1 回归模型](#_Toc164943967)** [10](#_Toc164943967)

**[2.4.2 分类模型](#_Toc164943968)** [11](#_Toc164943968)

**[2.4.3 集成学习模型](#_Toc164943969)** [13](#_Toc164943969)

**[三、 实验部署](#_Toc164943970)** [16](#_Toc164943970)

**[3.1 数据集介绍](#_Toc164943971)** [16](#_Toc164943971)

**[3.2 数据集处理](#_Toc164943972)** [18](#_Toc164943972)

**[3.2.1箱型图统计](#_Toc164943973)** [18](#_Toc164943973)

**[3.2.2 异常值去除](#_Toc164943974)** [18](#_Toc164943974)

**[3.2.3 设计编码格式](#_Toc164943975)** [19](#_Toc164943975)

**[3.2.3 归一化和标准化](#_Toc164943976)** [19](#_Toc164943976)

**[3.2.4 相关性分析](#_Toc164943977)** [19](#_Toc164943977)

**[3.2.5 特征组合和舍弃](#_Toc164943978)** [21](#_Toc164943978)

**[3.3建立回归模型](#_Toc164943979)** [21](#_Toc164943979)

**[3.3.1最小二乘法](#_Toc164943980)** [22](#_Toc164943980)

**[3.3.2梯度下降法](#_Toc164943981)** [22](#_Toc164943981)

**[3.4建立分类模型](#_Toc164943982)** [23](#_Toc164943982)

**[3.4.1 神经网络](#_Toc164943983)** [23](#_Toc164943983)

**[3.4.2 支持向量机（SVM）](#_Toc164943984)** [23](#_Toc164943984)

**[3.4.3 逻辑回归](#_Toc164943985)** [24](#_Toc164943985)

**[3.5建立集成学习模型](#_Toc164943986)** [24](#_Toc164943986)

**[3.5.1随机森林算法](#_Toc164943987)** [24](#_Toc164943987)

**[3.5.2 自定义学习器集成学习算法](#_Toc164943988)** [25](#_Toc164943988)

**[3.6 模型超参数优化](#_Toc164943989)** [25](#_Toc164943989)

**[四、 结果分析](#_Toc164943990)** [27](#_Toc164943990)

**[4.1线性回归拟合效果](#_Toc164943991)** [27](#_Toc164943991)

**[4.1.1最小二乘法](#_Toc164943992)** [27](#_Toc164943992)

**[4.1.2梯度下降](#_Toc164943993)** [28](#_Toc164943993)

**[4.2 分类模型预测效果](#_Toc164943994)** [28](#_Toc164943994)

**[4.2.1 SVM模型](#_Toc164943995)** [28](#_Toc164943995)

**[4.2.2 神经网络](#_Toc164943996)** [29](#_Toc164943996)

**[4.3 集成学习效果](#_Toc164943997)** [30](#_Toc164943997)

**[4.3.1随机森林法](#_Toc164943998)** [30](#_Toc164943998)

**[4.3.2自定义学习器集成学习算法](#_Toc164943999)** [31](#_Toc164943999)

**[4.4结论](#_Toc164944000)** [31](#_Toc164944000)

**一、 研究背景**

**1.1 项目背景**

随着互联网的普及和信息技术的迅速发展，网络安全问题已成为当今社会面临的重要挑战之一。网络入侵作为影响网络安全的主要因素之一，其对个人、企业甚至国家的信息资产和利益造成的威胁日益严重。网络入侵可能导致数据泄露、系统瘫痪、服务中断等严重后果，给各方带来巨大损失，因此，及时发现和有效防范网络入侵行为显得尤为重要。

传统的网络入侵检测系统主要依赖于预先定义的规则和特征库，这些规则和特征基于专家经验和已知攻击模式，用于识别网络流量中的异常行为。然而，随着网络环境的不断演变和攻击手段的日益复杂化，传统的基于规则和特征库的方法面临着一些挑战。首先，维护规则和特征库需要大量的人力和时间成本，并且难以跟上网络安全威胁的迅速变化。其次，这种方法往往只能检测到已知的攻击模式，对于新型的、未知的攻击形式则显得束手无策。因此，寻求一种更加智能、自适应的网络入侵检测方法势在必行。

在面对不断演变的网络安全威胁时，机器学习算法正成为网络安全领域的一项强大工具，尤其是在网络入侵检测方面。相比传统的基于规则和特征库的检测方法，基于机器学习的网络入侵检测系统具有显著的优势。通过利用大数据和深度学习技术，这些系统能够自动学习网络流量中的模式和趋势，从而实现对异常行为的实时监测和识别。此外，机器学习算法还能够不断优化自身模型，提高对新型攻击的识别能力，并及时调整防御策略以应对不断变化的威胁。因此，基于机器学习的网络入侵检测系统已经成为保护网络安全的重要手段，为网络管理员提供了强大而高效的安全保障。

然而，并非所有的机器学习算法在网络入侵检测中都表现出相同的效果和性能。不同的算法可能对不同类型的网络攻击有不同的适应性和准确性，因此有必要对各种机器学习算法在网络入侵检测中的效果进行深入比较与分析。这样的研究不仅有助于选择最合适的算法用于构建网络入侵检测系统，还可以为提高系统的性能和准确性提供重要参考和指导。

因此，本研究旨在对多种机器学习算法在网络入侵检测中的效果进行全面比较与分析。通过评估各算法在不同网络环境和攻击场景下的表现，探究其优缺点和适用性，以期为构建高效、可靠的网络入侵检测系统提供理论指导和实践支持。

**1.2 与本专业的关系**

笔者为计算机科学与技术专业，选修方向为人工智能，而该项目与本专业息息相关。首先，网络入侵检测是本专业领域内的一个重要问题，计科专业培养的网络工程师和网络安全专家需要掌握这方面的知识和技能。通过研究网络入侵检测算法，可以深入了解计算机网络结构、通信协议和安全机制，为今后从事网络安全相关工作打下坚实基础。

其次，项目涉及的机器学习算法是专业的研究热点之一。在网络入侵检测中，利用机器学习算法对网络流量进行分析和分类，可以实现对异常行为的检测和识别。因此，对机器学习算法在网络安全领域的应用和效果进行研究，对于本专业来说具有重要的理论和实践意义。

此外，项目还涉及到数据处理、特征工程、模型评估等技术，这些都是计算机科学与技术专业和人工智能的核心内容。通过对网络入侵检测项目的研究和实践，可以提升数据处理能力、算法设计能力和系统开发能力，为将来从事网络安全、数据分析和人工智能等领域的工作做好准备。

**1.3 与本课程的关系**

本项目与《机器学习》课程有密切的关系，因为该课程提供了理论和实践基础，为项目的实施提供了支持。首先，课程教材《机器学习原理与实践》介绍了各种常用的机器学习算法及其应用场景，为项目提供了算法选择和实现的基础。项目中所采用的算法均取材于该教材，例如线性回归、神经网络、支持向量机等。

其次，课程中的实践环节和项目作业提供了实际操作的机会，帮助掌握机器学习算法的实现和应用。通过课程中的编程练习和项目作业，可以学习到数据预处理、特征工程、模型训练与评估等实际操作技能，为项目的顺利进行提供了技术支持。

除了应用课程中所学的知识外，还可以结合课程知识进行创新。在项目中通过尝试探索一些新的算法改进或者应用新的技术手段来提高网络入侵检测系统的性能。例如，尝试结合了深度学习技术和传统的机器学习算法，构建更加复杂和高效的网络入侵检测模型。另外，还引入增强学习算法来优化入侵检测系统的决策策略，使其在动态网络环境中更加灵活和智能。通过这些创新尝试，可以进一步拓展和应用课程所学的知识，提升项目的研究价值和实用性。

**二、 算法设计**

**2.1 算法整体设计思路**

本项目采用了两种集成学习算法模型对网络入侵检测问题进行求解，在用机器学习模型求解时遵循着数据读入、数据预处理、数据统计和特征组合、库函数导入、模型建立与求解这五大步骤，流程图如图2-1所示：

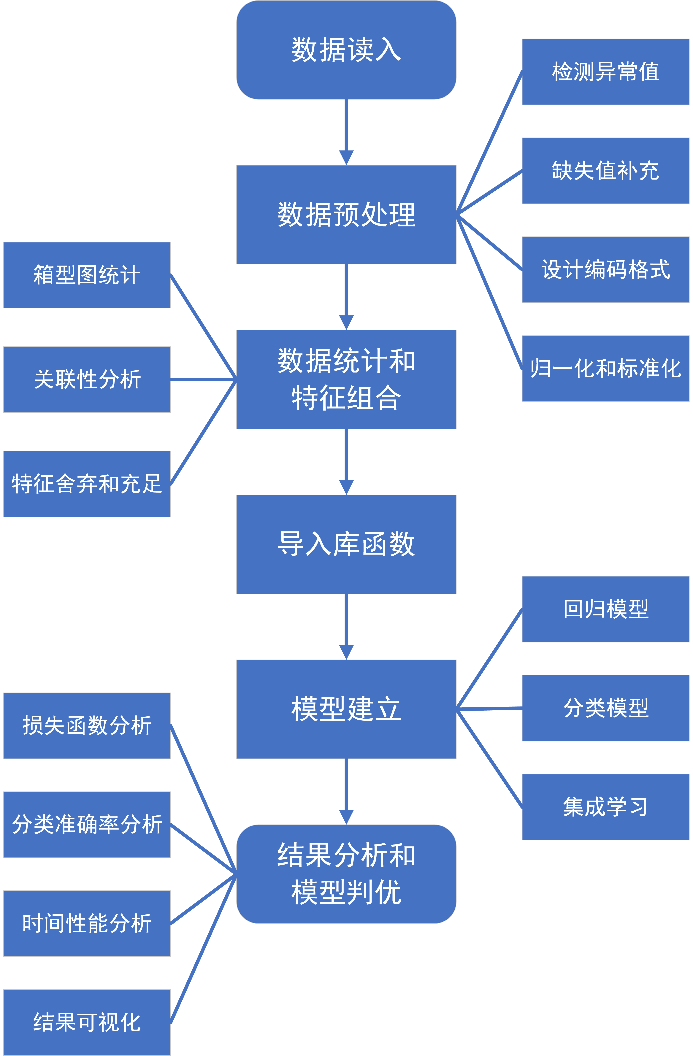


图2-1 算法设计流程图

**2.2 数据预处理**

**2.2.1 检测异常值**

对于每个特征，计算其Z-score值，即将每个数据点的数值减去该特征的均值，再除以该特征的标准差，得到标准化后的数值。对于每个数据点，将所有特征的Z-score值进行汇总，可以采用求和或计算平均值的方式。这一步旨在将多个特征的异常情况综合考虑，从而更全面地判断数据点是否异常。根据想需要，在这一步骤可以去除数据集中1%的数据，可以根据此设定异常值。将被认定为异常值的数据点进行标记或者删除，以保证数据集的质量和模型的稳定性。

**2.2.2 缺失值补充**

K近邻（KNN）是一种基于实例的学习算法，利用数据间的距离度量来进行分类或回归预测。在数据缺失值填充中，KNN算法通过计算样本间的相似度，选取最相近的K个样本来填补缺失值。运用KNN填充缺失值时，首先选择合适的K值，然后计算待填充样本与其他样本的距离，选取距离最近的K个样本，利用它们的特征值进行加权平均或投票决策来填充缺失值。在本项目的数据集中，KNN方法能够有效地填补数据集中的缺失值，并保持数据集的分布和特征间的关系。

**2.3.3 设计编码格式**

设计编码格式是将非数值类型的数据（如字符串）转换为数值型数据的过程，以便于机器学习算法的处理和分析。常见的编码方法包括标签编码和独热编码。本项目采用标签编码将不同的类别标签映射为整数值，适用于分类变量。

**2.3.4 归一化和标准化**

本项目的标准化是将部分数据特征缩放到均值为0、标准差为1的标准正态分布上。具体而言，对于每个特征，通过减去特征的均值，然后除以特征的标准差来实现标准化。标准化后的数据特征具有零均值和单位方差。

本项目的归一化是指将数据特征缩放到指定的范围内，通常是0~1范围内，采用如下公式进行：

**2.3 数据统计和特征组合**

**2.3.1 箱型统计**

箱型图是一种常用的数据可视化工具，用于呈现数据的分布情况和统计特征。它通过展示数据的五个统计量：最小值、第一四分位数（下四分位数）、中位数、第三四分位数（上四分位数）和最大值，来帮助理解数据的整体分布情况。箱型图的中间箱子代表了数据的中间50%范围，上下边界分别表示了数据的上下四分位数。通过观察箱型图，我们可以直观地了解数据的中心趋势、离散程度以及检查异常值。

**2.3.2 关联性分析**

关联性分析是一种用于评估数据集中各个特征之间相关性的方法。通过计算特征之间的相关系数或其他统计指标，可以揭示它们之间的线性或非线性关系。这有助于理解数据集中不同特征之间的相互影响程度，以及它们对目标变量的贡献程度。通过关联性分析，可以识别出与目标变量高度相关的特征，从而有针对性地进行特征选择或建模，以提高模型的预测性能和解释能力。在本项目中，采用皮尔逊相关系数来计算，其计算公式如下：

****

其中：*ρX*,*Y*是特征*X*和*Y*之间的皮尔逊相关系数。cov(*X*,*Y*) 是特征*X*和*Y*的协方差。*σX*和*σY*分别是特征*X*和*Y*的标准差。该公式可以衡量两个变量之间的线性相关程度。

**2.3.3 特征组合和舍弃**

特征组合和舍弃是一种基于先前分析结果的数据预处理方法，旨在优化特征集合以提高模型性能。根据之前的箱型图和关联性分析，可以识别出与目标变量高度相关或者重要的特征，然后将它们进行组合，以创造新的、更具信息量的特征。这样的特征组合可以帮助模型更好地捕捉数据中的模式和规律，从而提高预测性能。

同时，根据之前的分析结果也可以识别出一些无用或者不好用的特征，这些特征可能会干扰模型的学习过程，降低模型的性能。因此选择舍弃这些特征，以简化模型并提高其泛化能力。特征组合和舍弃的过程需要结合领域知识和实际经验，以确保选取的特征集合能够最大程度地反映数据的特征，并为后续建模过程提供更可靠的数据基础。

**2.4** **机器学习模型**

**2.4.1 回归模型**

**基于最小二乘法的线性回归：**使用最小二乘法计算线性回归的公式可以通过矩阵运算来表示。假设有一个包含 m 个样本的数据集合，其中每个样本有 n 个特征，表示为一个 m×(n+1) 的特征矩阵 **X**（第一列全为1，表示截距项），和一个 m×1 的目标值向量 **y**。线性回归模型的参数向量为 **θ**，长度为 (n+1)，则模型的预测值为：

损失函数采用的是残差平方和，即：

通过最小化损失函数，可以得到最优参数向量θ的闭式解：

以上方法可以通过一次矩阵运算即可求解出最优参数向量 θ。

**基于梯度下降的线性回归：**基于梯度下降的方法进行线性拟合的过程是通过迭代更新模型参数来最小化损失函数。模型的预测值为：

采用MSE作为代价函数：

其偏导数为：

****

梯度下降过程可表示为：

****

其中：****

**2.4.2 分类模型**

**逻辑回归：**在逻辑回归中，分类过程的计算公式为：

****

其中，w是特征权重向量，b是偏置项，x是输入特征向量，sigmoid是激活函数。代价函数则采用交叉熵代价函数：

****

其中，m是样本数量，y是真实标签，y\_hat是预测标签，ln是自然对数。

**支持向量机（SVM）:** 支持向量机（Support Vector Machine，SVM）是一种强大的监督学习算法，用于分类和回归分析。对于分类任务，SVM 的目标是找到一个最优的超平面，将不同类别的数据点分隔开,以实现分类的目的。

在二维空间中，超平面是一条直线；在更高维空间中，它是一个超平面。超平面的一般形式可以表示为：



决策函数可以表示为



其中，f(x)是预测的类别，sign()是符号函数，如果其中的值>0则预测为正类别，否则预测为负类别。SVM 通过最大化类别间的间隔来优化超平面的位置。间隔定义为超平面到离它最近的样本点的距离，其预测过程的训练问题可表示为：

 (1)

subject to  (2)

其中，subject to 的意思是“在……条件下”，上面的(1)式为目标函数，(2)式为约束函数。

**随机森林算法**[1]**：**在分类任务当中，随机森林法（Random Forest）是一个强大而灵活的机器学习算法，它能够应对各种复杂的数据情况。随机森林实质上是一组决策树的集合，每个决策树都是基于随机选取的特征子集和随机选择的数据样本进行训练的。这样的随机性使得每棵树都具有独特性，避免了单棵树容易陷入过拟合的问题。

在随机森林中，每个决策树都会对数据进行分裂，直到将数据划分为纯净的叶子节点或达到预定的停止条件。然后，通过投票或平均的方式，将每棵树的预测结果整合起来，作为随机森林的最终预测结果。这种集成方法使得模型对于噪声和异常值具有很好的鲁棒性，同时能够有效地处理高维度数据和大规模数据集。算法流程图如下图2-4.1所示：

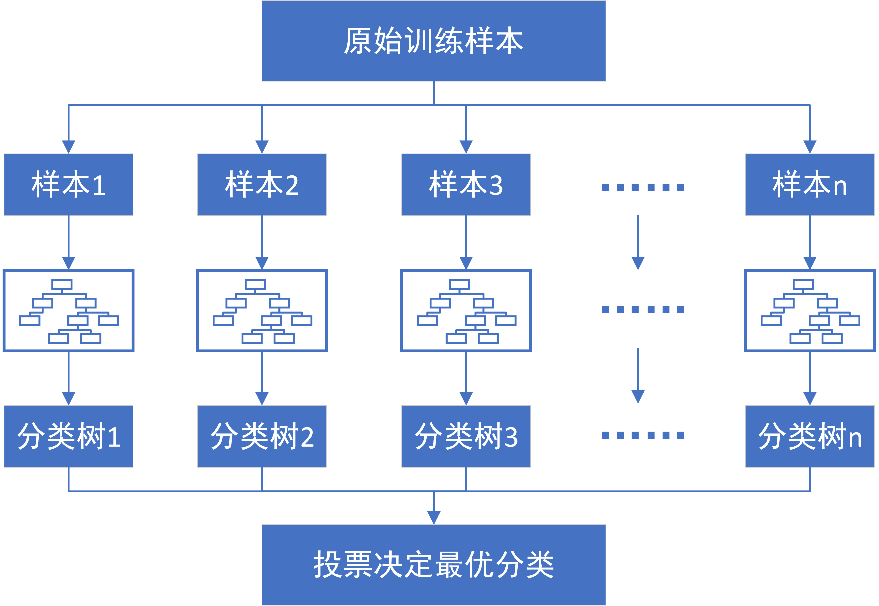


图2-4.1 随机森林算法设计流程图

**2.4.3 集成学习模型**

集成学习ensemble learning通过构建并结合多个学习器来完成学习任务（nulti-classifier system），有时也被称为多分类器系统、基于委员会的学习（committee-based learning）。

集成学习的一般结构，先产生一组“个体学习器”，再用某种策略将它们结合起来。集成中只包含同种类型的个体学习器，称为“同质的（homogeneous）”，集成学习中包含不同类型的个体学习器，这样的集成是“异质的（heterogenous）”，集成学习的结果通过Voting产生，即“少数服从多数”，一般来说，要获得好的集成，个体学习器应该“好而不同”，考虑二分类问题和真实函数，假定基分类器的错误率为，即对每个基分类器，有，假设集成通过简单投票法结合*T*个分类器，若有超过半数的基分类器正确，则集成分类就正确，即：



假设基分类器的错误率互相独立，则有*Heoffding*不等式可知，集成的错误率为：



上式表明，随着集成数*T*的增大，集成的错误率将指数下降，最终趋于0，但该问题的前提假设在于“基学习器的误差相互独立”，但现实中不存在这种相互独立的情况。个体学习器的准确性和多样性本身就存在冲突，一般准确率很高就意味着需要牺牲多样性。算法的流程如图2-4.2所示，

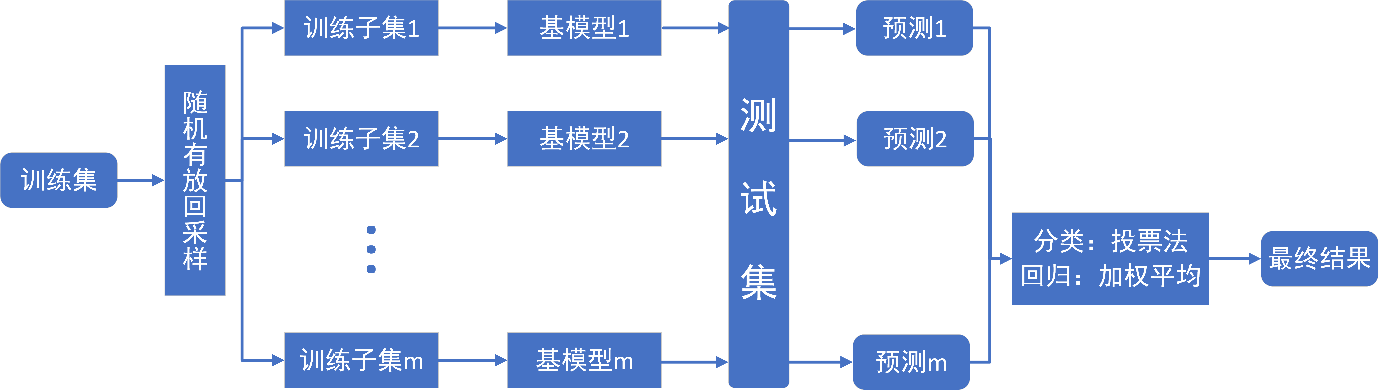


图2-4.2 集成学习算法设计流程图

本项目的算法使用自助采样法随机有放回地对样本进行采样，构建出样本量相等的相互独立的样本数据集，在同一算法中训练出不同的模型。该算法对于分类任务集成策略对于分类问题，一般通过投票法，以多数模型预测结果为最终结果。在此基础之上，本项目将集成策略，即投票法改进为Stacking堆叠泛化法，对于整个数据集，使用不同的基础模型进行训练和预测，得到每个基础模型的预测结果。将这些预测结果作为新的特征，组合成一个新的训练数据集。使用这个新的训练数据集来训练一个元模型，例如逻辑回归、决策树等。最后，使用训练好的元模型来对测试数据进行预测。算法流程如图2-4.3所示

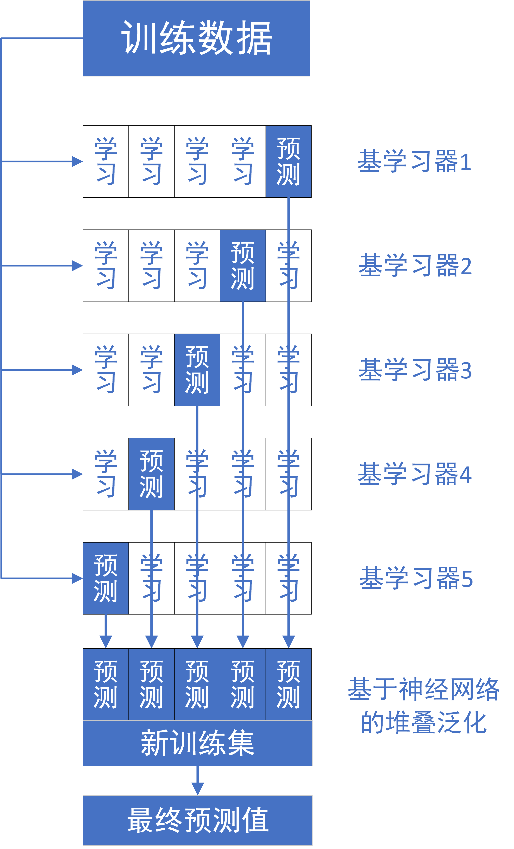


图2-4.3 集成学习算法流程图

上图即为图中上半部分是用一个基础模型进行5折交又验证，在基学习器训练阶段，将数据集划分为5组，每个基学习器将其中不同的四组作为训练集，剩余一组作为测试集输出预测值，在5组训练完成后，将会得到五组不同的预测值和真实值的键值对，将此作为新训练集送入最终基于神经网络的堆叠泛化部分，这一部分将以此进行训练，得出的模型将由测试集进行测试表2-4.1给出了集成学习算法设计伪代码：

表2-4.1 集成学习算法设计伪代码

|  |
| --- |
| **集成学习伪代码** |
| **输入：**  训练集数据;  基学习算法  训练轮数  **过程：**  *For t = 1,2,...,m do*  ;  *End for*  **输出：** |

**三、 实验部署**

**3.1 数据集介绍**

NSL-KDD数据集是一个广泛用于网络入侵检测研究的标准数据集，由美国国防部资助，用于评估网络入侵检测系统的性能。它是KDD Cup 1999数据集的改进版本，去除了KDD Cup 1999中的一些重复和不必要的数据，并增加了一些新的网络流量特征。NSL-KDD数据集共包含4个子集，分别是训练集、测试集、训练20%子集和测试20%子集，其中训练集和测试集用于模型的训练和评估，训练20%子集和测试20%子集用于性能评估的交叉验证。

NSL-KDD数据集中的每条数据代表了网络流量中的一个网络连接，并包含了多个特征来描述这个网络连接。这些特征通常可以分为以下几类：

（1）基本特征：包括源IP地址、目标IP地址、源端口号、目标端口号等基本网络连接信息。

（2）统计特征：包括网络连接持续时间、发送的字节数、接收的字节数、连接的状态等统计信息。

（3）时序特征：包括网络连接的开始时间、结束时间等时间相关信息。

（4）协议特征：包括使用的网络协议类型、传输层协议类型等协议相关信息。

（5）网络流量特征：包括数据包长度、数据包流量、数据包类型等网络流量相关信息。

（6）安全特征：包括是否出现了异常行为、是否属于某种已知的网络入侵类型等安全相关信息。

每条数据都会根据这些特征被标记为正常连接或是特定类型的网络入侵。通过对这些特征的分析和建模，可以帮助识别出潜在的网络安全威胁，并实现网络入侵检测的目标。其详细内容如表3-1：

**表3-1 数据集各特征解释**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 特征英文名称 | 解释 | 数据类型 | |
| 1 | Duration | 连接持续时间 | | int |
| 2 | Protocol\_type | 连接的协议类型，如tcp、udp等 | | str |
| 3 | Service | 服务类型，指网络连接所使用的服务 | | str |
| 4 | Flag | 连接状态标志，如SF、REJ、RSTO等 | | str |
| 5 | Src\_bytes | 从源主机到目标主机的字节数 | | int |
| 6 | Dst\_bytes | 从目标主机到源主机的字节数 | | int |
| 7 | Land | 标志位，连接相同的源IP为1 | | bool |
| 8 | Wrong\_fragment | 错误分段数量 | | int |
| 9 | Urgent | 紧急数据包数量 | | int |
| 10 | Hot | 连接中的“热”指标数量 | | int |
| 11 | Num\_failed\_logins | 登录失败次数 | | int |
| 12 | Logged\_in | 表示用户是否成功登录。 | | bool |
| 13 | Num\_compromised | 受到攻击影响的主机数量。 | | int |
| 14 | Root\_shell | 是否拥有root权限。 | | bool |
| 15 | Su\_attempted | 是否尝试以超级用户身份登录。 | | int |
| 16 | Num\_root | root用户访问次数。 | | int |
| 17 | Num\_file\_creations | 文件创建操作次数。 | | int |
| 18 | Num\_shells | shell命令执行次数。 | | int |
| 19 | Num\_access\_files | 访问控制文件的操作次数。 | | int |
| 20 | Num\_outbound\_cmds | 发送命令控制操作的次数。 | | int |
| 21 | Is\_host\_login | 是否是主机登录。 | | bool |
| 22 | Is\_guest\_login | 是否是访客登录。 | | bool |
| 23 | Count | 2s内连接到同目标主机连接数量 | | int |
| 24 | Srv\_count | 2s内连接到同服务连接数量。 | | int |
| 25 | Serror\_rate | “SYN”错误率 | | float |
| 26 | Srv\_serror\_rate | 2s内连接到同服务“SYN”错误率 | | float |
| 27 | Rerror\_rate | “REJ”错误率 | | float |
| 28 | Srv\_rerror\_rate | 2s内连接到同服务“REJ”错误率 | | float |
| 29 | Same\_srv\_rate | 2s内连接到相同服务连接比。 | | float |
| 30 | Diff\_srv\_rate | 2s内连接不同服务的连接比。 | | float |
| 31 | Srv\_diff\_host\_rate | 2s内连接不同目标主机的连接比 | | float |
| 32 | Dst\_host\_count | 连接到同目标主机的连接数量 | | int |
| 33 | Dst\_host\_srv\_count | 2s内连接同目标主机不同服务连接数 | | int |
| 34 | Dst\_host\_same\_srv\_rate | 连接同一目标主机相同服务连接 | | float |
| 35 | Dst\_host\_diff\_srv\_rate | 2s内连接同一目标主机不同服务连接 | | float |
| 36 | Dst\_host\_same\_src\_port\_rate | 连接同目标主机同源连接比 | | float |
| 37 | Dst\_host\_srv\_diff\_host\_rate | 2s内连接同目标主机不同源连接比率 | | float |
| 38 | Dst\_host\_serror\_rate | 连接同目标主机“SYN”错误率 | | float |
| 39 | Dst\_host\_srv\_serror\_rate | 2s内连接同目标主机“SYN”错误率 | | float |
| 40 | Dst\_host\_rerror\_rate | 连接同目标主机“REJ”错误率 | | float |
| 41 | Dst\_host\_srv\_rerror\_rate | 2s内连接同目标主机“REJ”错误率 | | float |
| 42 | Class | 标签（正常或攻击） | | str |
| 43 | Index | 分数（流量输入本身的严重性） | | int |

**3.2 数据集处理**

**3.2.1箱型图统计**

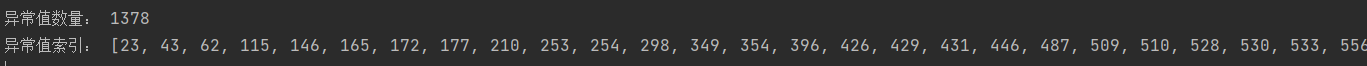
在此步骤的实施中，核心代码是plt.boxplot(df[column], labels=[column])，其可以利用matplotlib中集成的箱型图统计函数直接绘制箱型图，代码部分运行结果如图3-2.1所示：

|  |  |
| --- | --- |
|  |  |
| (a)特征count的箱型图统计 | (b)特征Srv\_count的箱型图统计 |
|  |  |
| (c)特征Diff\_srv\_rate的箱型图统计 | (d)特征Srv\_diff\_srv\_rate的箱型图统计 |
| **图3-2.1 箱型图统计部分结果** | |

根据箱型图统计结果，发现该数据集仍有很多异常值需要去除。

**3.2.2 异常值去除**

采用基于 Z-score 的数据集异常值检测方法，其核心函数为 z\_scores = (features - features.mean()) / features.std()，通过设置适当的阈值参数，对训练集和测试集中的共约 140,000 条数据进行异常值检测。在这个过程中，共发现了 1,578 条异常数据，并将其从数据集中删除。该步骤运行结果如图3-2.2所示：



**图3-2.2 异常值去除**

**3.2.3 设计编码格式**

在本项目的数据集中，每条数据的多个特征均为str类型，不方便后续模型的读入，因此将其转换成数值类型。实现流程则为遍历str类型特征列中的每个数据，若其未出现过，则为其编码并存入字典，若其已在字典中，则将字典中的相同值对应的编码赋予它，该流程伪代码如下表：

**表3-2.1 设计编码格式**

|  |
| --- |
| **设计编码格式伪代码** |
| **输入：**  训练集数据;  **过程：**  *for m = 1,2,...,t do*  *if x not in label\_map:*  *label\_map[m] = x;*  *x = new\_label;*  *end for*  **输出：**  *label\_map* |

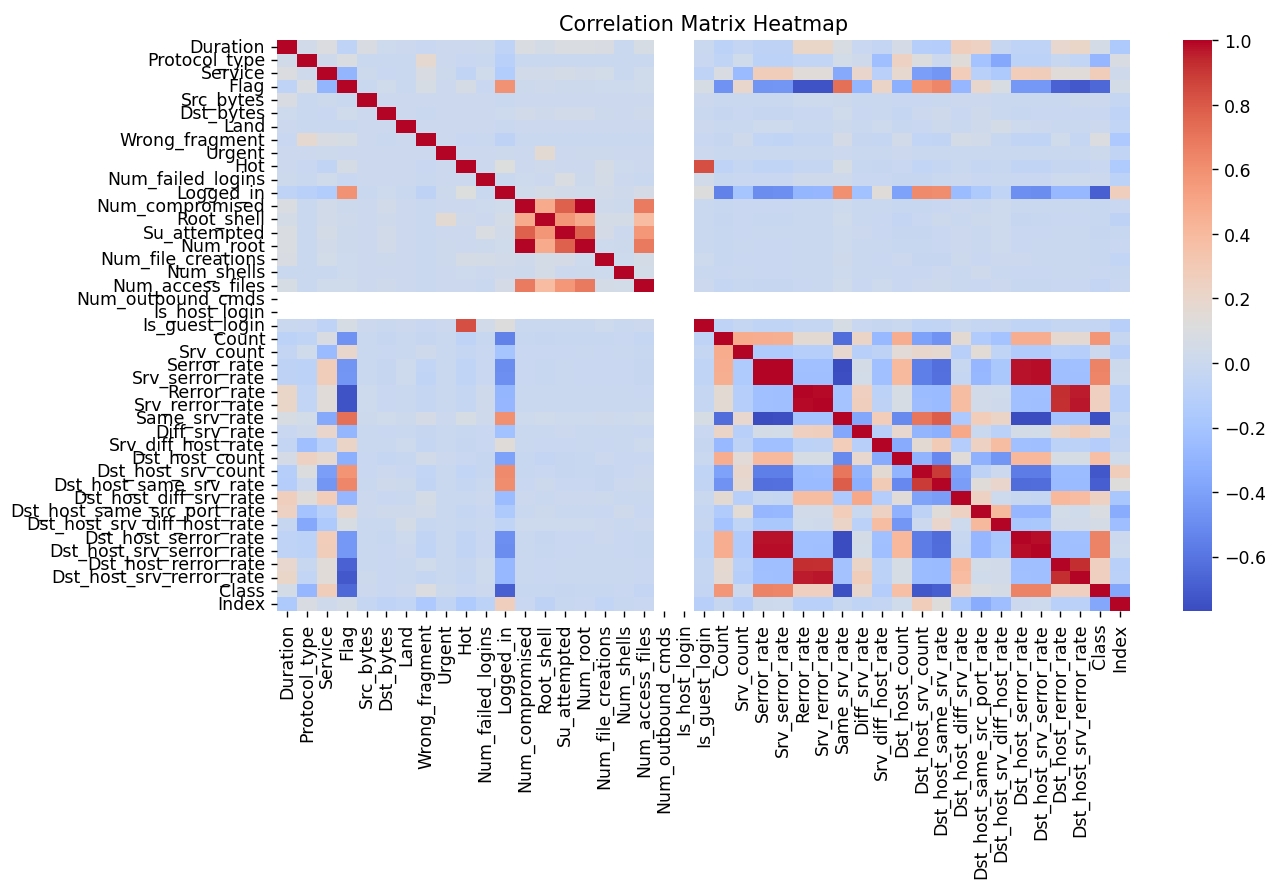
在本项目的数据集中，每条数据的唯一标签也为str值，为方便后续进行回归、分类任务，将其中的“normal”值赋值为0，将非“normal”数据赋值为1。即将其转换为一个类似二分类任务的数据集。

**3.2.3 归一化和标准化**

在本项目中，对于数值相对于其他列特征明显较大的特征，采用基于Min-Max的归一化方法，其核心代码为scaled\_data = (data - min\_val) / (max\_val - min\_val)，其可以将将数据缩放到指定的范围内（[0, 1]），计算方法是将原始数据减去最小值，然后除以最大值减最小值。

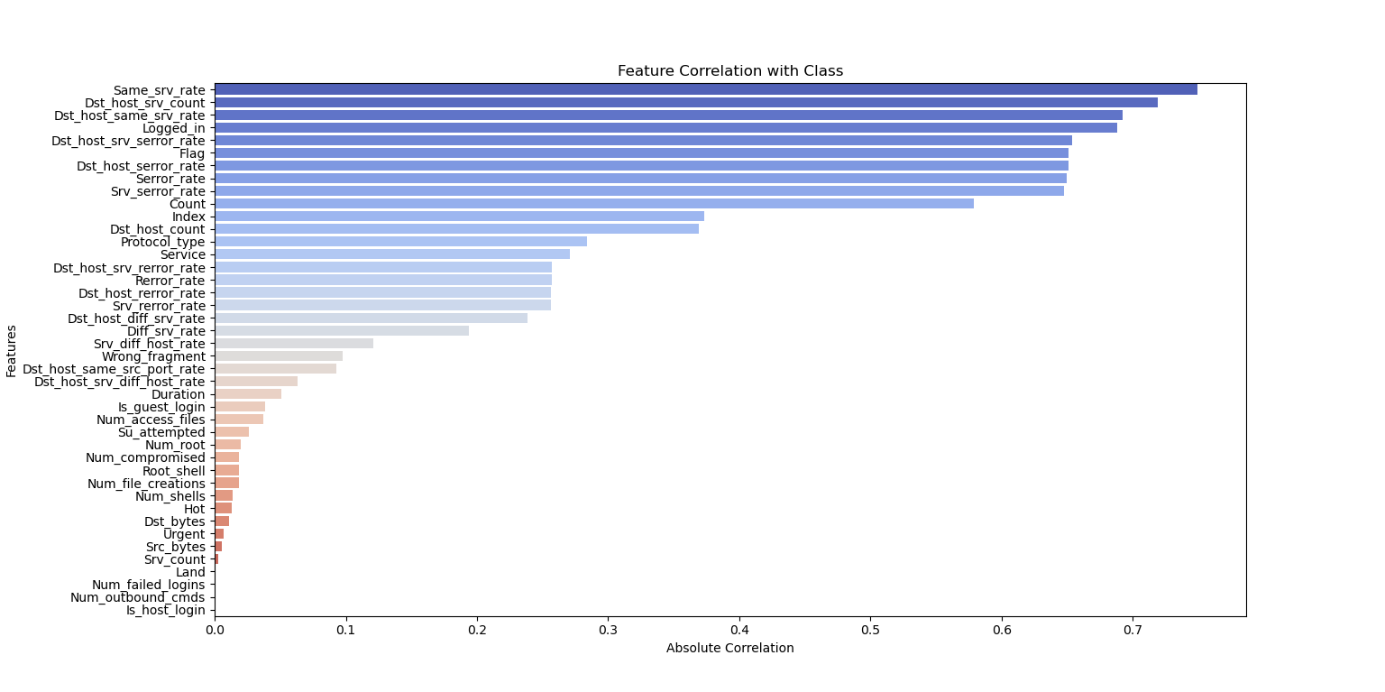
**3.2.4 相关性分析**

计算协方差的代码为*cov = np.mean(x \* y) - np.mean(x) \* np.mean(y)*，计算标准差的代码为*std\_x = np.std(x)*，而皮尔逊相关系数则可通过*pearson\_corr = cov / (std\_x \* std\_y)*得到。最后可视化数据集中各个特征值之间的关系，如图3-2.3所示：



**图3-2.3 特征相关性热力图**

在此基础之上，通过相关性分析得出和数据标签即本项目预测目标相关性较大的特征列，结果如图3-2.4所示：



**图3-2.4 与标签相关性排行**

**3.2.5 特征组合和舍弃**

根据特征间相关性分析得出，Srv\_serror\_rate（过去两秒内“SYN”错误率）和Serror\_rate（“SYN”错误率）相关性较高，对其进行组合，同理，对于Srv\_rerror\_rate（过去两秒内“REJ”错误率）和Rerror\_rate（“REJ”错误率）, Dst\_host\_serror\_rate（连接同目标主机“SYN”错误率）和Dst\_host\_srv\_serror\_rate（过去2s连接同目标主机“SYN”错误率），Dst\_host\_rerror\_rate（连接同目标主机“REJ”错误率）和Dst\_host\_srv\_rerror\_rate（过去2s连接同目标主机“REJ”错误率）进行组合。

根据与数据标签的相关性排行图中所示，数据集将保留与标签相关性较高的前12列数据，其余将被舍弃，以方便后续预测。现保存数据特征列如下表所示：

**表3-2.2 现保留数据特征**

|  |  |  |
| --- | --- | --- |
| 序号 | 特征名称 | 解释 |
| 1 | Protocol\_type | 连接的协议类型，如tcp、udp等 |
| 2 | Service | 服务类型，指网络连接所使用的服务 |
| 3 | Flag | 连接的状态标志，如SF、REJ、RSTO等 |
| 4 | Logged\_in | 表示用户是否成功登录。 |
| 5 | Count | 2s内连接到同目标主机连接数量 |
| 6 | Same\_srv\_rate | 2s内连接到相同服务连接比。 |
| 7 | Dst\_host\_count | 连接到同目标主机的连接数量 |
| 8 | Dst\_host\_srv\_count | 2s内连接同目标主机不同服务连接数 |
| 9 | Dst\_host\_same\_srv\_rate | 连接同一目标主机相同服务连接 |
| 10 | Dst\_host\_diff\_srv\_rate | 2s内连接同一目标主机不同服务连接 |
| 11 | Serror | “SYN”错误率 |
| 12 | Rerror | “REJ”错误率 |

**3.3建立回归模型**

本项目面对的是一个典型的二分类任务，即数据集的标签仅包含两个类别，分别用0和1表示。尽管二分类问题通常会选择专门设计的分类算法，如逻辑回归或支持向量机，但是仍然可以考虑使用线性回归算法来处理这样的问题。线性回归算法是一种常见的回归算法，用于建立特征与连续型目标变量之间的线性关系。然而，在二分类任务中可以将数据标签视为一种连续型目标变量，并尝试使用线性回归模型来拟合这些标签，以便根据多维数据特征对标签进行预测。

尽管线性回归算法最初是为解决连续型目标变量的预测问题而设计的，但在二分类任务中也具有一定的合理性。这是因为线性回归模型所假设的线性关系可以在一定程度上描述不同类别之间的分界面。例如，在二维空间中，线性回归模型可以用一条直线来分割两个类别。虽然这种分界面可能不是最优的，但在某些情况下，线性回归模型仍然能够提供良好的分类效果。

此外，线性回归算法具有简单、易于理解和实现的特点，对于对项目初步探索来说是一种很好的起点。通过将二分类任务视为回归问题，可以利用已有的线性回归算法和工具来进行建模和分析，而无需引入额外的复杂性。

**3.3.1最小二乘法**

当建立基于最小二乘法的线性回归模型时，首先准备以Excel表格的形式存储的数据，其中包含了数据特征和目标标签。读取Excel数据使用data = *pd.read\_excel("train.xlsx")*，而对于特征和标签的分离则使用*X = data.iloc[:, 1:]*(特征)和*y = data.iloc[:, 0]*(标签)。

接下来使用最小二乘法来计算回归系数。最小二乘法是一种经典的线性回归方法，通过最小化预测值与实际值之间的平方误差来拟合数据。计算出的回归系数表示了特征变量与目标变量之间的线性关系。首先需要添加偏置项*X['intercept'] = 1*，使用代码*coefficients = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)*计算回归系数，一旦得到了回归系数就可以使用它们进行预测。通过将特征数据与回归系数相乘，再加上偏置项，即可以得到对目标变量的预测值，使用代码为*predictions = X.dot(coefficients)*。

**3.3.2梯度下降法**

首先需要读取数据并进行预处理，这一步骤同3.3.1中相关内容一致。然后，初始化回归系数为零，并设置学习率和迭代次数，即*coefficients = [0.0] \* len(X[0])，learning\_rate = 0.01*，*n\_iterations = 1000*。然后，使用梯度下降算法来迭代更新回归系数。该步骤伪代码如表3-2.3所示

**表3-3.1 现保留数据特征**

|  |
| --- |
| **梯度下降伪代码** |
| **输入：**  迭代次数*n\_iterations*;  学习率*learning\_rate*  **过程：**  *for i in range(n\_iterations):*  *gradient = [0.0] \* len(X[0]) 初始化梯度*  *for i, row in enumerate(X):*  *prediction = sum(coefficients[j]\*row[j] for j in range(len(row)))计算预测值*  *error = y[i] - prediction计算误差*  *for j in range(len(row)): 更新梯度*  *gradient[j] += (-2 \* error \* row[j])*  *for j in range(len(coefficients)): 更新回归系数*  *coefficients[j] -= learning\_rate \* gradient[j]*  **输出：**  模型参数*coefficients* |

最后，可以使用得到的回归系数进行预测：*predictions = [sum(coefficients[j] \* row[j] for j in range(len(row))) for row in X]*。并根据实际值和预测值评估模型性能：*mse = sum((predictions[i] - y[i]) \*\* 2 for i in range(len(y))) / len(y)*。

**3.4建立分类模型**

**3.4.1 神经网络**

在本项目中使用神经网络模型进行分类任务。神经网络是一种强大的机器学习模型，能够自动学习数据中的复杂模式和关系，因此在分类任务中被广泛应用。

首先，我们需要定义神经网络的结构。本次使用的激活函数为Relu函数：*np.maximum(0, x)*和sigmoid函数*np.maximum(0, x)*，同时具有两个隐藏层，在每个隐藏层中，用*weights\_1 = np.random.randn(12, 64)*初始化模型参数，使用*biases\_1 = np.random.randn(64)*初始化偏置值，使用*layer\_1\_output = relu(np.dot(input\_data, weights\_1) + biases\_1)*或*layer\_1\_output = sigmoid(np.dot(input\_data, weights\_1) + biases\_1)*定义其激活函数，对于前向传播，则用*prediction = neural\_network(input\_data)*结合以上代码即可完成。之后便可计算代价函数，即为交叉熵：*loss = -target \* np.log(output) - (1 - target) \* np.log(1 - output)*，之后计算输出层的误差和梯度。通过反向传播算法，依次计算隐藏层的误差和梯度：*d\_loss\_output = (output - target) / (output \* (1 - output))，d\_output\_layer = d\_loss\_output \* output \* (1 - output)*。最后，根据梯度和学习率，更新网络的权重和偏置：*weights\_output -= learning\_rate \* np.dot(layer\_2\_output.T, d\_output\_layer)*，*biases\_output -= learning\_rate \* np.sum(d\_output\_layer, axis=0)*。以上即为本项目神经网络的建立方法。

**3.4.2 支持向量机（SVM）**

在本项目中建立了SVM模型进行分类任务，初始化支持向量机模型的参数：用*weights = np.zeros(X\_train.shape[1])*将模型参数初始化为全0矩阵，*bias = 0*初始化偏置为0。之后训练支持向量机模型的函数。采用梯度下降法来更新参数。首先计算预测值计算预测值*y\_pred = np.dot(self.weights, x) + self.bias*，之后更新参数：*if y\_train[i] \* y\_pred <= 0*判断预测值与真实标签的符号是否相同，*weights += learning\_rate \* y\_train[i] \* x*更新权重，*bias += learning\_rate \* y\_train[i]*更新偏置。在多轮迭代之后，运用测试集评估模型效果：使用*prediction = np.dot(self.weights, x) + self.bias*计算预测值，*y\_pred.append(1 if prediction >= 0 else 0)*预测值大于等于0时输出1，否则输出0。以上即为本项目建立SVM模型的基本流程。

**3.5建立集成学习模型**

**3.5.1随机森林算法**

随机森林是一种集成学习方法，通过构建多个决策树来进行分类或回归任务。每棵决策树都是根据随机抽样的数据集和随机选择的特征进行训练的。随机森林的基本思想是通过多个弱学习器的集成来提高整体模型的准确性和泛化能力。

构建决策树：随机森林中的每棵决策树都是独立构建的。首先从训练集中随机抽取一定比例的样本，这个过程称为bootstrap抽样。然后，在每个节点处，随机选择一定数量的特征进行评估，并选择最佳的特征和阈值来进行划分，直到达到指定的树的深度或节点样本数量不再增加为止。构建决策树需要定义树的划分索引index，之后定义左子树右子树等节点，最后定义停止条件（当节点样本数小于等于阈值时停止）*if len(y) <= self.min\_samples\_split:return LeafNode(y)*。

在构建完单个决策树模型后，即可构建整个随机森林类，需要给出决策树数量及每棵树的最大深度。之后训练随机森林：*for i in range(self.n\_estimators): tree = self.\_build\_tree(X, y)，self.trees.append(tree)*。最后取每棵树的预测值进行预测：*for i, tree in enumerate(self.trees): predictions[:, i] = self.\_predict\_tree(X, tree)，return np.round(np.mean(predictions, axis=1))* ，即取所有树的平均值，并进行四舍五入。

**3.5.2 自定义学习器集成学习算法**

在基于多决策树和样本抽样的随机森林集成学习算法基础上，将弱学习器的类别从决策树扩展到了更多样化的机器学习算法，同时由于各个学习器种类不同，全部样本也可以送入各个学习器重，在最后可用神经网络耦合各个学习器的预测值，并得到最终输出。

以一个基于随机森林和最小二乘法线性回归的集成学习模型为例，本模型先将完整数据分别送入随机森林学习器和线性回归学习器，在两个学习器预测完成后将其预测值作为神经网络的输入，神经网络将根据学习器的输出以数据标签为目标值进行最终预测，起流程如表3-5.1所示：

**表3-5.1 集成学习伪代码**

|  |
| --- |
| **集成学习伪代码** |
| **输入：**  X\_train, y\_train = load\_data();输入数据  **过程：**  *svm\_model = SVM();*初始化SVM学习器  *linear\_regression\_model = LinearRegression();*初始化线性回归学习器  *neural\_network\_model = NeuralNetwork();*初始化神经网络  *svm\_model.train(X\_train, y\_train);*SVM学习器训练  *linear\_regression\_model.train(X\_train, y\_train);*线性回归学习器计算  *svm\_predictions = svm\_model.predict(X\_train);*SVM学习器预测  *linear\_regression\_predictions = linear\_regression\_model.predict(X\_train)*线性回归学习器预测  *X\_input = np.concatenate((svm\_predictions, linear\_regression\_predictions), axis=1);*  *neural\_network\_model.train(X\_input, y\_train);* 将学习器的输出作为神经网络输入进行训练  *final\_predictions = neural\_network\_model.predict(X\_input);*得出最终结果  **输出：**  最终结果*final\_predictions* |

**3.6 模型超参数优化**

在本项目中主要采用贝叶斯优化[2]进行模型超参数选取。贝叶斯优化方法（顺序优化方法的一种，sequential model-besed optimization）可以借鉴已有的结果进而影响后续的模型超参数选择。这也限制了模型训练评估的计算次数，因为只有有望提高模型性能的超参数组合才会被进行计算。

贝叶斯优化是通过构造一个函数的后验分布（高斯过程）来工作的，该后验分布最好地描述了要优化的函数。随着观测次数的增加，后验分布得到改善，算法更加确定参数空间中哪些区域值得探索，哪些区域不值得探索。其实现步骤如表3-6.1所示：

**表3-6.1 贝叶斯优化伪代码**

|  |
| --- |
| **贝叶斯优化伪代码** |
| **输入：**  *F*模型函数  *X*超参搜索空间  *S*采样数据集  *M*输出值  **过程：**  *for i to T do*循环选T次参数   1. 用当前数据集 拟合代理模型 ，实现模型更新   选择使采集函数 最大的超参组合  将参数带入函数中，得到输出值  将输出值加入到现有数据集中  *end for*  **输出：**  最优参数*x* |

**四、 结果分析**

**4.1线性回归拟合效果**

**4.1.1最小二乘法**

在使用最小二乘法对数据特征和数据标签之间的关系进行线性拟合时，得出的RMSE为0.218347，训练时间：0.013s，为其加上门限后，准确率为0.9484，拟合效果可视化如图4-1.1所示。

如图4-1.1（a）所示，其中图片横轴为真实值，纵轴为预测值，在可视化时对数据的类别数值加上了小范围高斯噪声以将其离散化。根据预测结果（蓝色点），算法给出了拟合的曲线（红色虚线），可以看出模型得出了一条使误差最低的拟合直线，从数据分布上可以得出大多数据点紧密分布在拟合曲线的两端，但仍有部分数据点与拟合曲线的预测结果相反，由此可知，线性回归算法也可用于在二分类任务中对数据类别的大致预测，但预测精度较低

为了更直观的表示随着预测的数据和真实数据的分布，在评价该模型时选取了和数据类别有最高相关性的特征Serror\_srv\_rate为横坐标，数据类别为纵坐标如图4-1.1（b）所示，其中图片横轴为最高相关性特征（Serror\_srv\_rate）真实值，纵轴为预测值（数据标签），从图上可以看出Serror\_srv\_rate=1.0附近的数据并未得到很好预测，由此可得算法的预测值和真实值拟合得准确度仍有待提升。

|  |  |
| --- | --- |
|  |  |
| (a)最小二乘法线性回归拟合 | (b)通过最高相关性特征进行预测散点图 |
| **图4-1.1 最小二乘法线性回归可视化** | |

**4.1.2梯度下降**

在使用梯度下降法对数据特征和数据标签之间的关系进行线性拟合时，得出的RMSE为0.227106，得到最小误差的训练时间约为：0.1515s，为其加上门限后，准确率为0.948799，训练效果可视化如图4-1.2所示：

|  |  |
| --- | --- |
|  |  |
| (a)损失函数下降趋势 | (b)预测值和真实值对比 |
| **图4-1.2 梯度下降训练效果可视化** | |

如图4-1.2（a）所示，图片展示了随着迭代次数的提升，整体误差有变小的趋势，在总共4000轮迭代中，在3500轮到达最小值。

如图4-1.2（b）所示，其中图片蓝色直线为数据的两个分类，而红色散点即为算法输出的预测值，可以看出大部分预测数据密集分布在真实值附近，但仍有小部分数据在真实值标签之间，视为错误数据。由此可得，该算法的准确度仍有待提升。

**4.2 分类模型预测效果**

**4.2.1 SVM模型**

在使用SVM进行本项目的分类任务时，得出的RMSE为0.22095，运行时间约为：141.1394s，准确率为0.9511，训练效果可视化如图4-2.1所示。

如图4-2.1（a）所示，该图展示了SVM输出结果的混淆矩阵，其中真正例（True Positives，TP）： 表示模型正确预测为正类的样本数量，共2586条。假正例（False Positives，FP）： 表示模型错误地将负类样本预测为正类的数量，共88条。真负例（True Negatives，TN）： 表示模型正确预测为负类的样本数量，共158条。假负例（False Negatives，FN）： 表示模型错误地将正类样本预测为负类的数量，共2207条。

如图4-2.1（b）所示，该图展示了SVM的预测效果，其中黄色点为预测值，而蓝色点为真实值，在可视化时对数据的类别数值加上了小范围高斯噪声以将其离散化。可以看出大部分预测值都紧密分布在真实值附近，即为分类正确，但仍有少部分预测值出现了错误，集中在图中的左上角和右下角。

|  |  |
| --- | --- |
|  |  |
| (a)混淆矩阵 | (b)预测值和真实值对比 |
| **图4-2.1 SVM效果可视化** | |

**4.2.2 神经网络**

在使用神经网络进行本项目的分类任务时，超参数经优化后设置为每个epoch630个step，共训练100epoch得出的RMSE为0.10911，运行时间约为：106.8682s，准确率为0.98968，训练效果可视化如图4-2.2所示。

如图4-2.1（a）所示，该图展示了神经网络再多轮迭代中的误差下降趋势，在约100epoch后降到最低。

如图4-2.1（b）所示，该图展示了神经网络预测效果，其中红色点为预测值，而蓝色点为真实值，在可视化时对数据的类别数值加上了小范围高斯噪声以将其离散化。可以看出大部分预测值都紧密分布在真实值附近，即为分类正确，但仍有极少部分预测值出现了错误，集中在图中的左上角和右下角，错误率相比SVM算法得到了显著降低。

|  |  |
| --- | --- |
|  |  |
| (a)损失函数变化趋势 | (b)预测值和真实值对比 |
| **图4-2.2 神经网络效果可视化** | |

**4.3 集成学习效果**

**4.3.1随机森林法**

在使用随机森林法进行本项目的分类任务时，得出的RMSE为0.08797，运行时间约为：1.0320s，准确率为0.99226，训练效果可视化如图4-3.1所示：

|  |  |
| --- | --- |
|  |  |
| (a)混淆矩阵 | (b)预测值和真实值对比 |
| **图4-3.1 随机森林法效果可视化** | |

如图4-2.1（a）所示，该图展示了随机森林输出结果的混淆矩阵，其中真正例（True Positives，TP）： 表示模型正确预测为正类的样本数量，共2658条。假正例（False Positives，FP）： 表示模型错误地将负类样本预测为正类的数量，共16条。真负例（True Negatives，TN）： 表示模型正确预测为负类的样本数量，共23条。假负例（False Negatives，FN）： 表示模型错误地将正类样本预测为负类的数量，共2342条。可以看出较SVM相比，该算法在假正例和真负例的错误率上显著低于SVM模型。

如图4-2.1（b）所示，该图展示了随机森林的预测效果，其中黄色点为预测值，而蓝色点为真实值，在可视化时对数据的类别数值加上了小范围高斯噪声以将其离散化。可以看出大部分预测值都紧密分布在真实值附近，即为分类正确，但仍有极少部分预测值出现了错误，集中在图中的左上角和右下角。该算法效果与神经网络拟合值相近，显著优于SVM算法。

**4.3.2自定义学习器集成学习算法**[3]

如表4-3.2所示，该图展示了在多次试验后集成学习各部件及总的平均预测效果，该算法共消耗98.55s，在准确率上达到0.9936，分别在随机森林学习器和神经网络学习器上实现0.14%和0.5%的提升。

**表4-3.2 集成学习效果**

|  |  |  |
| --- | --- | --- |
| **部分** | **准确率** | **运行时间** |
| 随机森林学习器 | 0.9922 | 1.010s |
| 神经网络学习器 | 0.9884 | 78.545s |
| 集成学习 | 0.9936 | 18.995s |

**4.4结论**

本项目深入探讨了机器学习算法在网络入侵检测领域的应用，并对线性回归、神经网络、支持向量机、随机森林等多种常见算法进行了细致比较，并在最后根据集成学习的理论和stacking堆叠泛化的方法建立了自定义多学习器模型，通过数据预处理、特征工程和贝叶斯优化等方法对模型的训练进行优化，研究结果显示，在交叉验证中，随机森林和神经网络表现出色，其准确率超过98%。值得一提的是，随机森林的训练时间为1.032秒，展现了不仅在性能上优异，而且在效率上也有相当的竞争力。

本项目在基于集成学习的多学习器算法中进一步提升了预测准确性，最高准确率达到了0.9936。这一结果突显了集成学习在网络入侵检测中的潜力，并强调了利用多种算法的组合来提高系统性能的重要性。整体实验数据如表4-4.1

**表4-4.1 集成学习效果**

|  |  |  |  |
| --- | --- | --- | --- |
| **算法类别** | **RMSE** | **准确率** | **运行时间** |
| 最小二乘线性回归 | 0.2183 | 0.9484 | 0.013s |
| 梯度下降线性回归 | 0.2271 | 0.9488 | 0.152s |
| SVM模型 | 0.2210 | 0.9511 | 141.139s |
| 神经网络分类模型 | 0.1091 | 0.9897 | 106.868s |
| 随机森林法 | 0.0880 | 0.9923 | 1.032s |
| 自定义集成学习 | 0.0864 | 0.9936 | 98.55s |

综上，通过对不同机器学习算法在网络入侵检测中的深入比较和分析，本研究为构建高效、可靠的网络入侵检测系统提供了理论支持和实践指导。此项目不仅验证了机器学习技术在网络安全领域的适用性和有效性，还展示了通过集成算法和模型创新可以进一步优化系统性能的潜力。

**参考文献**

[1]Breiman, Leo. "Random forests." *Machine learning* 45 (2001): 5-32.

[2]Hutter, Frank, Holger H. Hoos, and Kevin Leyton-Brown. "Sequential model-based optimization for general algorithm configuration." *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*. Springer Berlin Heidelberg, 2011.

[3]Dong, \*\*bin, et al. "A survey on ensemble learning." *Frontiers of Computer Science* 14 (2020): 241-258.