ICS 35.040

CCS L 71

团体标准

人工智能 算子接口

第3部分：机器学习类

Artificial Intelligence — Operator Interface—Part 3: Machine Learning Classes

点击此处添加与国际标准一致性程度的标识

|  |
| --- |
| （征求意见稿） |
| 在提交反馈意见时，请将您知道的相关专利连同支持性文件一并附上。 |

XXXX - XX - XX实施

中关村视听产业技术创新联盟 发布

XXXX - XX - XX发布

T/AI XXX.XX—XXXX

|  |  |
| --- | --- |
|   |       |

目  次

[前言 II](#_Toc8570)

[引言 III](#_Toc28571)

[1 范围 1](#_Toc27227)

[2 规范性引用文件 1](#_Toc6028)

[3 术语定义 1](#_Toc20127)

[4 缩略语 1](#_Toc12128)

[5 通则 1](#_Toc30051)

[6 数据结构 1](#_Toc15352)

[7 机器学习类算子接口 1](#_Toc27148)

[7.1 接口列表 1](#_Toc14153)

[7.2 函数接口列表 2](#_Toc29270)

[附录A（资料性附录） 量化原理 47](#_Toc7501)

[A.1 非对称量化方式（Asymmetric, ASYM） 47](#_Toc31426)

[A.2 对称量化方式（Symmetric，SYM） 47](#_Toc12104)

[附录B（资料性附录） C语言参考定义 48](#_Toc13111)

[B.1 数据结构 48](#_Toc27218)

[B.2 机器学习操作 48](#_Toc17915)

 前  言

本文件按照GB/T　1.1-2020《标准化工作导则　第1部分：标准化文件的结构和起草规则》的规定起草。

本文件为T/AI XXXX《人工智能 算子接口》的第3部分。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件由新一代人工智能产业技术创新战略联盟AI标准工作组提出。

本文件由中关村视听产业技术创新联盟归口。

本文件起草单位：北京大学 、北京大学长沙计算与数字经济研究院、鹏城实验室、中国科学院软件研究所、深圳市海思半导体有限公司、北京百度网讯科技有限公司。

本文件起草人：杨超、熬玉龙、黎子毅、李雨芮、李克森、范睿博、段炼、勾海鹏、张军、刘益群、洪明、胡晓光、李笑如、樊春、马银萍、赵海英、杨宏辉、李若淼、付振新。

引  言

人工智能算子是构建人工智能应用的基础运算，是相关硬件操作的封装，人工智能软件通过调用算子接口来使用硬件资源完成计算，算子接口是人工智能软硬件衔接的桥梁。制定T/AI XXXXX《人工智能 算子接口》，是对人工智能算子的核心数据结构、功能和接口参数的规范化和标准化，是降低人工智能软硬件适配难度、促进生态建设的基础性工作。

T/AI XXXXX《人工智能 算子接口》拟由六个部分构成。

——第1部分：基础数学类算子。目的在于确立适用于人工智能算子接口的总则与核心数据结构，以及规范基础数学类算子接口的基本功能和参数的要求。

——第2部分：神经网络类算子。目的在于规范神经网络类算子的基本功能和参数的要求。

——第3部分：机器学习类算子。目的在于规范机器学习类算子的基本功能和参数的要求。

——第4部分：内核开发接口。目的在于规范内核开发接口的基本功能和参数要求。

——第5部分：自动化测试框架。目的在于为算子接口提供规范的参考实现与自动化的测试方法。

——第6部分：大模型类算子。目的在于规范大模型类算子的基本功能和参数要求。

人工智能 算子接口 第3部分：机器学习类

1. 范围

本文件规定了面向人工智能领域的机器学习类算子接口的基本功能和参数的要求。

本文件适用于人工智能算子库的设计、开发与应用，也可用于指导人工智能领域计算框架与算子库和芯片的系统集成与开发。

1. 规范性引用文件

下列文件对于本文件的应用是必不可少的。凡是注日期的引用文件，仅注日期的版本适用于本文件。凡是不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

T/AI XXXX.1—XXXX《人工智能 算子接口 第1部分：基础数学操作》

1. 术语定义

T/AI XXXX.1—XXXX《人工智能 算子接口 第1部分：基础数学操作》界定的适用于本文件。

1. 缩略语

下列缩略语适用于本文件。

KNN:K-最近邻(k-Nearest Neighbor)

SVM：支持向量机（Support Vector Machine）

DT：决策树分类(Decision Tree)

RF：随机森林回归(Random Forest)

1. 通则

本章内容应符合T/AI XXXX.1—XXXX《人工智能 算子接口 第1部分：基础数学操作》中第5章要求。

1. 数据结构

本章内容应符合T/AI XXXX.1—XXXX《人工智能 算子接口 第1部分：基础数学操作》中第6章要求。

1. 机器学习类算子接口
	1. 接口列表

本章节主要针对机器学习模块，共提出14类共62个函数接口，详见表1：

表 1 机器学习函数接口列表

|  |  |
| --- | --- |
| 类别 | 名称 |
| K-最近邻算法（KNN） | 创建KNN回归模型，创建KNN分类模型，训练KNN模型，使用KNN模型进行预测，销毁KNN模型。 |
| 支持向量机算法（SVM） | 创建C-SVC分类模型，创建nu-SVC分类模型，创建one-class SVM模型，创建epsilon-SVR模型，创建nu-SVR模型，训练SVM模型，使用SVM模型进行预测，销毁SVM模型。 |
| 线性回归算法 | 创建线性回归模型，训练线性回归模型，使用线性回归模型进行预测，销毁线性回归模型。 |
| 逻辑回归算法 | 创建逻辑回归模型，训练逻辑回归模型，使用逻辑回归模型进行预测，销毁逻辑回归模型。 |
| 模型评估 | 均方误差，最大误差，R2得分，正确性得分 |
| 主成分分析 | 创建主成分分析模型，训练主成分分析模型，使用主成分分析模型进行降维，销毁主成分分析模型 |
| 线性判别分析 | 创建线性判别分析模型，训练线性判别分析模型，使用线性判别分析模型进行降维，使用线性判别模型进行分类，销毁线性判别模型 |
| 高斯朴素贝叶斯 | 创建高斯朴素贝叶斯模型，训练高斯朴素贝叶斯模型，使用高斯朴素贝叶斯模型进行预测，销毁高斯朴素贝叶斯模型 |
| 决策树分类模型(DT Classifier ) | 创建DT Classifier模型，训练DT Classifier模型，使用DT Classifier模型进行分类，销毁DT Classifier模型。 |
| 决策树回归模型(DT Regressor) | 创建DT Regressor模型，训练DT Regressor模型，使用DT Regressor模型进行回归，销毁DT Regressor模型。 |
| 随机森林分类模型(RF Classifier) | 创建RF Classifier模型，训练RF Classifier模型，使用RF Classifier模型进行分类，销毁RF Classifier模型。 |
| 随机森林回归模型(RF Regressor) | 创建RF Regressor模型，训练RF Regressor模型，使用RF Regressor模型进行回归，销毁RF Regressor模型。 |
| 自适应提升分类模型(AdaBoost Classifier) | 创建AdaBoost Classifier模型，训练AdaBoost Classifier模型，使用AdaBoost Classifier模型进行分类，销毁AdaBoost Classifier模型。 |
| 自适应提升回归模型(AdaBoost Regressor） | 创建AdaBoost Regressor模型，训练AdaBoost Regressor模型，使用AdaBoost Regressor模型进行回归，销毁AdaBoost Regressor模型。 |

* 1. 函数接口列表
		1. K-最近邻算法（KNN）
			1. 创建KNN回归模型
				1. 功能

创建KNN回归模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建KNN回归模型函数前向接口应符合表2，C代码示例见B2.1.1。

表 2 创建KNN回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 邻近点个数 | 输入 | 代表选择的邻近点个数。 |
| 权重计算方式 | 输入 | 枚举类型变量。KNN回归模型在预测时采用的权重计算方式，可以为UNIFORM, DISTANCE等。 |
| 计算邻近点的额外参数 | 输入 | 结构体变量，用于设置KNN采用的计算最邻近点算法类型以及所需要的额外参数。 |
| 计算距离矩阵的额外参数 | 输入 | 结构体变量，用于设置KNN采用的计算距离矩阵算法类型以及所需要的额外参数。 |
| 模型 | 输出 | 新创建的KNN回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 创建KNN分类模型
				1. 功能

创建KNN分类模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建KNN分类模型函数前向接口应符合表3，C代码示例见B2.1.2。

表 3 创建KNN分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 邻近点个数 | 输入 | 代表选择的邻近点个数。 |
| 权重计算方式 | 输入 | 枚举类型变量。KNN回归模型在预测时采用的权重计算方式，可以为UNIFORM, DISTANCE等。 |
| 计算邻近点的额外参数 | 输入 | 结构体变量，用于设置KNN采用的计算最邻近点算法类型以及所需要的额外参数。 |
| 计算距离矩阵的额外参数 | 输入 | 结构体变量，用于设置KNN采用的计算距离矩阵算法类型以及所需要的额外参数。 |
| 模型 | 输出 | 新创建的KNN分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 训练KNN模型
				1. 功能

KNN算法的训练过程，根据KNN模型中的设置参数，选用对应的算法进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练KNN模型函数前向接口应符合表4，C代码示例见B2.1.3。

表 4 训练KNN模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型可以为FLOAT32, FLOAT64。 |
| 训练集数据标签张量 | 输入 | 表示输入的训练集数据标签张量。其形状为[num\_vects], 元素类型可以为UINT8, INT8, UINT16, INT16, UINT32, INT32, UINT64, INT64, FLOAT32, FLOAT64。其中浮点类型对应于knn\_regressor类型，整数类型对应于knn\_classifier类型 |
| 模型 | 输入输出 | KNN模型。 |

* + - * 1. 接口返回值

没有错误: 表示训练成功。

参数错误: 表示参数出错。

* + - 1. 使用KNN模型进行预测
				1. 功能

KNN算法的预测过程，利用KNN模型中的设定参数和训练得到的参数，选用对应的算法进行预测。

* + - * 1. 接口参数

使用KNN模型进行预测函数前向接口应符合表5，C代码示例见B2.1.4。

表 5 使用KNN模型进行预测参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 测试集特征张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre], 元素类型可以为FLOAT32, FLOAT64。 |
| 模型 | 输入 | KNN模型。 |
| 预测结果张量 | 输出 | 表示输出的预测结果张量，其形状为[num\_vects\_pre], 元素类型可以为UINT8, INT8, UINT16, INT16, UINT32, INT32, UINT64, INT64, FLOAT32, FLOAT64。其数据类型应与label保持一致。 |

* + - * 1. 接口返回值

没有错误: 表示预测成功。

参数错误: 表示参数出错。

* + - 1. 销毁KNN模型
				1. 功能

销毁某个指定的KNN模型。

* + - * 1. 接口参数

销毁KNN模型函数前向接口应符合表6，C代码示例见B2.1.5。

表 6 销毁KNN模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入输出 | 待销毁的KNN模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功销毁KNN模型。

参数错误: 表示参数出错。

* + - 1. 保存KNN回归模型
				1. 功能

保存KNN回归模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存KNN回归模型函数前向接口应符合表7。

表 7 保存KNN回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| KNN回归模型 | 输入 | 表示已经训练好的KNN回归模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载KNN回归模型
				1. 功能

加载一个保存好的KNN回归模型，用于后续预测任务中。

* + - * 1. 接口参数

加载KNN回归模型函数前向接口应符合表8。

表 8 加载KNN回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| KNN回归模型 | 输入输出 | 表示存储加载结果的KNN回归模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

* + - 1. 保存KNN分类模型
				1. 功能

保存KNN分类模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存KNN分类模型函数前向接口应符合表9。

表 9 保存KNN分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| KNN分类模型 | 输入 | 表示已经训练好的KNN分类模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载KNN分类模型
				1. 功能

加载一个保存好的KNN分类模型，用于后续预测任务中。

* + - * 1. 接口参数

加载KNN分类模型函数前向接口应符合表10。

表 10 加载KNN分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| KNN分类模型 | 输入输出 | 表示存储加载结果的KNN分类模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

* + 1. 支持向量机算法（SVM）
			1. 创建C-SVC分类模型
				1. 功能

创建C-SVC分类模型，用于保存用户传入且设定的模型参数，以及保存后续训练过程中生成的模型参数。

* + - * 1. 接口参数

加载C-SVC分类模型函数前向接口应符合表11，C代码示例见B2.2.1。

表 11 创建C-SVC分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| C-SVC核参数 | 输入 | 用于设置SVM算法采用的核方法类型以及所需要的额外参数。 |
| 惩罚参数 | 输入 | 软间隔的惩罚参数 |
| 精度 | 输入 | 终止条件 |
| C-SVC模型 | 输出 | 新创建的C-SVC分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 创建nu-SVC分类模型
				1. 功能

创建nu-SVC分类模型，用于保存用户传入且设定的模型参数，以及保存后续训练过程中生成的模型参数。

* + - * 1. 接口参数

创建nu-SVC分类模型函数前向接口应符合表12，C代码示例见B2.2.2。

表 12 创建nu-SVC分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| nu-SVC核参数 | 输入 | 用于设置nu-SVC算法采用的核方法类型以及所需要的额外参数。 |
| nu参数 | 输入 | 控制参数nu。 |
| 精度 | 输入 | 终止条件 |
| nu-SVC模型 | 输出 | 新创建的nu-SVC分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 创建one-class SVM模型
				1. 功能

创建one-class SVM分类模型，用于保存用户传入且设定的模型参数，以及保存后续训练过程中生成的模型参数。

* + - * 1. 接口参数

创建one-class SVM分类模型函数前向接口应符合表13，C代码示例见B2.2.3。

表 13 创建one-class-SVC模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| one-class SVM核参数 | 输入 | 用于设置one-class SVM算法采用的核方法类型以及所需要的额外参数。 |
| nu参数 | 输入 | 控制参数nu。 |
| 精度 | 输入 | 终止条件 |
| one-class SVM模型 | 输出 | 新创建的one-class SVM分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 创建epsilon-SVR回归模型
				1. 功能

创建epsilon-SVR回归模型，用于保存用户传入且设定的模型参数，以及保存后续训练过程中生成的模型参数。

* + - * 1. 接口参数

创建epsilon-SVR回归模型函数前向接口应符合表14，C代码示例见B2.2.4。

表 14 创建epsilon-SVC回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| epsilon-SVR核参数 | 输入 | 用于设置SVM算法采用的核方法类型以及所需要的额外参数。 |
| 惩罚参数 | 输入 | 软间隔的惩罚参数 |
| 精度 | 输入 | 终止条件 |
| epsilon-SVR模型 | 输出 | 新创建的epsilon-SVR回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 创建nu-SVR回归模型
				1. 功能

创建nu-SVR回归模型，用于保存用户传入且设定的模型参数，以及保存后续训练过程中生成的模型参数。

* + - * 1. 接口参数

创建nu-SVR回归模型函数前向接口应符合表15，C代码示例见B2.2.5。

表 15 创建nu-SVR回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| nu-SVR SVM核参数 | 输入 | 用于设置nu-SVR算法采用的核方法类型以及所需要的额外参数。 |
| 惩罚参数 | 输入 | 软间隔的惩罚参数 |
| nu参数 | 输入 | 控制参数nu。 |
| 精度 | 输入 | 终止条件 |
| nu-SVRSVM模型 | 输出 | 新创建的nu-SVR SVM回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 训练SVM模型
				1. 功能

SVM算法的训练过程，根据SVM模型中的设置参数，选用对应的算法进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练SVM模型函数前向接口应符合表16，C代码示例见B2.2.6。

表 16 训练SVM模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 输入张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型为浮点类型 |
| 输入张量 | 输入 | 表示输入的训练集数据标签张量。 |
| 模型 | 输入输出 | SVM模型。 |

* + - * 1. 接口返回值

没有错误: 表示训练成功。

参数错误: 表示参数出错。

* + - 1. 使用SVM模型进行预测
				1. 功能

SVM算法的预测过程，根据SVM模型中的设定参数和训练得到的参数，选用对应的算法进行预测。

* + - * 1. 接口参数

使用SVM模型进行预测函数前向接口应符合表17，C代码示例见B2.2.7。

表 17 使用SVM模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 输入张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre], 元素类型为浮点类型。 |
| 模型 | 输入 | SVM模型 |
| 输出张量 | 输入输出 | 表示输出的预测结果张量，其形状为[num\_vects\_pre], |

* + - * 1. 接口返回值

没有错误: 表示训练成功。

参数错误: 表示参数出错。

* + - 1. 销毁SVM模型
				1. 功能

销毁某个指定的SVM模型。

* + - * 1. 接口参数

销毁SVM模型函数前向接口应符合表18，C代码示例见B2.2.8。

表 18 销毁SVM模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入输出 | 待销毁的SVM模型。 |

* + - * 1. 接口返回值

没有错误: 表示训练成功。

参数错误: 表示参数出错。

* + - 1. 保存SVM模型
				1. 功能

保存SVM回归模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存SVM回归模型函数前向接口应符合表19。

表 19 保存SVM回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| SVM模型 | 输入 | 表示已经训练好的SVM模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 加载SVM模型
				1. 功能

加载一个保存好的KNN回归模型，用于后续预测任务中。

* + - * 1. 接口参数

加载SVM回归模型函数前向接口应符合表20。

表 20 加载SVM回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| SVM模型 | 输入输出 | 表示存储加载结果的SVM模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

* + 1. 线性回归算法（Linear Regression）
			1. 创建线性回归模型
				1. 功能

创建线性回归模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建线性回归模型函数前向接口应符合表21，C代码示例见B2.3.1。

表 21 创建线性回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 正则化信息 | 输入 | 包括用于设置线性回归算法采用的正则化类型和值 |
| 线性回归模型 | 输出 | 新创建的线性回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 训练线性回归模型
				1. 功能

线性回归算法的训练过程，根据线性回归模型中的设置参数，选用对应的算法进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练线性回归模型函数前向接口应符合表22，C代码示例见B2.3.2。

表 22 训练线性回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 输入张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型为浮点类型 |
| 输入张量 | 输入 | 表示输入的训练集数据标签张量。 |
| 模型 | 输入输出 | 线性回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

* + - 1. 使用线性回归模型进行预测
				1. 功能

线性回归算法的预测过程，利用线性回归模型中的设定参数和训练得到的参数，选用对应的算法进行预测。

* + - * 1. 接口参数

使用线性回归模型进行预测函数前向接口应符合表23，C代码示例见B2.3.3。

表 23 使用线性回归模型预测参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 输入张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre], 元素类型为浮点类型。 |
| 模型 | 输入 | 线性回归模型 |
| 输出张量 | 输入输出 | 表示输出的预测结果张量，其形状为[num\_vects\_pre], |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

* + - 1. 销毁线性回归模型
				1. 功能

销毁某个指定的线性回归模型。

* + - * 1. 接口参数

销毁线性回归模型函数前向接口应符合表24，C代码示例见B2.3.4。

表 24 销毁线性回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入输出 | 待销毁的线性回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示销毁成功。

参数错误: 表示参数出错。

* + - 1. 保存线性回归模型
				1. 功能

保存线性回归模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存线性回归模型函数前向接口应符合表25。

表 25 保存线性回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 线性回归模型 | 输入 | 表示已经训练好的线性回归模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载线性回归模型
				1. 功能

加载一个保存好的线性回归模型，用于后续预测任务中。

* + - * 1. 接口参数

加载线性回归模型函数前向接口应符合表26。

表 26 加载线性回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| 线性回归模型 | 输入输出 | 表示存储加载结果的线性回归模型 |

* + - * 1. 接口返回值

没有错误: 表示销毁成功。

参数错误: 表示参数出错。

* + 1. 逻辑回归算法（Logistic Regression）
			1. 创建逻辑回归模型
				1. 功能

创建逻辑回归模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建逻辑回归模型函数前向接口应符合表27，C代码示例见B2.4.1。

表 27 创建逻辑回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 求解器器信息 | 输入 | 包括用于设置逻辑回归算法采用的求解器的类型和求解器的参数 |
| 正则化信息 | 输入 | 包括用于设置逻辑回归算法采用的正则化类型和值 |
| 逻辑回归模型 | 输出 | 新创建的逻辑回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 训练逻辑回归模型
				1. 功能

逻辑回归算法的训练过程，根据逻辑回归模型中的设置参数，选用对应的算法进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练逻辑回归模型函数前向接口应符合表28，C代码示例见B2.4.2。

表 28 训练逻辑回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 输入张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型为浮点类型 |
| 输入张量 | 输入 | 表示输入的训练集数据标签张量。 |
| 模型 | 输入输出 | 逻辑回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功训练模型。

非法参数: 表示参数出错。

* + - 1. 使用逻辑回归模型进行预测
				1. 功能

逻辑回归算法的预测过程，利用逻辑回归模型中的设定参数和训练得到的参数，选用对应的算法进行预测。

* + - * 1. 接口参数

使用逻辑回归模型预测函数前向接口应符合表29，C代码示例见B2.4.3。

表 29 使用逻辑回归模型预测参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 输入张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre], 元素类型为浮点类型。 |
| 模型 | 输入 | 逻辑回归模型 |
| 输出张量 | 输入输出 | 表示输出的预测结果张量，其形状为[num\_vects\_pre], |

* + - * 1. 接口返回值

没有错误: 表示预测成功。

参数错误: 表示参数出错。

* + - 1. 销毁逻辑回归模型
				1. 功能

销毁某个指定的逻辑回归模型。

* + - * 1. 接口参数

销毁逻辑回归模型函数前向接口应符合表30，C代码示例见B2.4.4。

表 30 销毁逻辑回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入输出 | 待销毁的逻辑回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示销毁成功。

参数错误: 表示参数出错。

* + - 1. 保存逻辑回归模型
				1. 功能

保存逻辑回归模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存逻辑回归模型函数前向接口应符合表31。

表 31 保存辑回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 逻辑回归模型 | 输入 | 表示已经训练好的逻辑回归模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载逻辑回归模型
				1. 功能

加载一个保存好的逻辑回归模型，用于后续预测任务中。

* + - * 1. 接口参数

加载逻辑回归模型函数前向接口应符合表32。

表 32 加载逻辑回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| 逻辑回归模型 | 输入输出 | 表示存储加载结果的逻辑回归模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

* + 1. 模型评估（Model Evaluation）
			1. 均方误差
				1. 功能

计算真实值和预测值的均方误差。见式（1）：

 （1）

式中：

——第一个输入张量；

——第二个输入张量；

——输出张量。

* + - * 1. 接口参数

均方误差函数前向接口应符合表33，C代码示例见B2.5.1。

表 33 均方误差参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 真实值 | 输入 | 表示实际值，一般形状为[num\_sample, 1] |
| 预测值 | 输入 | 表示使用模型预测的结果，一般形状为[num\_sample, 1] |
| 误差结果 | 输出 | 均方误差 |

* + - * 1. 接口返回值

没有错误: 表示成功计算。

非法参数: 表示参数出错。

* + - 1. 最大误差
				1. 功能

计算真实值和预测值的最大误差，见式（2）：

 （2）

式中：

——第一个输入张量；

——第二个输入张量；

——取最大值；

——输出张量。

* + - * 1. 接口参数

最大误差函数前向接口应符合表34，C代码示例见B2.5.2。

表 34 最大误差参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 真实值 | 输入 | 表示实际值，一般形状为[num\_sample, 1] |
| 预测值 | 输入 | 表示使用模型预测的结果，一般形状为[num\_sample, 1] |
| 误差结果 | 输出 | 计算得到的最大误差 |

* + - * 1. 接口返回值

没有错误: 表示成功计算。

非法参数: 表示参数出错。

* + - 1. R2得分
				1. 功能

计算回归的R2得分，均方得分越趋近于1表示模型表现越好，见式（3）：

 （3）

式中：

——第一个输入张量；

——第二个输入张量；

——第一个输入张量取平均值；

——输出张量。

* + - * 1. 接口参数

R2得分函数前向接口应符合表35，C代码示例见B2.5.3。

表 35 R2得分参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 真实值 | 输入 | 表示实际值，一般形状为[num\_sample, 1] |
| 预测值 | 输入 | 表示使用模型预测的结果，一般形状为[num\_sample, 1] |
| 均方得分 | 输出 | 计算得到的均方得分 |

* + - * 1. 接口返回值

没有错误: 表示成功计算。

参数错误: 表示参数出错。

* + - 1. 正确性得分
				1. 功能

计算分类的正确性，计算正确的结果和预测的结果之间差别的个数，以此计算正确性，见式（4）：

 （4）

式中：

——第一个输入张量；

——第二个输入张量；

——输出张量。

* + - * 1. 接口参数

正确性得分函数前向接口应符合表36，C代码示例见B2.5.4。

表 36 正确性得分参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 真实值 | 输入 | 表示实际值，一般形状为[num\_sample, 1] |
| 预测值 | 输入 | 表示使用模型预测的结果，一般形状为[num\_sample, 1] |
| 正确性结果 | 输出 | 计算得到正确性得分 |

* + - * 1. 接口返回值

没有错误: 表示成功计算。

参数错误: 表示参数出错。

* + 1. 主成分分析算法 (Pricipal Component Analysis)
			1. 创建主成分分析模型
				1. 功能

创建PCA主成分分析模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建PCA主成分分析模型函数前向接口应符合表37，C代码示例见B2.6.1。

表 37 创建PCA主成分分析模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 主成分个数 | 输入 | 代表要保留的主成分个数。 |
| 奇异值分解求解器 | 输入 | 结构体。PCA模型在计算时对协方差矩阵进行SVD分解的求解器类型以及相关的参数。 |
| 模型 | 输出 | 新创建的PCA主成分分析模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 训练PCA主成分分分析模型
				1. 功能

PCA算法的训练过程，根据PCA模型中的设置参数，选用对应的算法进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练PCA主成分分析模型函数前向接口应符合表37，C代码示例见B2.6.2。

表 38 训练PCA模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型可以为FLOAT32, FLOAT64。 |
| 模型 | 输入输出 | PCA模型。 |

* + - * 1. 接口返回值

没有错误: 表示训练成功。

参数错误: 表示参数出错。

* + - 1. 使用主成分分析模型进行降维
				1. 功能

PCA算法降维过程。利用PCA模型中训练得到的参数，对数据集进行特征降维。

* + - * 1. 接口参数

使用主成分分析模型进行降维模型函数前向接口应符合表39，C代码示例见B2.6.3。

表 39 使用PCA主成分分析模型进行降维参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 数据集特征张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects, dim\_vects], 元素类型可以为FLOAT32, FLOAT64。 |
| 模型 | 输入 | PCA模型。 |
| 预测结果张量 | 输出 | 表示数据集的降维后的特征张量，其形状为[num\_vects, dim\_components], 元素类型可以为FLOAT32, FLOAT64。 |

* + - * 1. 接口返回值

没有错误: 表示降维成功。

参数错误: 表示参数出错。

* + - 1. 销毁主成分分析模型
				1. 功能

销毁某个指定的PCA模型。

* + - * 1. 接口参数

销毁主成分分析模型函数前向接口应符合表40，C代码示例见B2.6.4。

表 40 销毁PCA模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入输出 | 待销毁的PCA模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功销毁PCA模型。

参数错误: 表示参数出错。

* + - 1. 保存PCA模型
				1. 功能

保存PCA模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存PCA模型函数前向接口应符合表41。

表 41 保存PCA模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| PCA模型 | 输入 | 表示已经训练好的PCA模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载PCA模型
				1. 功能

加载一个保存好的PCA模型，用于后续预测任务中。

* + - * 1. 接口参数

加载PCA模型函数前向接口应符合表42。

表 42 加载PCA模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| PCA模型 | 输入输出 | 表示存储加载结果的PCA模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

* + 1. 线性判别分析算法 (Linear Discriminant Analysis)
			1. 创建线性判别分析模型
				1. 功能

创建LDA线性判别分析模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建LDA线性判别分析模型函数前向接口应符合表43，C代码示例见B2.7.1。

表 43 创建LDA线性判别分析模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 降维维数 | 输入 | 代表LDA降维后的维度数。 |
| 求解器信息 | 输入 | 对LDA模型进行求解的求解器类型和求解器的参数 |
| 模型 | 输出 | 新创建的LDA线性判别分析模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 训练线性判别分析模型
				1. 功能

LDA算法的训练过程，根据LDA模型中的设置参数，选用对应的算法进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练LDA线性判别分析模型函数前向接口应符合表44，C代码示例见B2.7.2。

表 44 训练LDA线性判别分析模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型可以为FLOAT32, FLOAT64。 |
| 训练集标签张量 | 输入 | 表示输入的训练集标签张量。其形状为[num\_vects], 元素类型可以为INT32。 |
| 模型 | 输入输出 | LDA模型。 |

* + - * 1. 接口返回值

没有错误: 表示训练成功。

参数错误: 表示参数出错。

* + - 1. 使用线性判别分析模型进行降维
				1. 功能

LDA算法降维过程。利用LDA模型中训练得到的参数，对数据集进行特征降维。

* + - * 1. 接口参数

使用LDA线性判别分析模型进行降维函数前向接口应符合表45，C代码示例见B2.7.3。

表45 使用LDA线性判别分析模型进行降维参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 数据集特征张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects, dim\_vects], 元素类型可以为FLOAT32, FLOAT64。 |
| 模型 | 输入 | LDA模型。 |
| 预测结果张量 | 输出 | 表示数据集的降维后的特征张量，其形状为[num\_vects, dim\_components], 元素类型可以为FLOAT32, FLOAT64。 |

* + - * 1. 接口返回值

没有错误: 表示降维成功。

参数错误: 表示参数出错。

* + - 1. 使用线性判别分析模型进行分类
				1. 功能

LDA算法的分类过程，利用逻LDA模型中的设定参数和训练得到的参数，选用对应的算法进行分类。

* + - * 1. 接口参数

使用LDA线性判别分析模型进行分类函数前向接口应符合表46，C代码示例见B2.7.4。

表 46 使用LDA模型分类参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 输入张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre], 元素类型为浮点类型。 |
| 模型 | 输入 | LDA模型 |
| 输出张量 | 输入输出 | 表示输出的预测结果张量，其形状为[num\_vects\_pre], |

* + - * 1. 接口返回值

没有错误: 表示预测成功。

参数错误: 表示参数出错。

* + - 1. 销毁线性判别分析模型
				1. 功能

销毁某个指定的LDA模型。

* + - * 1. 接口参数

销毁LDA线性判别分析模型函数前向接口应符合表47，C代码示例见B2.7.5。

表 47 销毁LDA模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入输出 | 待销毁的LDA模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功销毁LDA模型。

参数错误: 表示参数出错。

* + - 1. 保存LDA模型
				1. 功能

保存LDA模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存LDA线性判别分析模型函数前向接口应符合表48。

表 48 保存LDA模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| LDA模型 | 输入 | 表示已经训练好的LDA模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载LDA模型
				1. 功能

加载一个保存好的LDA模型，用于后续预测任务中。

* + - * 1. 接口参数

加载LDA线性判别分析模型函数前向接口应符合表49。

表 49 加载LDA模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| LDA模型 | 输入输出 | 表示存储加载结果的LDA模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

* + 1. 高斯朴素贝叶斯算法(Gaussian Naïve Bayesian)
			1. 创建高斯朴素贝叶斯模型
				1. 功能

创建高斯朴素贝叶斯模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建高斯朴素贝叶斯模型函数前向接口应符合表50，C代码示例见B2.8.1。

表 50 创建高斯朴素贝叶斯模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 先验概率 | 输入 | 浮点数数组，表示每一类的先验概率。 |
| 平滑参数 | 输入 | 为了计算的稳定性采用的平滑参数 |
| 高斯朴素贝叶斯模型 | 输出 | 新创建的高斯朴素贝叶斯模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 训练高斯朴素贝叶斯模型
				1. 功能

高斯朴素贝叶斯算法的训练过程，根据高斯朴素贝叶斯模型中的设置参数，对模型进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练高斯朴素贝叶斯模型函数前向接口应符合表51，C代码示例见B2.8.2。

表 51 训练高斯朴素贝叶斯模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 输入张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型为浮点类型 |
| 输入张量 | 输入 | 表示输入的训练集数据标签张量。 |
| 模型 | 输入输出 | 逻辑高斯朴素贝叶斯模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功训练模型。

非法参数: 表示参数出错。

* + - 1. 使用高斯朴素贝叶斯模型进行预测
				1. 功能

高斯朴素贝叶斯算法的预测过程，利用高斯朴素贝叶斯模型训练得到的参数，选用对应的算法进行预测。

* + - * 1. 接口参数

使用高斯朴素贝叶斯模型进行预测模型函数前向接口应符合表52，C代码示例见B2.8.3。

表 52 使用逻辑回归模型预测参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 输入张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre], 元素类型为浮点类型。 |
| 模型 | 输入 | 高斯朴素贝叶斯模型 |
| 输出张量 | 输入输出 | 表示输出的预测结果张量，其形状为[num\_vects\_pre], |

* + - * 1. 接口返回值

没有错误: 表示预测成功。

参数错误: 表示参数出错。

* + - 1. 销毁高斯朴素贝叶斯模型
				1. 功能

销毁某个指定的高斯朴素贝叶斯模型。

* + - * 1. 接口参数

销毁高斯朴素贝叶斯模型模型函数前向接口应符合表53，C代码示例见B2.8.4。

表 53 销毁高斯朴素贝叶斯模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入输出 | 待销毁的高斯朴素贝叶斯模型。 |

* + - * 1. 接口返回值

没有错误: 表示销毁成功。

参数错误: 表示参数出错。

* + - 1. 保存GNB模型
				1. 功能

保存GNB模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存GNB模型函数前向接口应符合表54。

表 54 保存GNB模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| GNB模型 | 输入 | 表示已经训练好的GNB模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载GNB模型
				1. 功能

加载一个保存好的GNB模型，用于后续预测任务中。

* + - * 1. 接口参数

加载GNB模型函数前向接口应符合表55。

表 55 加载GNB模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| GNB模型 | 输入输出 | 表示存储加载结果的GNB模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

* + 1. 决策树分类模型 (Decision Tree Classifier)
			1. 创建决策树分类器模型
				1. 功能

创建决策树分类模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建决策树分类模型函数前向接口应符合表56，C代码示例见B2.9.1。

表 56 创建决策树分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 特征选择准则 | 输入 | 字符串，表示决策树生成过程中进行特征选择的标准。可取值：gini, entropy。 |
| 最大深度 | 输入 | 整数。表示生成决策树的最大深度。 |
| 最小分裂点样本数 | 输入 | 整数，表示决策树生成过程中在一个点进行分裂操作时这个点所需要包括的最小样本数。 |
| 最小叶节点样本数 | 输入 | 整数，表示生成叶节点所需的最小样本数。 |
| 决策树分类器模型 | 输出 | 新创建的决策树分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 训练决策树分类模型
				1. 功能

决策树分类模型的训练过程，根据决策树分类模型中的设置参数，对模型进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练决策树分类模型函数前向接口应符合表57，C代码示例见B2.9.2。

表 57 训练决策树分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型为浮点类型。 |
| 训练集数据标签张量 | 输入 | 表示输入的训练集数据标签张量。 |
| 模型 | 输入输出 | 表示决策树分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示训练成功。

参数错误: 表示参数出错。

* + - 1. 使用决策树分类模型进行分类
				1. 功能

决策树分类模型的预测过程，利用决策树分类模型训练得到的参数，选用对应的算法进行预测。

* + - * 1. 接口参数

使用决策树分类模型进行分类模型函数前向接口应符合表58，C代码示例见B2.9.3。

表 58 使用决策树分类模型进行分类参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre], 元素类型为浮点类型。 |
| 模型 | 输入 | 决策树分类模型。 |
| 分类结果张量 | 输入输出 | 表示输出的分类结果张量，其形状为[num\_vects\_pre]。 |

* + - * 1. 接口返回值

没有错误: 表示预测成功。

参数错误: 表示参数出错。

* + - 1. 销毁决策树分类模型
				1. 功能

销毁某个指定的决策树分类模型。

* + - * 1. 接口参数

销毁决策树分类模型函数前向接口应符合表59，C代码示例见B2.9.4。

表 59 销毁决策树分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入 | 待销毁的决策树分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功销毁决策树分类模型。

参数错误: 表示参数出错。

* + - 1. 保存决策树分类模型
				1. 功能

保存决策树分类模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存决策树分类模型函数前向接口应符合表60。

表 60 保存决策树分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 决策树分类模型 | 输入 | 表示已经训练好的决策树分类模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载决策树分类模型
				1. 功能

加载一个保存好的决策树分类模型，用于后续预测任务中。

* + - * 1. 接口参数

加载决策树分类模型函数前向接口应符合表61。

表 61 加载决策树分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 参数名 | 类型 | 描述 |
| 决策树分类模型 | 输入输出 | 表示存储加载结果的决策树分类模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

* + 1. 决策树回归模型 (Decision Tree Regressor)
			1. 创建决策树回归模型
				1. 功能

创建决策树回归模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建决策树回归模型函数前向接口应符合表62，C代码示例见B2.10.1。

表 62 创建决策树回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 节点分裂度量函数 | 输入 | 字符串，表示决策树回归过程中对节点分裂质量的度量函数。可取值：mse, friedman\_mse, mae, poisson。 |
| 最大深度 | 输入 | 整数。表示生成决策树的最大深度。 |
| 最小分裂点样本数 | 输入 | 整数，表示决策树生成过程中在一个点进行分裂操作时这个点所需要包括的最小样本数。 |
| 最小叶节点样本数 | 输入 | 整数，表示生成叶节点所需的最小样本数。 |
| 决策树回归器模型 | 输出 | 新创建的决策树回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 训练决策树回归模型
				1. 功能

决策树回归模型的训练过程，根据决策树回归模型中的设置参数，选用对应的算法进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练决策树回归模型函数前向接口应符合表63，C代码示例见B2.10.2。

表 63 训练决策树回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型为浮点类型。 |
| 训练集数据标签张量 | 输入 | 表示输入的训练集数据标签张量。 |
| 模型 | 输入输出 | 表示决策树回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示训练成功。

参数错误: 表示参数出错。

* + - 1. 使用决策树回归模型进行回归
				1. 功能

决策树回归过程。利用决策树回归模型中训练得到的参数，对数据集进行回归运算。

* + - * 1. 接口参数

使用决策树回归模型进行回归函数前向接口应符合表64，C代码示例见B2.10.3。

表 64 使用决策树回归模型进行回归参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre], 元素类型为浮点类型。 |
| 模型 | 输入 | 决策树回归模型。 |
| 回归结果张量 | 输出 | 表示对数据集进行回归的结果张量，其形状为[num\_vects\_pre]。 |

* + - * 1. 接口返回值

没有错误: 表示降维成功。

参数错误: 表示参数出错。

* + - 1. 销毁决策树回归模型
				1. 功能

销毁某个指定的决策树回归模型。

* + - * 1. 接口参数

销毁决策树回归模型函数前向接口应符合表65，C代码示例见B2.10.4。

表 65 销毁决策树回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入 | 待销毁的决策树回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功销毁决策树回归模型。

参数错误: 表示参数出错。

* + - 1. 保存决策树回归模型
				1. 功能

保存决策树回归模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存决策树回归模型函数前向接口应符合表66。

表 66 保存决策树回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 决策树回归模型 | 输入 | 表示已经训练好的决策树回归模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载决策树回归模型
				1. 功能

加载一个保存好的决策树回归模型，用于后续预测任务中。

* + - * 1. 接口参数

加载决策树回归模型函数前向接口应符合表67。

表 67 加载决策树回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| 决策树回归模型 | 输入输出 | 表示存储加载结果的决策树回归模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

* + 1. 随机森林分类模型 (Random Forest Classifier)
			1. 创建随机森林分类模型
				1. 功能

创建随机森林分类模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建随机森林分类模型函数前向接口应符合表68，C代码示例见B2.11.1。

表 68 创建随机森林分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 树的个数 | 输入 | 整数，表示随机森林中树的个数。 |
| 特征选择准则 | 输入 | 字符串，表示随机森林中每一棵决策树生成过程中进行特征选择的标准。可取值：gini, entropy。 |
| 最大深度 | 输入 | 整数。表示随机森林中每一棵决策树的最大深度。 |
| 最小分裂点样本数 | 输入 | 整数，表示随机森林中每一棵决策树生成过程中在一个点进行分裂操作时这个点所需要包括的最小样本数。 |
| 最小叶节点样本数 | 输入 | 整数，表示生成叶节点所需的最小样本数。 |
| 自助法抽样标志 | 输入 | 整数，表示构造树的时候是否采用自助法抽取样本进行训练。 |
| 随机种子 | 输入 | 整数，控制生成树的过程中的随机性。 |
| 随机森林分类器模型 | 输出 | 新创建的随机森林分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 训练随机森林分类模型
				1. 功能

随机森林分类模型的训练过程，根据随机森林分类模型中的设置参数，选用对应的算法进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练随机森林分类模型函数前向接口应符合表69，C代码示例见B2.11.2。

表 69 训练随机森林分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型为浮点类型。 |
| 训练集数据标签张量 | 输入 | 表示输入的训练集数据标签张量。 |
| 模型 | 输入输出 | 表示随机森林分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示训练成功。

参数错误: 表示参数出错。

* + - 1. 使用随机森林分类模型进行分类
				1. 功能

随机森林分类过程。利用随机森林分类模型中训练得到的参数，对数据集进行分类。

* + - * 1. 接口参数

使用随机森林分类模型进行分类函数前向接口应符合表70，C代码示例见B2.11.3。

表 70 使用随机森林分类模型进行分类参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre], 元素类型为浮点类型。 |
| 模型 | 输入 | 随机森林分类模型。 |
| 分类结果张量 | 输出 | 表示对数据集进行分类的结果张量，其形状为[num\_vects\_pre]。 |

* + - * 1. 接口返回值

没有错误: 表示降维成功。

参数错误: 表示参数出错。

* + - 1. 销毁随机森林分类模型
				1. 功能

销毁某个指定的随机森林分类模型。

* + - * 1. 接口参数

销毁随机森林分类模型函数前向接口应符合表71，C代码示例见B2.11.4。

表 71 销毁随机森林分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入 | 待销毁的随机森林分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功销毁随机森林分类模型。

参数错误: 表示参数出错。

* + - 1. 保存随机森林分类模型
				1. 功能

保存随机森林分类模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存随机森林分类模型函数前向接口应符合表72。

表 72 保存随机森林分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 随机森林分类模型 | 输入 | 表示已经训练好的随机森林分类模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载随机森林分类模型
				1. 功能

加载一个保存好的随机森林分类模型，用于后续预测任务中。

* + - * 1. 接口参数

加载随机森林分类模型函数前向接口应符合表73。

表 73 加载随机森林分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| 随机森林分类模型 | 输入输出 | 表示存储加载结果的随机森林分类模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

* + 1. 随机森林回归模型 (Random Forest Regressor)
			1. 创建随机森林回归模型
				1. 功能

创建随机森林回归模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建随机森林回归模型函数前向接口应符合表74，C代码示例见B2.12.1。

表 74 创建随机森林回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 节点分裂度量函数 | 输入 | 字符串，表示决策树回归过程中对节点分裂质量的度量函数。可取值：mse, mae。 |
| 最大深度 | 输入 | 整数。表示生成决策树的最大深度。 |
| 最小分裂点样本数 | 输入 | 整数，表示决策树生成过程中在一个点进行分裂操作时这个点所需要包括的最小样本数。 |
| 最小叶节点样本数 | 输入 | 整数，表示生成叶节点所需的最小样本数。 |
| 自助法抽样标志 | 输入 | 整数，表示构造树的时候是否采用自助法抽取样本进行训练。 |
| 随机种子 | 输入 | 整数，控制生成树的过程中的随机性。 |
| 随机森林回归模型 | 输出 | 新创建的随机森林回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 训练随机森林回归模型
				1. 功能

随机森林回归模型的训练过程，根据随机森林回归模型中的设置参数，选用对应的算法进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练随机森林回归模型函数前向接口应符合表75，C代码示例见B2.12.2。

表 75 训练随机森林回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型为浮点类型。 |
| 训练集数据标签张量 | 输入 | 表示输入的训练集数据标签张量。 |
| 模型 | 输入输出 | 表示随机森林回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示训练成功。

参数错误: 表示参数出错。

* + - 1. 使用随机森林回归模型进行回归
				1. 功能

随机森林回归过程。利用随机森林回归模型中训练得到的参数，对数据集进行回归运算。

* + - * 1. 接口参数

使用随机森林回归模型进行回归函数前向接口应符合表76，C代码示例见B2.12.3。

表 76 使用随机森林回归模型进行回归参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre], 元素类型为浮点类型。 |
| 模型 | 输入 | 随机森林回归模型。 |
| 回归结果张量 | 输出 | 表示对数据集进行回归的结果张量，其形状为[num\_vects\_pre]。 |

* + - * 1. 接口返回值

没有错误: 表示降维成功。

参数错误: 表示参数出错。

* + - 1. 销毁随机森林回归模型
				1. 功能

销毁某个指定的随机森林回归模型。

* + - * 1. 接口参数

销毁随机森林回归模型函数前向接口应符合表77，C代码示例见B2.12.4。

表 77 销毁随机森林回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入 | 待销毁的随机森林回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功销毁随机森林回归模型。

参数错误: 表示参数出错。

* + - 1. 保存随机森林回归模型
				1. 功能

保存随机森林回归模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存随机森林回归模型函数前向接口应符合表78。

表 78 保存随机森林回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 随机森林回归模型 | 输入 | 表示已经训练好的随机森林回归模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载随机森林回归模型
				1. 功能

加载一个保存好的随机森林回归模型，用于后续预测任务中。

* + - * 1. 接口参数

加载随机森林回归模型函数前向接口应符合表79。

表 79 加载随机森林回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| 随机森林回归模型 | 输入输出 | 表示存储加载结果的随机森林回归模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

* + 1. 自适应提升分类模型 (Adaptive Boosting Classifier)
			1. 创建自适应提升分类模型
				1. 功能

创建自适应提升分类模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建自适应提升分类模型函数前向接口应符合表80，C代码示例见B2.13.1。

表 80 创建自适应提升分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 基准分类模型 | 输入 | 分类模型，表示提升方法中使用的基准分类器，缺省值为决策树分类模型。 |
| 最大基准分类器数目 | 输入 | 整数。表示提升方法终止时所能使用的基准分类器的最大数目。 |
| 学习率 | 输入 | 浮点数，表示提升方法每一轮迭代中应用到分类器的权重。 |
| 自适应提升分类模型 | 输出 | 新创建的自适应提升分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 训练自适应提升分类模型
				1. 功能

自适应提升分类的训练过程，根据自适应提升分类中的设置参数，对模型进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练自适应提升分类器模型函数前向接口应符合表81，C代码示例见B2.13.2。

表 81 训练自适应提升分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型为浮点类型。 |
| 训练集数据标签张量 | 输入 | 表示输入的训练集数据标签张量。 |
| 模型 | 输入输出 | 表示自适应提升分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示训练成功。

参数错误: 表示参数出错。

* + - 1. 使用自适应提升分类模型进行分类
				1. 功能

自适应提升分类模型的预测过程，利用自适应提升分类模型训练得到的参数，选用对应的算法进行预测。

* + - * 1. 接口参数

使用自适应提升分类模型进行分类函数前向接口应符合表82，C代码示例见B2.13.3。

表 82 使用自适应提升分类模型进行分类参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre], 元素类型为浮点类型。 |
| 模型 | 输入 | 自适应提升分类模型。 |
| 分类结果张量 | 输入输出 | 表示输出的分类结果张量，其形状为[num\_vects\_pre]。 |

* + - * 1. 接口返回值

没有错误: 表示预测成功。

参数错误: 表示参数出错。

* + - 1. 销毁自适应提升分类模型
				1. 功能

销毁某个指定的自适应提升分类模型。

* + - * 1. 接口参数

销毁自适应提升分类器模型函数前向接口应符合表83，C代码示例见B2.13.4。

表 83 销毁自适应提升分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入 | 待销毁的自适应提升分类模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功销毁自适应提升分类模型。

参数错误: 表示参数出错。

* + - 1. 保存自适应提升分类模型
				1. 功能

保存自适应提升分类模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存自适应提升分类器模型函数前向接口应符合表84。

表 84 保存自适应提升分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 自适应提升分类模型 | 输入 | 表示已经训练好的自适应提升分类模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载自适应提升分类模型
				1. 功能

加载一个保存好的自适应提升分类模型，用于后续预测任务中。

* + - * 1. 接口参数

加载自适应提升分类器模型函数前向接口应符合表85。

表 85加载自适应提升分类模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| 自适应提升分类模型 | 输入输出 | 表示存储加载结果的自适应提升分类模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

* + 1. 自适应提升回归模型 (Adaptive Boosting Regressor)
			1. 创建自适应提升回归模型
				1. 功能

创建自适应提升回归模型，用于保存用户传入的模型参数，以及后续训练完成后得到的模型参数。

* + - * 1. 接口参数

创建自适应提升回归模型函数前向接口应符合表86，C代码示例见B2.14.1。

表 86 创建自适应提升回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 基准回归模型 | 输入 | 回归模型，表示提升方法中使用的基准回归模型，缺省值为决策树回归模型。 |
| 最大基准回归模型数目 | 输入 | 整数。表示提升方法终止时所能使用的基准回归模型的最大数目。 |
| 学习率 | 输入 | 浮点数，表示提升方法每一轮迭代中应用到回归模型的权重。 |
| 自适应提升回归模型 | 输出 | 新创建的自适应提升回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功创建模型。

非法参数: 表示参数出错。

空间错误: 表示创建模型分配空间不足。

* + - 1. 训练自适应提升回归模型
				1. 功能

自适应提升回归的训练过程，根据自适应提升回归中的设置参数，对模型进行训练，并将训练得到的模型参数保存于model结构体。

* + - * 1. 接口参数

训练自适应提升回归模型函数前向接口应符合表87，C代码示例见B2.14.2。

表 87 训练自适应提升回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects], 元素类型为浮点类型。 |
| 训练集数据标签张量 | 输入 | 表示输入的训练集数据标签张量。 |
| 模型 | 输入输出 | 表示自适应提升回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示训练成功。

参数错误: 表示参数出错。

* + - 1. 使用自适应提升回归模型进行回归
				1. 功能

自适应提升回归模型的预测过程，利用自适应提升回归模型训练得到的参数，选用对应的算法进行预测。

* + - * 1. 接口参数

使用自适应提升回归模型进行回归函数前向接口应符合表88，C代码示例见B2.14.3。

表 88 使用自适应提升回归模型进行回归参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 训练集特征张量 | 输入 | 表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre], 元素类型为浮点类型。 |
| 模型 | 输入 | 自适应提升回归模型。 |
| 回归结果张量 | 输出 | 表示对数据集进行回归的结果张量，其形状为[num\_vects\_pre]。 |

* + - * 1. 接口返回值

没有错误: 表示预测成功。

参数错误: 表示参数出错。

* + - 1. 销毁自适应提升回归模型
				1. 功能

销毁某个指定的自适应提升回归模型。

* + - * 1. 接口参数

销毁自适应提升回归模型函数前向接口应符合表89，C代码示例见B2.14.4。

表 89 销毁自适应提升回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型 | 输入 | 待销毁的自适应提升回归模型。 |

* + - * 1. 接口返回值

没有错误: 表示成功销毁自适应提升回归模型。

参数错误: 表示参数出错。

* + - 1. 保存自适应提升回归模型
				1. 功能

保存自适应提升回归模型，用于保存已经训练好的模型，用于后续的模型加载。

* + - * 1. 接口参数

保存自适应提升回归模型函数前向接口应符合表90。

表 90 保存自适应提升回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 自适应提升回归模型 | 输入 | 表示已经训练好的自适应提升回归模型 |
| 模型存储路径 | 输入 | 字符串，表示模型的保存路径 |
| 模型存储名称 | 输入 | 字符串，表示模型保存的文件名 |

* + - * 1. 接口返回值

没有错误: 表示成功保存模型。

非法参数: 表示参数出错。

空间错误: 表示存储模型分配空间不足。

* + - 1. 加载自适应提升回归模型
				1. 功能

加载一个保存好的自适应提升回归模型，用于后续预测任务中。

* + - * 1. 接口参数

加载自适应提升回归模型函数前向接口应符合表91。

表 91 加载自适应提升回归模型参数列表

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 描述 |
| 模型存储路径 | 输入 | 字符串，表示待加载模型的存储路径 |
| 模型名称 | 输入 | 字符串，表示待加载模型的名称。 |
| 自适应提升回归模型 | 输入输出 | 表示存储加载结果的自适应提升回归模型 |

* + - * 1. 接口返回值

没有错误: 表示加载成功。

参数错误: 表示参数出错。

1.

（资料性附录）

量化原理

* 1. 非对称量化方式（Asymmetric, ASYM）

本文件使用的非对称量化方式原理与《人工智能算子接口标准——第1部分：基础数学操作》中的非对称量化方式原理保持一致。

* 1. 对称量化方式（Symmetric，SYM）

本文件使用的对称量化方式原理与《人工智能算子接口标准——第1部分：基础数学操作》中的对称量化方式原理保持一致。

1.

（资料性附录）

C语言参考定义

* 1. 数据结构

本文件使用的C语言参考定义数据结构与《人工智能算子接口标准——第1部分：基础数学操作》中的C语言参考定义数据结构保持一致。

* 1. 机器学习操作
		1. K-最近邻算法（KNN）
			1. 创建KNN回归模型

**C语法：**

Status aitisa\_create\_knn\_regressor(const int k, const KnnWeightType wtype,

const KnnAlgorithm algo,

const KnnMetric metric,

KnnModel\* model);

**参数：**

k (IN)：代表选择的邻近点个数。

wtype (IN): 枚举类型变量。

algo (IN): 结构体变量，计算最邻近点算法类型以及所需要的额外参数。

metric (IN): 结构体变量，计算距离矩阵算法类型以及所需要的额外参数。

model (OUT): 新创建的KNN回归模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\* k: 1

 algo: { type: BRUTE , leaf\_size: 0}

 metric: {type: EUCLIDEAN, p: 0, v: 0, w: 0}

\*/

KnnModel model;

aitisa\_create\_knn\_regressor(k, UNIFORM, algo, metric, &model);

* + - 1. 创建KNN分类模型

**C语法：**

Status aitisa\_create\_knn\_classifier(const int k, const WeightType wtype,

 const Algorithm algo,

const Metric metric,

KnnModel\* model);

**参数：**

k (IN)：代表选择的邻近点个数。

wtype (IN): 创建的KNN分类模型在预测时采用的权重

algo (IN): Algorithm结构体变量

metric (IN): Metric结构体变量

model (OUT): 新创建的KNN分类模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\* k: 1

 algo: { type: BRUTE , leaf\_size: 0}

 metric: {type: EUCLIDEAN, p: 0, v: 0, w: 0}

\*/

KnnModel model;

aitisa\_create\_knn\_classifier(k, UNIFORM, algo, metric, &model);

* + - 1. 训练KNN模型

**C语法：**

Status aitisa\_train\_knn (const Tensor ref,

const Tensor label,

KnnModel\* model);

**参数：**

ref (IN)：表示输入的训练集特征张量

label (IN): 表示输入的训练集数据标签张量。

model (INOUT): KNN模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[1, 2, 3], [1, 5, 6], [2, 7, 3]]

 label = [[0], [1], [0]]

\*/

aitisa\_train\_knn(ref, label, &model);

* + - 1. 使用KNN模型进行预测

**C语法：**

Status aitisa\_predict\_knn (const Tensor query,

const KnnModel model,

Tensor\* output);

**参数：**

query (IN)：表示输入的测试集特征张量。

model (IN): KNN模型。

output (INOUT): 表示输出的预测结果张量。

**返回值：**

STATUS\_SUCCESS: 表示预测成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* query = [[1.1, 2.1, 3.1]]

\*/

Tensor output;

aitisa\_predict\_knn(query, model, &output);

* + - 1. 销毁KNN模型

**C语法：**

Status aitisa\_destroy\_knn (KnnModel\* model);

**参数：**

model (INOUT): 待销毁的KNN模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁KNN模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_knn(&model);

* + 1. 支持向量机算法（SVM）
			1. 创建C-SVC分类模型

**C语法：**

Status aitisa\_create\_c\_svc( const SvmKernel kernel,

const double cost,

const double epsilon,

SvmModel\* model);

**参数：**

kernel (IN)：结构体变量，用于设置SVM算法采用的核方法类型以及所需要的额外参数。

 cost(IN)：惩罚参数C。

 epsilon(IN)：精度，终止误差。

model (OUT): 新创建的C-SVC分类模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 kernel: {type: RBF, degree: 0, gamma: 0.5, coef0: 0}

 cost: 4.0

 epsilon: 0.001

\*/

SvmModel model;

aitisa\_create\_c\_svc(kernel, cost, epsilon, &model);

* + - 1. 创建nu-SVC分类模型

**C语法：**

Status aitisa\_create\_nu\_svc(const SvmKernel kernel,

const double nu,

const double epsilon,

SvmModel\* model);

**参数：**

kernel (IN)：结构体变量，用于设置SVM算法采用的核方法类型以及所需要的额外参数。

 nu(IN)：控制参数nu。

 epsilon(IN)：精度，终止误差。

model (OUT): 新创建的C-SVC分类模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 kernel: {type: RBF, degree: 0, gamma: 0.5, coef0: 0}

 nu: 0.5

 epsilon: 0.001

\*/

SvmModel model;

aitisa\_create\_nu\_svc(kernel, cost, epsilon, &model);

* + - 1. 创建one-class SVM模型

**C语法：**

Status aitisa\_create\_oneclass\_svm ( const SvmKernel kernel,

const double nu,

const double epsilon,

SvmModel\* model);

**参数：**

 kernel (IN)：结构体变量，用于设置SVM算法采用的核方法类型以及所需要的额外参数。

 nu(IN)：控制参数nu。

 epsilon(IN)：精度，终止误差。

model (OUT): 新创建的one-class SVM模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 kernel: {type: RBF, degree: 0, gamma: 0.5, coef0: 0}

 nu: 0.5

 epsilon: 0.001

\*/

SvmModel model;

aitisa\_create\_oneclass\_svm(kernel, cost, epsilon, &model);

* + - 1. 创建epsilon-SVR回归模型

**C语法：**

Status aitisa\_create\_eps\_svr( const SvmKernel kernel,

const double cost,

const double epsilon,

SvmModel\* model);

**参数：**

 kernel (IN)：结构体变量，用于设置SVM算法采用的核方法类型以及所需要的额外参数。

 cost(IN)：惩罚参数C。

 epsilon(IN)：精度，终止误差。

model (OUT): 新创建的epsilon-SVR回归模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 kernel: {type: RBF, degree: 0, gamma: 0.5, coef0: 0}

 cost: 4.0

 epsilon: 0.001

\*/

SvmModel model;

aitisa\_create\_eps\_svr(kernel, cost, epsilon, &model);

* + - 1. 创建nu-SVR回归模型

**C语法：**

Status aitisa\_create\_nu\_svr( const SvmKernel kernel,

const double cost,

const double nu,

const double epsilon,

SvmModel\* model);

**参数：**

kernel (IN)：结构体变量，用于设置SVM算法采用的核方法类型以及所需要的额外参数。

 cost(IN)：惩罚参数C。

nu(IN)：控制参数nu。

 epsilon(IN)：精度，终止误差。

model (OUT): 新创建的epsilon-SVR回归模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 kernel: {type: RBF, degree: 0, gamma: 0.5, coef0: 0}

 cost: 4.0

 nu: 0.5

 epsilon: 0.001

\*/

SvmModel model;

aitisa\_create\_nu\_svr(kernel, cost, nu, epsilon, &model);

* + - 1. 训练SVM模型

**C语法：**

Status aitisa\_train\_svm (const Tensor ref,

const Tensor label,

SvmModel\* model);

**参数：**

ref (IN)：表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects]。

label (IN): 表示输入的训练集数据标签张量。

model (INOUT): SVM模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[1, 2, 3], [1, 5, 6], [2, 7, 3]]

 label = [[0], [1], [0]]

\*/

aitisa\_train\_svm(ref, label, &model);

* + - 1. 使用SVM模型进行预测

**C语法：**

Status aitisa\_predict\_svm (const Tensor query,

const SvmModel model,

Tensor\* output);

**参数：**

query (IN)：表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre]。

model (IN): SVM模型。

output (INOUT): 表示输出的预测结果张量，其形状为[num\_vects\_pre]。

**返回值：**

STATUS\_SUCCESS: 表示预测成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* query = [[1.1, 2.1, 3.1]]

\*/

Tensor output;

aitisa\_predict\_svm(query, model, &output);

* + - 1. 销毁SVM模型

**C语法：**

Status aitisa\_destroy\_svm (SvmModel\* model);

**参数：**

model (INOUT): 待销毁的SVM模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁SVM模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_svm(&model);

* + 1. 线性回归算法（Linear Regression）
			1. 创建Linear Regression模型

**C语法：**

Status aitisa\_create\_linear\_reg( const Regularization reg,

LinearRegModel\* model);

**参数：**

reg(IN)：结构体变量，用于设置线性回归算法采用的正则化类型和值。

model (OUT): 新创建的Linear Regression模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 reg:{type: L2, value: 0.5}

\*/

LinearRegModel model;

aitisa\_create\_linear\_reg (reg, &model);

* + - 1. 训练Linear Regression模型

**C语法：**

Status aitisa\_train\_linear\_reg (const Tensor ref,

const Tensor label,

LinearRegModel\* model);

**参数：**

ref (IN)：表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects]。

label (IN): 表示输入的训练集数据标签张量。

model (INOUT): Linear Regression模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[0, 0, 0], [1, 1, 1], [2, 2, 2]]

 label = [[0], [3], [6]]

\*/

aitisa\_train\_linear\_reg (ref, label, &model);

* + - 1. 使用Linear Regression模型进行预测

**C语法：**

Status aitisa\_ predict\_linear\_reg (const Tensor query,

const LinearRegModel model,

Tensor\* output);

**参数：**

query (IN)：表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre]。

model (IN): Linear Regression模型。

output (INOUT): 表示输出的预测结果张量，其形状为[num\_vects\_pre]。

**返回值：**

STATUS\_SUCCESS: 表示预测成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\*

query = [[1.5, 1.5, 1.5]]

\*/

Tensor output;

aitisa\_predict\_linear\_reg (query, model, &output);

* + - 1. 销毁Linear Regression模型

**C语法：**

Status aitisa\_destroy\_linear\_reg (LinearRegModel \* model);

**参数：**

model (INOUT): 待销毁的Linear Regression模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁Linear Regression模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_linear\_reg (&model);

* + 1. 逻辑回归算法（Logistic Regression）
			1. 创建Logistic Regression模型

**C语法：**

Status aitisa\_create\_logistic\_reg( const Solver solver,

const Regularization reg,

LogisticRegModel\* model);

**参数：**

solver (IN)：结构体变量，用于设置Logistic Regression模型使用的求解器信息，包括求解器类型和参数。

Reg(IN)：结构体变量，用于设置Logistic Regression算法采用的正则化类型和值。

model (OUT): 新创建的Logistic Regression模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 solver:{type: Momentum, learning\_rate: 0.001, momentum:0.9}

reg:{type: L2, value: 0.5}

\*/

LogisticRegModel model;

aitisa\_create\_logistic\_reg (solver, reg, &model);

* + - 1. 训练Logistic Regression模型

**C语法：**

Status aitisa\_ train\_logistic\_reg (const Tensor ref,

const Tensor label,

LogisticRegModel\* model);

**参数：**

ref (IN)：表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects]。

label (IN): 表示输入的训练集数据标签张量。

model (INOUT): Logistic Regression模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[0, 0, 0], [-1, -1, -1]]

 label = [[0], [1]]

\*/

aitisa\_train\_logistic\_reg (ref, label, &model);

* + - 1. 使用Logistic Regression模型进行预测

**C语法：**

Status aitisa\_ predict\_logistic\_reg (const Tensor query,

const LogisticRegModel model,

Tensor\* output);

**参数：**

query (IN)：表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre]。

model (IN): Logistic Regression模型。

output (INOUT): 表示输出的预测结果张量，其形状为[num\_vects\_pre]。

**返回值：**

STATUS\_SUCCESS: 表示预测成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* query = [[1, 1, 1]]

\*/

Tensor output;

aitisa\_predict\_logistic\_reg (query, model, &output);

* + - 1. 销毁Logistic Regression模型

**C语法：**

Status aitisa\_destroy\_logistic\_reg (LogisticRegModel \* model);

**参数：**

model (INOUT): 待销毁的Logistic Regression模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁Logistic Regression模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_logistic\_reg (&model);

* + 1. 模型评估（Model Evaluation）
			1. 均方误差

**C语法：**

Status aitisa\_mean\_squared\_error ( const Tensor y\_true,

const Tensor y\_pred,

float\* error);

**参数：**

y\_true (IN)：真实值

y\_pred (IN): 预测值

error(OUT)：均方误差

**返回值：**

STATUS\_SUCCESS: 表示计算成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\*

 y\_true = {3, -0.5, 2, 7}

 y\_pred = {2.5, 0.0, 2, 8}

 float error = 0;

\*/

aitisa\_mean\_squared\_error(y\_true, y\_pred, &error);

/\*

 error = 0.375

\*/

* + - 1. 最大误差

**C语法：**

Status aitisa\_max\_error ( const Tensor y\_true,

const Tensor y\_pred,

float\* error);

**参数：**

y\_true (IN)：真实值

y\_pred (IN): 预测值

error(OUT)：最大误差

**返回值：**

STATUS\_SUCCESS: 表示计算成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\*

 y\_true = {3, 2, 7, 1}

 y\_pred = {4, 2, 7, 1}

 float error = 0;

\*/

aitisa\_max\_error(y\_true, y\_pred, &error);

/\*

 error = 1

\*/

* + - 1. R2得分

**C语法：**

Status aitisa\_r2\_score ( const Tensor y\_true,

const Tensor y\_pred,

float\* score);

**参数：**

y\_true (IN)：真实值

y\_pred (IN): 预测值

score(OUT)：均方得分

**返回值：**

STATUS\_SUCCESS: 表示计算成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\*

 y\_true = {3, -0.5, 2, 7}

 y\_pred = {2.5, 0.0, 2, 8}

 float score = 0;

\*/

aitisa\_r2\_score(y\_true, y\_pred, &score);

/\*

 score= 0.948

\*/

* + - 1. 正确性得分

**C语法：**

Status aitisa\_accuracy\_score ( const Tensor y\_true,

const Tensor y\_pred,

float\* score);

**参数：**

y\_true (IN)：真实值

y\_pred (IN): 预测值

score(OUT)：正确性得分

**返回值：**

STATUS\_SUCCESS: 表示计算成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\*

 y\_true = {0, 2, 1, 3}

 y\_pred = {0, 1, 2, 3}

 float score = 0;

\*/

aitisa\_max\_error(y\_true, y\_pred, &error);

/\*

 score = 0.5

\*/

* + 1. 主成分分析 (Principal Component Analysis)
			1. 创建PCA降维模型

**C语法：**

Status aitisa\_create\_pca(const int k, const SvdSolver solver,

PcaModel\* model);

**参数：**

k (IN)：代表需要保存的主成分个数。

SvdSolver (IN): 结构体变量，计算奇异值分解时用的求解器类型以及所需要的额外参数。

model (OUT): 新创建的PCA主成分分析模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\* k: 10

 solver: { type: auto ,tol = 0.1, iterated\_power = 1000}

\*/

PcaModel model;

aitisa\_create\_pca(k, solver, &model);

* + - 1. 训练PCA模型

**C语法：**

Status aitisa\_train\_pca (const Tensor ref,

PcaModel\* model);

**参数：**

ref (IN)：表示输入的训练集特征张量

model (INOUT): PCA模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[1, 2, 3], [1, 5, 6], [2, 7, 3]]

\*/

aitisa\_train\_pca(ref, &model);

* + - 1. 利用PCA模型对数据集降维

**C语法：**

Status aitisa\_ dim\_reduction\_pca (const Tensor query,

const PcaModel model,

Tensor\* output);

**参数：**

query (IN)：表示输入的测试集特征张量。

model (IN): PCA模型。

output (INOUT): 表示输出数据集降维后的结果张量。

**返回值：**

STATUS\_SUCCESS: 表示降维成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* query = [[1.1, 2.1, 3.1]]

\*/

Tensor output;

aitisa\_ dim\_reduction\_pca (query, model, &output);

* + - 1. 销毁PCA模型

**C语法：**

Status aitisa\_destroy\_pca (PcaModel \* model);

**参数：**

model (INOUT): 待销毁的PCA模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁PCA模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_pca (&model);

* + 1. 线性判别分析 (Linear Discriminant Analysis)
			1. 创建LDA线性判别分析模型

**C语法：**

Status aitisa\_create\_lda(const int k,

const LdaSolver solver,

LdaModel\* model);

**参数：**

k (IN)：代表需要降维的维度数。

LdaSolver (IN): 结构体变量，求解LDA模型时用的求解器类型以及所需要的额外参数。

model (OUT): 新创建的LDA主成分分析模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\* k: 10

 solver: { type: svd ,tol = 1e-4, shrinkage:auto }

\*/

LdaModel model;

aitisa\_create\_lda(k, solver, &model);

* + - 1. 训练LDA模型

**C语法：**

Status aitisa\_ train\_lda (const Tensor ref,

const Tensor label,

LdaRegModel\* model);

**参数：**

ref (IN)：表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects]。

label (IN): 表示输入的训练集数据标签张量。

model (INOUT): LDA模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[0, 0, 0], [-1, -1, -1]]

 label = [[0], [1]]

\*/

aitisa\_train\_lda (ref, label, &model);

* + - 1. 利用LDA模型进行特征降维

**C语法：**

Status aitisa\_ dim\_reduction\_lda (const Tensor query,

const LdaModel model,

Tensor\* output);

**参数：**

query (IN)：表示输入的测试集特征张量。

model (IN): LDA模型。

output (INOUT): 表示输出数据集降维后的结果张量。

**返回值：**

STATUS\_SUCCESS: 表示降维成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* query = [[1.1, 2.1, 3.1]]

\*/

Tensor output;

aitisa\_ dim\_reduction\_lda (query, model, &output);

* + - 1. 利用LDA模型进行分类

**C语法：**

Status aitisa\_ predict\_lda (const Tensor query,

const LdaModel model,

Tensor\* output);

**参数：**

query (IN)：表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre]。

model (IN): LDA模型。

output (INOUT): 表示输出的预测结果张量，其形状为[num\_vects\_pre]。

**返回值：**

STATUS\_SUCCESS: 表示预测成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* query = [[1, 1, 1]]

\*/

Tensor output;

aitisa\_predict\_lda (query, model, &output);

* + - 1. 销毁LDA模型

**C语法：**

Status aitisa\_destroy\_lda (LdaModel \* model);

**参数：**

model (INOUT): 待销毁的LDA模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁LDA模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_lda (&model);

* + 1. 高斯朴素贝叶斯算法 (Gaussian Naïve Bayesian)
			1. 创建高斯朴素贝叶斯模型

**C语法：**

Status aitisa\_create\_ gaussian\_nb ( float priors[],

 float smooth,

GaussianNBModel\* model);

**参数：**

priors(IN)：浮点数数组，用于设置高斯朴素贝叶斯算法中每类的先验概率分布。

smooth(IN): 浮点数，用于设置平滑参数。

model (OUT): 新创建的Gaussian Naïve Bayesian模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 priors: [0.2, 0.3, 0.3]

 smooth: 1e-6

\*/

GaussianNB model;

aitisa\_create\_ gaussian\_nb (priors, smooth, &model);

* + - 1. 训练高斯朴素贝叶斯模型

**C语法：**

Status aitisa\_train\_ gaussian\_nb (const Tensor ref,

const Tensor label,

GaussianNBModel \* model);

**参数：**

ref (IN)：表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects]。

label (IN): 表示输入的训练集数据标签张量。

model (INOUT): Gaussian Naïve Bayesian模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[0, 0, 0], [1, 1, 1], [2, 2, 2]]

 label = [[0], [3], [6]]

\*/

aitisa\_train\_ gaussian\_nb (ref, label, &model);

* + - 1. 使用高斯朴素贝叶斯模型进行预测

**C语法：**

Status aitisa\_ predict\_ gaussian\_nb (const Tensor query,

const GaussianNBModel model,

Tensor\* output);

**参数：**

query (IN)：表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre]。

model (IN): Gaussian Naïve Bayesian模型。

output (INOUT): 表示输出的预测结果张量，其形状为[num\_vects\_pre]。

**返回值：**

STATUS\_SUCCESS: 表示预测成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\*

query = [[1.5, 1.5, 1.5]]

\*/

Tensor output;

aitisa\_predict\_ gaussian\_nb (query, model, &output);

* + - 1. 销毁高斯朴素贝叶斯模型

**C语法：**

Status aitisa\_destroy\_ gaussian\_nb (GaussianNBModel \* model);

**参数：**

model (INOUT): 待销毁的Gaussian Naïve Bayesian模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁Gaussian Naïve Bayesian模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_ gaussian\_nb (&model);

* + 1. 决策树分类模型 (Decision Tree Classifier)
			1. 创建决策树分类模型

**C语法：**

Status aitisa\_create\_ decisiontree\_classifier ( char criterion[],

 int max\_depth,

 int min\_num\_split,

 int min\_num\_leaf,

DecisionTreeClassifierModel\* model)

**参数：**

criterion(IN): 字符串，用于设置决策树生成过程中进行特征选择的标准。

max\_depth(IN): 整数，用于设置生成决策树的最大深度。

max\_num\_split(IN): 整数，设置进行节点分裂时所需要的最小样本数。

max\_num\_leaf(IN): 整数，设置生成叶节点所需最小样本数。

model (OUT): 新创建的Decision Tree Classifier模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 criterion: “gini”

 max\_depth: 20

 min\_num\_split: 10

 min\_num\_leaf: 5

\*/

DecisionTreeClassifierModel model;

aitisa\_create\_ decisiontree\_classifier (criterion,max\_depth,

min\_num\_split,min\_num\_leaf,&model);

* + - 1. 训练决策树分类模型

**C语法：**

Status aitisa\_train\_ decisiontree\_classifier: (const Tensor ref,

const Tensor label,

DecisionTreeClassifierModel \* model)

**参数：**

ref (IN)：表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects]。

label (IN): 表示输入的训练集数据标签张量。

model (INOUT): Decision Tree Classifier模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[0, 0, 0], [1, 1, 1], [2, 2, 2]]

 label = [[0], [3], [6]]

\*/

DecisionTreeClassifierModel model;

aitisa\_train\_ decisiontree\_classifier (ref, label, &model);

* + - 1. 使用决策树分类模型进行预测

**C语法：**

Status aitisa\_ predict\_ decisiontree\_classifier (const Tensor query,

const DecisionTreeClassifierModel model,

Tensor\* output)

**参数：**

query (IN)：表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre]。

model (IN): Decision Tree Classifier模型。

output (INOUT): 表示输出的预测结果张量，其形状为[num\_vects\_pre]。

**返回值：**

STATUS\_SUCCESS: 表示预测成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\*

query = [[1.5, 1.5, 1.5]]

\*/

Tensor output;

aitisa\_predict\_ decisiontree\_classifier (query, model, &output);

* + - 1. 销毁决策树分类模型

**C语法：**

Status aitisa\_destroy\_ decisiontree\_classifier (DecisionTreeClassifierModel \* model)

**参数：**

model (INOUT): 待销毁的Decision Tree Classifier模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁Decision Tree Classifier模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_ decisiontree\_classifier (&model);

* + 1. 决策树回归模型 (Decision Tree Regressor)
			1. 创建决策树回归模型

**C语法：**

Status aitisa\_create\_ decisiontree\_ regressor ( char criterion[],

 int max\_depth,

 int min\_num\_split,

 int min\_num\_leaf,

DecisionTreeRegressor Model\* model)

**参数：**

criterion(IN): 字符串，用于设置决策树生成过程中节点分裂质量的度量函数。

max\_depth(IN): 整数，用于设置生成决策树的最大深度。

max\_num\_split(IN): 整数，设置进行节点分裂时所需要的最小样本数。

max\_num\_leaf(IN): 整数，设置生成叶节点所需最小样本数。

model (OUT): 新创建的Decision Tree Regressor模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 criterion: “mse”

 max\_depth: 20

 min\_num\_split: 10

 min\_num\_leaf: 5

\*/

DecisionTreeRegressorModel model;

aitisa\_create\_ decisiontree\_ regressor(criterion,max\_depth,

min\_num\_split,min\_num\_leaf,&model);

* + - 1. 训练决策树回归模型

**C语法：**

Status aitisa\_train\_ decisiontree\_ regressor: (const Tensor ref,

const Tensor label,

DecisionTreeRegressorModel \* model)

**参数：**

ref (IN)：表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects]。

label (IN): 表示输入的训练集数据标签张量。

model (INOUT): Decision Tree Regressor模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[0, 0, 0], [1, 1, 1], [2, 2, 2]]

 label = [[0], [3], [6]]

\*/

DecisionTreeRegressorModel model;

aitisa\_train\_ decisiontree\_ regressor (ref, label, &model);

* + - 1. 使用决策树回归模型进行预测

**C语法：**

Status aitisa\_ predict\_ decisiontree\_ regressor (const Tensor query,

const DecisionTreeRegressorModel model,

Tensor\* output)

**参数：**

query (IN)：表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre]。

model (IN): Decision Tree Regressor模型。

output (INOUT): 表示输出的预测结果张量，其形状为[num\_vects\_pre]。

**返回值：**

STATUS\_SUCCESS: 表示预测成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\*

query = [[1.5, 1.5, 1.5]]

\*/

Tensor output;

aitisa\_predict\_ decisiontree\_ regressor (query, model, &output);

* + - 1. 销毁决策树回归模型

**C语法：**

Status aitisa\_destroy\_ decisiontree\_ regressor (DecisionTreeRegreesorModel \* model)

**参数：**

model (INOUT): 待销毁的Decision TreeRegreesor模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁Decision Tree Regreesor模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_ decisiontree\_regressor (&model);

* + 1. 随机森林分类模型 (Random Forest Classifier)
			1. 创建随机森林分类模型

**C语法：**

Status aitisa\_create\_ randomforest \_classifier ( char criterion[],

 int num\_tree,

 int max\_depth,

 int min\_num\_split,

 int min\_num\_leaf,

 int bootstrap,

 int seed,

RandomForestClassifierModel\* model)

**参数：**

criterion(IN): 字符串，用于设置决策树生成过程中进行特征选择的标准。

num\_tree(IN): 整数，设置随机森林中树的个数。

max\_depth(IN): 整数，用于设置生成决策树的最大深度。

max\_num\_split(IN): 整数，设置进行节点分裂时所需要的最小样本数。

max\_num\_leaf(IN): 整数，设置生成叶节点所需最小样本数。

bootstrap(IN): 整数，表示构造树的时候是否采用自助法抽取样本进行训练的标志

seed(IN): 整数，控制生成树的过程中的随机性。

model (OUT): 新创建的Random Forest Classifier模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 criterion: “gini”

 max\_depth: 20

 min\_num\_split: 10

 min\_num\_leaf: 5

\*/

RandomForestClassifierModel model;

aitisa\_create\_ randomforest \_classifier (criterion,max\_depth,

min\_num\_split,min\_num\_leaf,&model);

* + - 1. 训练随机森林分类模型

**C语法：**

Status aitisa\_train\_ randomforest \_classifier: (const Tensor ref,

const Tensor label,

RandomForestClassifierModel \* model)

**参数：**

ref (IN)：表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects]。

label (IN): 表示输入的训练集数据标签张量。

model (INOUT): Random Forest Classifier模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[0, 0, 0], [1, 1, 1], [2, 2, 2]]

 label = [[0], [3], [6]]

\*/

RandomForestClassifierModel model

aitisa\_train\_ randomforest \_classifier (ref, label, &model);

* + - 1. 使用随机森林分类模型进行预测

**C语法：**

Status aitisa\_ predict\_ randomforest \_classifier (const Tensor query,

const DecisionTreeClassifierModel model,

Tensor\* output)

**参数：**

query (IN)：表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre]。

model (IN): Random Forest Classifier模型。

output (INOUT): 表示输出的预测结果张量，其形状为[num\_vects\_pre]。

**返回值：**

STATUS\_SUCCESS: 表示预测成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\*

query = [[1.5, 1.5, 1.5]]

\*/

Tensor output;

aitisa\_predict\_ randomforest \_classifier (query, model, &output);

* + - 1. 销毁随机森林分类模型

**C语法：**

Status aitisa\_destroy\_ randomforest \_classifier (DecisionTreeClassifierModel \* model)

**参数：**

model (INOUT): 待销毁的Random Forest Classifier模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁Random Forest Classifier模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_ randomforest \_classifier (&model);

* + 1. 随机森林回归模型 (Random Forest Regressor)
			1. aitisa\_create\_randomforest\_regressor: 创建随机森林回归模型

**C语法：**

Status aitisa\_create\_ randomforest \_ regressor ( char criterion[],

 int num\_tree,

 int max\_depth,

 int min\_num\_split,

 int min\_num\_leaf,

 int bootstrap,

 int seed,

RandomForestRegressorModel\* model)

**参数：**

criterion(IN): 字符串，用于设置决策树生成过程中进行特征选择的标准。

num\_tree(IN): 整数，设置随机森林中树的个数。

max\_depth(IN): 整数，用于设置生成决策树的最大深度。

max\_num\_split(IN): 整数，设置进行节点分裂时所需要的最小样本数。

max\_num\_leaf(IN): 整数，设置生成叶节点所需最小样本数。

bootstrap(IN): 整数，表示构造树的时候是否采用自助法抽取样本进行训练的标志

seed(IN): 整数，控制生成树的过程中的随机性。

model (OUT): 新创建的Random Forest Regressor模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 criterion: “mse”

 max\_depth: 20

 min\_num\_split: 10

 min\_num\_leaf: 5

\*/

RandomForestRegressorModel model;

aitisa\_create\_ randomforest \_ regressor (criterion,max\_depth,

min\_num\_split,min\_num\_leaf,&model);

* + - 1. 训练随机森林回归模型

**C语法：**

Status aitisa\_train\_ randomforest \_ regressor: (const Tensor ref,

const Tensor label,

RandomForestRegressorModel \* model)

**参数：**

ref (IN)：表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects]。

label (IN): 表示输入的训练集数据标签张量。

model (INOUT): Random Forest Regressor模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[0, 0, 0], [1, 1, 1], [2, 2, 2]]

 label = [[0], [3], [6]]

\*/

RandomForestRegressorModel model

aitisa\_train\_ randomforest \_ regressor (ref, label, &model);

* + - 1. 使用随机森林回归模型进行预测

**C语法：**

Status aitisa\_ predict\_ randomforest \_ regressor (const Tensor query,

const RandomForestRegressorModel model,

Tensor\* output);

**参数：**

query (IN)：表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre]。

model (IN): Random Forest Regressor模型。

output (INOUT): 表示输出的预测结果张量，其形状为[num\_vects\_pre]。

**返回值：**

STATUS\_SUCCESS: 表示预测成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\*

query = [[1.5, 1.5, 1.5]]

\*/

Tensor output;

aitisa\_predict\_ randomforest \_ regressor (query, model, &output);

* + - 1. 销毁随机森林回归模型

**C语法：**

Status aitisa\_destroy\_ randomforest \_ regressor (RandomForestRegressorModel \* model);

**参数：**

model (INOUT): 待销毁的Random Forest Regressor模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁Random Forest Regressor模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_ randomforest \_ regressor (&model);

* + 1. 自适应提升分类模型 (Adaptive Boosting Classifier)
			1. 创建自适应提升分类模型

**C语法：**

Status aitisa\_create\_ adaboost \_classifier (void \* base\_estimator,

 int max\_num\_estimators,

 float learning\_rate,

AdaBoostClassifierModel\* model)

**参数：**

base\_estimator (IN): 分类模型，设置提升方法中使用的基准分类器。

max\_num\_estimators (IN): 整数，用于设置提升方法终止时所能使用的基准分类器的最大数目。

learning\_rate (IN): 浮点数，设置提升方法每一轮迭代中应用到分类器的权重。

model (OUT): 新创建的AdaBoost Classifier模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 base\_estimator: DecisionTreeClassifier dt

 max\_num\_estimators,:40

 learning\_rate: 0.3

\*/

AdaBoostClassifierModel model;

aitisa\_create\_ adaboost \_classifier (dt,max\_num\_estimators, learning\_rate, &model);

* + - 1. 训练自适应提升分类模型

**C语法：**

Status aitisa\_train\_ adaboost \_classifier: (const Tensor ref,

const Tensor label,

AdaBoostClassifierModel \* model)

**参数：**

ref (IN)：表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects]。

label (IN): 表示输入的训练集数据标签张量。

model (INOUT): AdaBoost Classifier模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[0, 0, 0], [1, 1, 1], [2, 2, 2]]

 label = [[0], [3], [6]]

\*/

AdaBoostClassifierModel model

aitisa\_train\_ adaboost \_classifier (ref, label, &model);

* + - 1. 使用自适应提升分类模型进行预测

**C语法：**

Status aitisa\_ predict\_ adaboost \_classifier (const Tensor query,

const AdaBoostClassifierModel model,

Tensor\* output)

**参数：**

query (IN)：表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre]。

model (IN): AdaBoost Classifier模型。

output (INOUT): 表示输出的预测结果张量，其形状为[num\_vects\_pre]。

**返回值：**

STATUS\_SUCCESS: 表示预测成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\*

query = [[1.5, 1.5, 1.5]]

\*/

Tensor output;

aitisa\_predict\_ adaboost \_classifier (query, model, &output);

* + - 1. 销毁自适应提升分类模型

**C语法：**

Status aitisa\_destroy\_ adaboost \_classifier (AdaBoostClassifierModel \* model)

**参数：**

model (INOUT): 待销毁的AdaBoost Classifier模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁AdaBoost Classifier模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_ adaboost \_classifier (&model);

* + 1. 自适应提升回归模型 (Adaptive Boosting Regressor)
			1. 创建自适应提升回归模型

**C语法：**

Status aitisa\_create\_ adaboost \_ regressor (void \* base\_estimator,

 int max\_num\_estimators,

 float learning\_rate,

AdaBoostRegressorModel\* model)

**参数：**

base\_estimator (IN): 模型，设置提升方法中使用的基准回归模型。

max\_num\_estimators (IN): 整数，用于设置提升方法终止时所能使用的基准回归模型的最大数目。

learning\_rate (IN): 浮点数，设置提升方法每一轮迭代中应用到回归模型的权重。

model (OUT): 新创建的AdaBoost Regressor模型。

**返回值：**

STATUS\_SUCCESS: 表示成功创建模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

STATUS\_ALLOC\_FAILED: 表示创建模型分配空间不足。

**示例：**

/\*

 base\_estimator: DecisionTreeRegressor dt

 max\_num\_estimators,:40

 learning\_rate: 0.3

\*/

AdaBoostRegressorModel model;

aitisa\_create\_ adaboost \_ regressor (dt,max\_num\_estimators, learning\_rate, &model);

* + - 1. 训练自适应提升回归模型

**C语法：**

Status aitisa\_train\_ adaboost \_ regressor: (const Tensor ref,

const Tensor label,

AdaBoostRegressorModel \* model)

**参数：**

ref (IN)：表示输入的训练集特征张量。其形状为[num\_vects, dim\_vects]。

label (IN): 表示输入的训练集数据标签张量。

model (INOUT): AdaBoost Regressor模型。

**返回值：**

STATUS\_SUCCESS: 表示训练成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\* ref = [[0, 0, 0], [1, 1, 1], [2, 2, 2]]

 label = [[0], [3], [6]]

\*/

AdaBoostRegressorModel model

aitisa\_train\_ adaboost \_ regressor (ref, label, &model);

* + - 1. 使用自适应提升回归模型进行预测

**C语法：**

Status aitisa\_ predict\_ adaboost \_ regressor (const Tensor query,

const AdaBoosRegressorModel model,

Tensor\* output)

**参数：**

query (IN)：表示输入的测试集特征张量。其形状为[num\_vects\_pre, dim\_vects\_pre]。

model (IN): Adaptive Boosting Regressor模型。

output (INOUT): 表示输出的预测结果张量，其形状为[num\_vects\_pre]。

**返回值：**

STATUS\_SUCCESS: 表示预测成功。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

/\*

query = [[1.5, 1.5, 1.5]]

\*/

Tensor output;

aitisa\_predict\_ adaboost \_ regressor (query, model, &output);

* + - 1. 销毁自适应提升回归模型

**C语法：**

Status aitisa\_destroy\_ adaboost \_ regressor (AdaBoostRegressorModel \* model)

**参数：**

model (INOUT): 待销毁的AdaBoost Regressor模型。

**返回值：**

STATUS\_SUCCESS: 表示成功销毁AdaBoost Regressor模型。

STATUS\_INVALID\_ARGRUMENT: 表示参数出错。

**示例：**

aitisa\_destroy\_ adaboost \_ regressor (&model);