ICS 35.020
CCS L 70

团　　体　　标　　准

T/AI 131.3—2025

人工智能 算子接口
第 3 部分：机器学习类

Artificial intelligence — Operater interface—Part 3: Machine learning operators

2025 - 11 - 19 发布　　2025 - 11 - 19 实施

中关村视听产业技术创新联盟　发布

目　　次

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 131《人工智能 算子接口》的第3部分。T/AI 131已经发布了以下部分：

——第1部分：基础数学类；

——第2部分：神经网络类。

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、中科寒武纪科技股份有限公司。

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

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 131《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 131《人工智能 算子接口》拟由六个部分构成。

——第1部分：基础数学操作。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：神经网络操作。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。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。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及参数要求。

本文件适用于人工智能机器学习类算子库的设计、开发与应用，以及相关软硬件及系统的研制。

2 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 131.1—2025 人工智能 算子接口 第1部分：基础数学操作

3 术语和定义

T/AI 131.1—2025中界定的术语和定义适用于本文件。

4 缩略语

下列缩略语适用于本文件。

ABC：自适应提升分类（Adaptive Boosting Classifier）

ABR：自适应提升回归（Adaptive Boosting Regressor）

C-SVC：C-支持向量机分类（C-Support Vector Classification）

Nu-SVC：Nu-支持向量机分类（ν-Support Vector Classification）

DT：决策树（Decision Tree）

DTC：决策树分类（Decision Tree Classifier）

DTR：决策树回归（Decision Tree Regressor）

GNB：高斯朴素贝叶斯算法(Gaussian Naive Bayesian)

KNN：K-最近邻（k-Nearest Neighbor）

LDA：线性判别分析算法（Linear Discriminant Analysis）

LinR 逻辑回归（Logistic Regression）

LogR：线性回归（Linear Regression）

ME：模型评估（Model Evaluation）

PCA：主成分分析（Principle Component Analysis）

RF：随机森林（Random Forest）

RFC：随机森林分类（Random Forest Classifier）

RFR：随机森林回归（Random Forest Regressor）

SVC：支持向量机分类（Support Vector Classifier）

SVM：支持向量机（Support Vector Machine）

SVR：支持向量机回归（Support Vector Regressor）

5 通则

本章内容应符合 T/AI 131.1—2025 中第 5 章要求。

6 数据结构

本章内容应符合 T/AI 131.1—2025 中第 6 章要求。

7 机器学习类算子接口

7.1 接口列表

本条主要针对机器学习模块，共提出 14 类共 90 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	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、保存 KNN 回归模型、加载 KNN 回归模型、保存 KNN 分类模型、加载 KNN 分类模型（9 个）
支持向量机算法	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、保存 SVM 模型、加载 SVM 模型（10 个）
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、保存线性回归模型、加载线性回归模型（6 个）
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、保存逻辑回归模型、加载逻辑回归模型（6 个）
模型评估	均方误差、最大误差、R2 得分、正确性得分（4 个）
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、保存 PCA 模型、加载 PCA 模型（6 个）
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、保存 LDA 模型、加载 LDA 模型（7 个）
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、保存 GNB 模型、加载 GNB 模型（6 个）
决策树分类模型	创建决策树分类器模型、训练决策树分类模型、使用决策树分类模型进行分类、销毁决策树分类模型、保存决策树分类模型、加载决策树分类模型（6 个）

表 1　机器学习函数接口列表（续）

决策树回归模型	创建决策树回归模型、训练决策树回归模型、使用决策树回归模型进行回归、销毁决策树回归模型、保存决策树回归模型、加载决策树回归模型（6 个）
随机森林分类模型	创建随机森林分类模型、训练随机森林分类模型、使用随机森林分类模型进行分类、销毁随机森林分类模型、保存随机森林分类模型、加载随机森林分类模型（6 个）
随机森林回归模型	创建随机森林回归模型、训练随机森林回归模型、使用随机森林回归模型进行回归、销毁随机森林回归模型、保存随机森林回归模型、加载随机森林回归模型（6 个）
自适应提升分类模型	创建自适应提升分类模型、训练自适应提升分类模型、使用自适应提升分类模型进行分类、销毁自适应提升分类模型、保存自适应提升分类模型、加载自适应提升分类模型（6 个）
自适应提升回归模型	创建自适应提升回归模型、训练自适应提升回归模型、使用自适应提升回归模型进行回归、销毁自适应提升回归模型、保存自适应提升回归模型、加载自适应提升回归模型（6 个）

7.2　算子接口

7.2.1　K-最近邻算法

7.2.1.1　创建 KNN 回归模型

7.2.1.1.1　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。

7.2.1.1.2　接口参数

创建 KNN 回归模型函数前向接口应符合表 2，C 代码示例见附录 A2.1.1。

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

7.2.1.1.3　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。
其它错误：其他操作出错情况。

7.2.1.2 创建 KNN 分类模型

7.2.1.2.1 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。

7.2.1.2.2 接口参数

创建KNN分类模型函数前向接口应符合表3，C代码示例见附录A2.1.2。

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

7.2.1.2.3 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。
其它错误：其他操作出错情况。

7.2.1.3 训练 KNN 模型

7.2.1.3.1 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模型结构体。

7.2.1.3.2 接口参数

训练KNN模型函数前向接口应符合表4，C代码示例见附录A2.1.3。

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

表 4　训练 KNN 模型参数列表（续）

参数名	类型	描述
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

7.2.1.3.3　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。
其它错误：其他操作出错情况。

7.2.1.4　使用 KNN 模型进行预测

7.2.1.4.1　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。

7.2.1.4.2　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代码示例见附录A2.1.4。

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

7.2.1.4.3　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。
其它错误：其他操作出错情况。

7.2.1.5　销毁 KNN 模型

7.2.1.5.1　功能

销毁某个指定的KNN模型。

7.2.1.5.2　接口参数

销毁KNN模型函数前向接口应符合表6，C代码示例见附录A2.1.5。

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

7.2.1.5.3　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。
其它错误：其他操作出错情况。

7.2.1.6　保存 KNN 回归模型

7.2.1.6.1　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。

7.2.1.6.2　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

7.2.1.6.3　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。
其它错误：其他操作出错情况。

7.2.1.7　加载 KNN 回归模型

7.2.1.7.1　功能

加载一个保存好的 KNN 回归模型，用于后续预测任务中。

7.2.1.7.2　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

表 8　加载 KNN 回归模型参数列表（续）

参数名	类型	描述
KNN回归模型	输入输出	表示存储加载结果的KNN回归模型

7.2.1.7.3　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。
其它错误：其他操作出错情况。

7.2.1.8　保存 KNN 分类模型

7.2.1.8.1　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。

7.2.1.8.2　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

7.2.1.8.3　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。
其它错误：其他操作出错情况。

7.2.1.9　加载 KNN 分类模型

7.2.1.9.1　功能

加载一个保存好的 KNN 分类模型，用于后续预测任务中。

7.2.1.9.2　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

7.2.1.9.3 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。
其它错误：其他操作出错情况。

7.2.2 支持向量机算法

7.2.2.1 创建 C-SVC 分类模型

7.2.2.1.1 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。

7.2.2.1.2 接口参数

加载C-SVC分类模型函数前向接口应符合表11，C代码示例见附录A2.2.1。

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

7.2.2.1.3 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。
其它错误：其他操作出错情况。

7.2.2.2 创建 nu-SVC 分类模型

7.2.2.2.1 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。

7.2.2.2.2 接口参数

创建nu-SVC分类模型函数前向接口应符合表12，C代码示例见附录A2.2.2。

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

表 12　创建 nu-SVC 分类模型参数列表（续）

参数名	类型	描述
精度	输入	终止条件
nu-SVC模型	输出	新创建的nu-SVC分类模型

7.2.2.2.3　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。
其它错误：其他操作出错情况。

7.2.2.3　创建 one-class SVM 模型

7.2.2.3.1　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。

7.2.2.3.2　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代码示例见附录A2.2.3。

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

7.2.2.3.3　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。
其它错误：其他操作出错情况。

7.2.2.4　创建 epsilon-SVR 回归模型

7.2.2.4.1　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。

7.2.2.4.2 接口参数

创建epsilon-SVR回归模型函数前向接口应符合表14，C代码示例见附录A2.2.4。

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

7.2.2.4.3 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。
其它错误：其他操作出错情况。

7.2.2.5 创建 nu-SVR 回归模型

7.2.2.5.1 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。

7.2.2.5.2 接口参数

创建nu-SVR回归模型函数前向接口应符合表15，C代码示例见附录A2.2.5。

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

7.2.2.5.3 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。
其它错误：其他操作出错情况。

7.2.2.6 训练 SVM 模型

7.2.2.6.1 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模型结构体。

7.2.2.6.2 接口参数

训练SVM模型函数前向接口应符合表16，C代码示例见附录A2.2.6。

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

7.2.2.6.3 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。
其它错误：其他操作出错情况。

7.2.2.7 使用 SVM 模型进行预测

7.2.2.7.1 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。

7.2.2.7.2 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代码示例见附录A2.2.7。

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]

7.2.2.7.3 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。
其它错误：其他操作出错情况。

7.2.2.8 销毁 SVM 模型

7.2.2.8.1 功能

销毁某个指定的 SVM 模型。

7.2.2.8.2 接口参数

销毁SVM模型函数前向接口应符合表18，C代码示例见附录A2.2.8。

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

7.2.2.8.3 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。
其它错误：其他操作出错情况。

7.2.2.9 保存 SVM 模型

7.2.2.9.1 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。

7.2.2.9.2 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

7.2.2.9.3 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。
其它错误：其他操作出错情况。

7.2.2.10 加载 SVM 模型

7.2.2.10.1 功能

加载一个保存好的 KNN 回归模型，用于后续预测任务中。

7.2.2.10.2 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

7.2.2.10.3　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。
其它错误：其他操作出错情况。

7.2.3　线性回归算法

7.2.3.1　创建线性回归模型

7.2.3.1.1　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。

7.2.3.1.2　接口参数

创建线性回归模型函数前向接口应符合表21，C代码示例见附录A2.3.1。

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

7.2.3.1.3　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。
其它错误：其他操作出错情况。

7.2.3.2　训练线性回归模型

7.2.3.2.1　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模型结构体。

7.2.3.2.2　接口参数

训练线性回归模型函数前向接口应符合表22，C代码示例见附录A2.3.2。

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

7.2.3.2.3　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。
其它错误：其他操作出错情况。

7.2.3.3　使用线性回归模型进行预测

7.2.3.3.1　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。

7.2.3.3.2　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代码示例见附录A2.3.3。

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]

7.2.3.3.3　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。
其它错误：其他操作出错情况。

7.2.3.4　销毁线性回归模型

7.2.3.4.1　功能

销毁某个指定的线性回归模型。

7.2.3.4.2　接口参数

销毁线性回归模型函数前向接口应符合表24，C代码示例见附录A2.3.4。

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

7.2.3.4.3　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。
其它错误：其他操作出错情况。

7.2.3.5　保存线性回归模型

7.2.3.5.1　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。

7.2.3.5.2　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

7.2.3.5.3　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。
其它错误：其他操作出错情况。

7.2.3.6　加载线性回归模型

7.2.3.6.1　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。

7.2.3.6.2　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

表 26　加载线性回归模型参数列表（续）

参数名	类型	描述
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

7.2.3.6.3　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。
其它错误：其他操作出错情况。

7.2.4　逻辑回归算法

7.2.4.1　创建逻辑回归模型

7.2.4.1.1　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。

7.2.4.1.2　接口参数

创建逻辑回归模型函数前向接口应符合表27，C代码示例见附录A2.4.1。

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

7.2.4.1.3　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。
其它错误：其他操作出错情况。

7.2.4.2　训练逻辑回归模型

7.2.4.2.1　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模型结构体。

7.2.4.2.2　接口参数

训练逻辑回归模型函数前向接口应符合表28，C代码示例见附录A2.4.2。

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

7.2.4.2.3　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。
其它错误：其他操作出错情况。

7.2.4.3　使用逻辑回归模型进行预测

7.2.4.3.1　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。

7.2.4.3.2　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代码示例见附录A2.4.3。

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]

7.2.4.3.3　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。
其它错误：其他操作出错情况。

7.2.4.4　销毁逻辑回归模型

7.2.4.4.1　功能

销毁某个指定的逻辑回归模型。

7.2.4.4.2　接口参数

销毁逻辑回归模型函数前向接口应符合表30，C代码示例见附录A2.4.4。

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

7.2.4.4.3　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。
其它错误：其他操作出错情况。

7.2.4.5　保存逻辑回归模型

7.2.4.5.1　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。

7.2.4.5.2　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

7.2.4.5.3　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。
其它错误：其他操作出错情况。

7.2.4.6　加载逻辑回归模型

7.2.4.6.1　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。

7.2.4.6.2　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

表 32　加载逻辑回归模型参数列表（续）

参数名	类型	描述
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

7.2.4.6.3　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。
其它错误：其他操作出错情况。

7.2.5　模型评估

7.2.5.1　均方误差

7.2.5.1.1　功能

计算真实值和预测值的均方误差。见式（1）：

$$RMS=\frac{1}{n}\sum_{i=1}^{n}(Y_i-\hat{Y}_i)^2 \quad \text{.....（1）}$$

式中：
RMS——输出张量；

Y_i——第一个输入张量的第i个元素；

$\hat{Y}_i$——第二个输入张量的第i个元素。

7.2.5.1.2　接口参数

均方误差函数前向接口应符合表33，C代码示例见附录A2.5.1。

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

7.2.5.1.3　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。
其它错误：其他操作出错情况。

7.2.5.2　最大误差

7.2.5.2.1　功能

计算真实值和预测值的最大误差，见式（2）：

$$error = \max(|Y_i - \hat{Y}_i|) \quad (2)$$

式中：

error——输出张量；

Y_i——第一个输入张量的第i个元素；

$\hat{Y}_i$——第二个输入张量的第i个元素。

7.2.5.2.2 接口参数

最大误差函数前向接口应符合表34，C代码示例见附录A2.5.2。

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

7.2.5.2.3 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。

其它错误：其他操作出错情况。

7.2.5.3 R2 得分

7.2.5.3.1 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）：

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} \quad (3)$$

式中：

Y_i——第一个输入张量的第i个元素；

$\hat{Y}_i$——第二个输入张量的第i个元素；

$\bar{Y}$——第一个输入张量取平均值；

R——输出张量。

7.2.5.3.2 接口参数

R2得分函数前向接口应符合表35，C代码示例见附录A2.5.3。

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

7.2.5.3.3　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。
其它错误：其他操作出错情况。

7.2.5.4　正确性得分

7.2.5.4.1　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）：

$$\mathrm{acc}=\frac{1}{n}\left(\sum_{i=1}^{n}1\left(Y_i=\widehat{Y}_i\right)\right) \quad \text{（4）}$$

式中：

Y_i——第一个输入张量的第i个元素，即第i个样本的真实值；

$\widehat{Y}_i$——第二个输入张量的第i个元素，即第i个样本的预测值；

n——样本个数；
acc——输出张量。

7.2.5.4.2　接口参数

正确性得分函数前向接口应符合表36，C代码示例见附录A2.5.4。

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

7.2.5.4.3　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。
其它错误：其他操作出错情况。

7.2.6　主成分分析算法

7.2.6.1　创建主成分分析模型

7.2.6.1.1 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。

7.2.6.1.2 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代码示例见附录A2.6.1。

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

7.2.6.1.3 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。
其它错误：其他操作出错情况。

7.2.6.2 训练 PCA 主成分分析模型

7.2.6.2.1 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模型结构体。

7.2.6.2.2 接口参数

训练PCA主成分分析模型函数前向接口应符合表38，C代码示例见附录A2.6.2。

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

7.2.6.2.3 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。
其它错误：其他操作出错情况。

7.2.6.3 使用主成分分析模型进行降维

7.2.6.3.1 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。

7.2.6.3.2 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代码示例见附录A2.6.3。

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

7.2.6.3.3 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。
其它错误：其他操作出错情况。

7.2.6.4 销毁主成分分析模型

7.2.6.4.1 功能

销毁某个指定的 PCA 模型。

7.2.6.4.2 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代码示例见附录A2.6.4。

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

7.2.6.4.3 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。
其它错误：其他操作出错情况。

7.2.6.5 保存 PCA 模型

7.2.6.5.1 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。

7.2.6.5.2 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

7.2.6.5.3　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。
其它错误：其他操作出错情况。

7.2.6.6　加载 PCA 模型

7.2.6.6.1　功能

加载一个保存好的 PCA 模型，用于后续预测任务中。

7.2.6.6.2　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

7.2.6.6.3　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。
其它错误：其他操作出错情况。

7.2.7　线性判别分析算法

7.2.7.1　创建线性判别分析模型

7.2.7.1.1　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。

7.2.7.1.2　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代码示例见附录A2.7.1。

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

7.2.7.1.3　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。
其它错误：其他操作出错情况。

7.2.7.2　训练线性判别分析模型

7.2.7.2.1　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模型结构体。

7.2.7.2.2　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代码示例见附录A2.7.2。

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

7.2.7.2.3　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。
其它错误：其他操作出错情况。

7.2.7.3　使用线性判别分析模型进行降维

7.2.7.3.1　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。

7.2.7.3.2　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代码示例见附录A2.7.3。

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

7.2.7.3.3　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。
其它错误：其他操作出错情况。

7.2.7.4　使用线性判别分析模型进行分类

7.2.7.4.1　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。

7.2.7.4.2　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代码示例见附录A2.7.4。

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]

7.2.7.4.3　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。
其它错误：其他操作出错情况。

7.2.7.5　销毁线性判别分析模型

7.2.7.5.1　功能

销毁某个指定的 LDA 模型。

7.2.7.5.2　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代码示例见附录A2.7.5。

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

7.2.7.5.3　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。
其它错误：其他操作出错情况。

7.2.7.6　保存 LDA 模型

7.2.7.6.1　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。

7.2.7.6.2　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

7.2.7.6.3　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。
其它错误：其他操作出错情况。

7.2.7.7　加载 LDA 模型

7.2.7.7.1　功能

加载一个保存好的 LDA 模型，用于后续预测任务中。

7.2.7.7.2　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

表 49　加载 LDA 模型参数列表（续）

参数名	类型	描述
LDA模型	输入输出	表示存储加载结果的LDA模型

7.2.7.7.3　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。
其它错误：其他操作出错情况。

7.2.8　高斯朴素贝叶斯算法

7.2.8.1　创建高斯朴素贝叶斯模型

7.2.8.1.1　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。

7.2.8.1.2　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代码示例见附录A2.8.1。

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

7.2.8.1.3　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。
其它错误：其他操作出错情况。

7.2.8.2　训练高斯朴素贝叶斯模型

7.2.8.2.1　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模型结构体。

7.2.8.2.2　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代码示例见附录A2.8.2。

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

7.2.8.2.3　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。
其它错误：其他操作出错情况。

7.2.8.3　使用高斯朴素贝叶斯模型进行预测

7.2.8.3.1　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。

7.2.8.3.2　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代码示例见附录A2.8.3。

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]

7.2.8.3.3　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。
其它错误：其他操作出错情况。

7.2.8.4　销毁高斯朴素贝叶斯模型

7.2.8.4.1　功能

销毁某个指定的高斯朴素贝叶斯模型。

7.2.8.4.2　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代码示例见附录A2.8.4。

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

7.2.8.4.3　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。
其它错误：其他操作出错情况。

7.2.8.5　保存 GNB 模型

7.2.8.5.1　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。

7.2.8.5.2　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

7.2.8.5.3　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。
其它错误：其他操作出错情况。

7.2.8.6　加载 GNB 模型

7.2.8.6.1　功能

加载一个保存好的 GNB 模型，用于后续预测任务中。

7.2.8.6.2　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

表 55　加载 GNB 模型参数列表（续）

参数名	类型	描述
GNB模型	输入输出	表示存储加载结果的GNB模型

7.2.8.6.3　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。
其它错误：其他操作出错情况。

7.2.9　决策树分类模型

7.2.9.1　创建决策树分类器模型

7.2.9.1.1　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。

7.2.9.1.2　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代码示例见附录A2.9.1。

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

7.2.9.1.3　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。
其它错误：其他操作出错情况。

7.2.9.2　训练决策树分类模型

7.2.9.2.1　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模型结构体。

7.2.9.2.2　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代码示例见附录A2.9.2。

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

7.2.9.2.3　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。
其它错误：其他操作出错情况。

7.2.9.3　使用决策树分类模型进行分类

7.2.9.3.1　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。

7.2.9.3.2　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代码示例见附录A2.9.3。

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]

7.2.9.3.3　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。
其它错误：其他操作出错情况。

7.2.9.4　销毁决策树分类模型

7.2.9.4.1　功能

销毁某个指定的决策树分类模型。

7.2.9.4.2　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代码示例见附录A2.9.4。

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

7.2.9.4.3　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。
其它错误：其他操作出错情况。

7.2.9.5　保存决策树分类模型

7.2.9.5.1　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。

7.2.9.5.2　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

7.2.9.5.3　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。
其它错误：其他操作出错情况。

7.2.9.6　加载决策树分类模型

7.2.9.6.1　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。

7.2.9.6.2　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
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

7.2.9.6.3 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。
其它错误：其他操作出错情况。

7.2.10 决策树回归模型

7.2.10.1 创建决策树回归模型

7.2.10.1.1 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。

7.2.10.1.2 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代码示例见附录A2.10.1。

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

7.2.10.1.3 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。
其它错误：其他操作出错情况。

7.2.10.2 训练决策树回归模型

7.2.10.2.1 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模型结构体。

7.2.10.2.2 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代码示例见附录A2.10.2。

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

7.2.10.2.3　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。
其它错误：其他操作出错情况。

7.2.10.3　使用决策树回归模型进行回归

7.2.10.3.1　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。

7.2.10.3.2　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代码示例见附录A2.10.3。

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]

7.2.10.3.3　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。
其它错误：其他操作出错情况。

7.2.10.4　销毁决策树回归模型

7.2.10.4.1　功能

销毁某个指定的决策树回归模型。

7.2.10.4.2　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代码示例见附录A2.10.4。

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

7.2.10.4.3　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。
其它错误：其他操作出错情况。

7.2.10.5　保存决策树回归模型

7.2.10.5.1　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。

7.2.10.5.2　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

7.2.10.5.3　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。
其它错误：其他操作出错情况。

7.2.10.6　加载决策树回归模型

7.2.10.6.1　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。

7.2.10.6.2　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

表 67　加载决策树回归模型参数列表（续）

参数名	类型	描述
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

7.2.10.6.3　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。
其它错误：其他操作出错情况。

7.2.11　随机森林分类模型

7.2.11.1　创建随机森林分类模型

7.2.11.1.1　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。

7.2.11.1.2　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代码示例见附录A2.11.1。

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

7.2.11.1.3　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。
其它错误：其他操作出错情况。

7.2.11.2　训练随机森林分类模型

7.2.11.2.1　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模型结构体。

7.2.11.2.2 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代码示例见附录A2.11.2。

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

7.2.11.2.3 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。
其它错误：其他操作出错情况。

7.2.11.3 使用随机森林分类模型进行分类

7.2.11.3.1 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。

7.2.11.3.2 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代码示例见附录A2.11.3。

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]

7.2.11.3.3 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。
其它错误：其他操作出错情况。

7.2.11.4 销毁随机森林分类模型

7.2.11.4.1 功能

销毁某个指定的随机森林分类模型。

7.2.11.4.2 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代码示例见附录A2.11.4。

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

7.2.11.4.3 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。
其它错误：其他操作出错情况。

7.2.11.5 保存随机森林分类模型

7.2.11.5.1 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。

7.2.11.5.2 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

7.2.11.5.3 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。
其它错误：其他操作出错情况。

7.2.11.6 加载随机森林分类模型

7.2.11.6.1 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。

7.2.11.6.2 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

7.2.11.6.3　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。
其它错误：其他操作出错情况。

7.2.12　随机森林回归模型

7.2.12.1　创建随机森林回归模型

7.2.12.1.1　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。

7.2.12.1.2　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代码示例见附录A2.12.1。

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

7.2.12.1.3　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。
其它错误：其他操作出错情况。

7.2.12.2　训练随机森林回归模型

7.2.12.2.1　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模型结构体。

7.2.12.2.2 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代码示例见附录A2.12.2。

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

7.2.12.2.3 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。
其它错误：其他操作出错情况。

7.2.12.3 使用随机森林回归模型进行回归

7.2.12.3.1 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。

7.2.12.3.2 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代码示例见附录A2.12.3。

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]

7.2.12.3.3 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。
其它错误：其他操作出错情况。

7.2.12.4 销毁随机森林回归模型

7.2.12.4.1 功能

销毁某个指定的随机森林回归模型。

7.2.12.4.2 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代码示例见附录A2.12.4。

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

7.2.12.4.3 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。
其它错误：其他操作出错情况。

7.2.12.5 保存随机森林回归模型

7.2.12.5.1 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。

7.2.12.5.2 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

7.2.12.5.3 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。
其它错误：其他操作出错情况。

7.2.12.6 加载随机森林回归模型

7.2.12.6.1 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。

7.2.12.6.2 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

7.2.12.6.3 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。
其它错误：其他操作出错情况。

7.2.13 自适应提升分类模型

7.2.13.1 创建自适应提升分类模型

7.2.13.1.1 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。

7.2.13.1.2 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代码示例见附录A2.13.1。

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

7.2.13.1.3 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。
其它错误：其他操作出错情况。

7.2.13.2 训练自适应提升分类模型

7.2.13.2.1 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模型结构体。

7.2.13.2.2 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代码示例见附录A2.13.2。

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

7.2.13.2.3 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。
其它错误：其他操作出错情况。

7.2.13.3 使用自适应提升分类模型进行分类

7.2.13.3.1 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。

7.2.13.3.2 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代码示例见附录A2.13.3。

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]

7.2.13.3.3 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。
其它错误：其他操作出错情况。

7.2.13.4 销毁自适应提升分类模型

7.2.13.4.1 功能

销毁某个指定的自适应提升分类模型。

7.2.13.4.2 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代码示例见附录A2.13.4。

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

7.2.13.4.3 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。
其它错误：其他操作出错情况。

7.2.13.5 保存自适应提升分类模型

7.2.13.5.1 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。

7.2.13.5.2 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

7.2.13.5.3 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。
其它错误：其他操作出错情况。

7.2.13.6 加载自适应提升分类模型

7.2.13.6.1 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。

7.2.13.6.2 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

7.2.13.6.3 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。
其它错误：其他操作出错情况。

7.2.14 自适应提升回归模型

7.2.14.1 创建自适应提升回归模型

7.2.14.1.1 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。

7.2.14.1.2 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代码示例见附录A2.14.1。

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

7.2.14.1.3 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。
其它错误：其他操作出错情况。

7.2.14.2 训练自适应提升回归模型

7.2.14.2.1 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模型结构体。

7.2.14.2.2 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代码示例见附录A2.14.2。

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

7.2.14.2.3 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。
其它错误：其他操作出错情况。

7.2.14.3 使用自适应提升回归模型进行回归

7.2.14.3.1 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。

7.2.14.3.2 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代码示例见附录A2.14.3。

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]

7.2.14.3.3 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。
其它错误：其他操作出错情况。

7.2.14.4 销毁自适应提升回归模型

7.2.14.4.1 功能

销毁某个指定的自适应提升回归模型。

7.2.14.4.2 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代码示例见附录A2.14.4。

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

7.2.14.4.3 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。
其它错误：其他操作出错情况。

7.2.14.5 保存自适应提升回归模型

7.2.14.5.1 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。

7.2.14.5.2 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

7.2.14.5.3 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。
其它错误：其他操作出错情况。

7.2.14.6 加载自适应提升回归模型

7.2.14.6.1 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。

7.2.14.6.2 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

7.2.14.6.3 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。
其它错误：其他操作出错情况。

附 录 A
（资料性附录）
机器学习类算子接口 C 语言参考定义示例

A.1 数据结构

本文件使用的 C 语言参考定义数据结构与 T/AI 131.1—2025 中的 C 语言参考定义数据结构保持一致。

A.2 机器学习操作

A.2.1 K-最近邻算法

A.2.1.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.1.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.1.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* ref = [[1, 2, 3], [1, 5, 6], [2, 7, 3]]
   label = [[0], [1], [0]]
*/
aitisa_train_knn(ref, label, &model);
```

A.2.1.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* query = [[1.1, 2.1, 3.1]]
*/
Tensor output;
aitisa_predict_knn(query, model, &output);
```

A.2.1.5 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
aitisa_destroy_knn(&model);
```

A.2.2 支持向量机算法

A.2.2.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.2.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.2.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.2.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.2.5 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.2.6 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* ref = [[1, 2, 3], [1, 5, 6], [2, 7, 3]]
   label = [[0], [1], [0]]
*/
aitisa_train_svm(ref, label, &model);
```

A.2.2.7 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* query = [[1.1, 2.1, 3.1]]
*/
Tensor output;
aitisa_predict_svm(query, model, &output);
```

A.2.2.8 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
aitisa_destroy_svm(&model);
```

A.2.3 线性回归算法

A.2.3.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/*
   reg:{type: L2, value: 0.5}
*/
LinearRegModel model;
aitisa_create_linear_reg (reg, &model);
```

A.2.3.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* ref = [[0, 0, 0], [1, 1, 1], [2, 2, 2]]
   label = [[0], [3], [6]]
*/
```

```
aitisa_train_linear_reg (ref, label, &model);
```

A.2.3.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/*
   query = [[1.5, 1.5, 1.5]]
*/
Tensor output;
aitisa_predict_linear_reg (query, model, &output);
```

A.2.3.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
aitisa_destroy_linear_reg (&model);
```

A.2.4 逻辑回归算法

A.2.4.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.4.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* ref = [[0, 0, 0], [-1, -1, -1]]
   label = [[0], [1]]
*/
aitisa_train_logistic_reg (ref, label, &model);
```

A.2.4.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* query = [[1, 1, 1]]
*/
Tensor output;
aitisa_predict_logistic_reg (query, model, &output);
```

A.2.4.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
aitisa_destroy_logistic_reg (&model);
```

A.2.5 模型评估

A.2.5.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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
*/
```

A.2.5.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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
*/
```

A.2.5.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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
*/
```

A.2.5.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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
*/
```

A.2.6 主成分分析

A.2.6.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* k:    10
   solver: { type: auto ,tol = 0.1, iterated_power = 1000}
*/
PcaModel model;
aitisa_create_pca(k, solver, &model);
```

A.2.6.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* ref = [[1, 2, 3], [1, 5, 6], [2, 7, 3]]
*/
aitisa_train_pca(ref, &model);
```

A.2.6.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* query = [[1.1, 2.1, 3.1]]
*/
Tensor output;
aitisa_ dim_reduction_pca (query, model, &output);
```

A.2.6.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
aitisa_destroy_pca (&model);
```

A.2.7 线性判别分析

A.2.7.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* k:    10
   solver: { type: svd ,tol = 1e-4, shrinkage:auto }
*/
LdaModel model;
aitisa_create_lda(k, solver, &model);
```

A.2.7.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* ref = [[0, 0, 0], [-1, -1, -1]]
   label = [[0], [1]]
*/
aitisa_train_lda (ref, label, &model);
```

A.2.7.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* query = [[1.1, 2.1, 3.1]]
*/
Tensor output;
aitisa_ dim_reduction_lda (query, model, &output);
```

A.2.7.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* query = [[1, 1, 1]]
*/
Tensor output;
aitisa_predict_lda (query, model, &output);
```

A.2.7.5 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

aitisa_destroy_lda (&model);

A.2.1 高斯朴素贝叶斯算法

A.2.1.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/*
   priors: [0.2, 0.3, 0.3]
   smooth: 1e-6
*/
GaussianNB model;
aitisa_create_ gaussian_nb (priors, smooth, &model);
```

A.2.1.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/* ref = [[0, 0, 0], [1, 1, 1], [2, 2, 2]]
   label = [[0], [3], [6]]
*/
aitisa_train_ gaussian_nb (ref, label, &model);
```

A.2.1.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/*
   query = [[1.5, 1.5, 1.5]]
*/
Tensor output;
aitisa_predict_ gaussian_nb (query, model, &output);
```

A.2.1.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

aitisa_destroy_ gaussian_nb (&model);

A.2.2 决策树分类模型

A.2.2.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.2.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.2.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/*
   query = [[1.5, 1.5, 1.5]]
*/
Tensor output;
aitisa_predict_ decisiontree_classifier (query, model, &output);
```

A.2.2.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
aitisa_destroy_ decisiontree_classifier (&model);
```

A.2.3 决策树回归模型

A.2.3.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.3.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.3.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/*
   query = [[1.5, 1.5, 1.5]]
*/
Tensor output;
aitisa_predict_ decisiontree_ regressor (query, model, &output);
```

A.2.3.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
aitisa_destroy_ decisiontree_regressor (&model);
```

A.2.4 随机森林分类模型

A.2.4.1 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。
STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.4.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.4.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/*
   query = [[1.5, 1.5, 1.5]]
*/
Tensor output;
aitisa_predict_ randomforest _classifier (query, model, &output);
```

A.2.4.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

aitisa_destroy_ randomforest _classifier (&model);

A.2.5 随机森林回归模型

A.2.5.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.5.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.5.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/*
   query = [[1.5, 1.5, 1.5]]
*/
Tensor output;
aitisa_predict_ randomforest _ regressor (query, model, &output);
```

A.2.5.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
aitisa_destroy_ randomforest _ regressor (&model);
```

A.2.6 自适应提升分类模型

A.2.6.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.6.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.6.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/*
   query = [[1.5, 1.5, 1.5]]
*/
Tensor output;
aitisa_predict_ adaboost _classifier (query, model, &output);
```

A.2.6.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
aitisa_destroy_ adaboost _classifier (&model);
```

A.2.7 自适应提升回归模型

A.2.7.1 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.7.2 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

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);
```

A.2.7.3 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。

示例：

```
/*
    query = [[1.5, 1.5, 1.5]]
*/
Tensor output;
aitisa_predict_ adaboost _ regressor (query, model, &output);
```

A.2.7.4 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。

STATUS_OTHER_FAILED：表示其他操作出错情况。