

# 机器学习模型服务 大作业报告

组长:

软件02班 郭心源 2020012369

组员:

软件02班 谢苑瑜 2020080101

软件02班 范骏捷 2020011593

软件02班 幸若凡 2020012365

软件03班 王麒杰 2020010971

软件01班 顾洋丞 2019010461

清华Git地址:

<https://git.tsinghua.edu.cn/xy-guo20/ml-model-manage-system>

## 人员分工

郭心源: 搭建前后端基本框架、加入PKL格式的模型支持、协调小组工作、测试与汇总。

谢苑瑜: 设计并实现Vue前端页面的显示、完成前端主题替换功能、完成前端Live2D小助手功能。

范骏捷: 利用Axios实现前端向后端通信、实现自定义预处理脚本页面与相关功能。

幸若凡: 加入PMML和ONNX格式的模型支持, 准备测试用模型文件。

王麒杰: 使用MongoDB实现后端数据库相关功能、完成对模型、服务、任务、预处理的查找与更新。

顾洋丞: 完成Flask后端框架的进一步搭建、利用Celery与Redis实现服务的任务安排与处理。

## 前端设计

### 前端地址

本项目采用Vue作为前端框架, 前端运行的默认端口是8080。以本地测试举例, 在浏览器输入“<http://localhost:8080/>”即可进入系统主页。

前端各个页面地址汇总见下表 (MID是模型ID, SID是服务ID, TID是任务ID) :

页面名称	前端地址	页面说明
主页	/	欢迎内容
模型列表页面	/model	查看所有模型信息
模型上传页面	/upload	上传模型
模型详细信息页面	/model/[MID]	查看某一模型信息
模型测试页面	/test/[MID]	测试某一模型
预处理文件载入页面	/preprocess/[MID]	加载某一模型的预处理文件
服务列表页面	/service/[MID]	管理某一模型对应的服务
部署接口页面	/predict/[MID]/[SID]	测试某一服务
批量任务列表页面	/batch/[MID]/[SID]	管理某一服务对应的任务
任务详情页面	/task/[MID]/[SID]/[TID]	查看某一任务信息
404页面	其他	404错误页面

## 页面构成

The screenshot shows the 'Model List Page' (模型列表页面). At the top, there is a dark navigation bar with the TeAM logo, a 'Homepage' link, a 'Model List' link, and a theme switcher. The main content area has a title 'Model List Page' and a table displaying two models:

Model Name	Description	Type	Algorithm	Upload Time	Delete Model
TestModel1	This is a test model	pmmi	sklearn.linear_model._base.LinearRegression	2022/8/19 15:31:06	
TestModel2	This is another test model	onnx	Conv	2022/8/19 15:32:36	

Below the table is a large background graphic featuring gears and a character icon. A tooltip on the character icon says: "点我查看此模型的详情". At the bottom, there is a footer with contact information and a team member list.

以模型列表页面举例，每个前端页面由四部分构成：导航栏、内容、页脚、Live2D小助手。

## 导航栏

导航栏是页面上方的一个区域，显示了系统的Logo（三个斜放的正方形，形状类似卷积网络）、系统的名称（TeAM），“主页”链接、“模型列表”链接、主题更换下拉选择框。

## 内容

不同的页面包含着不同的内容，详见下文。

## 页脚

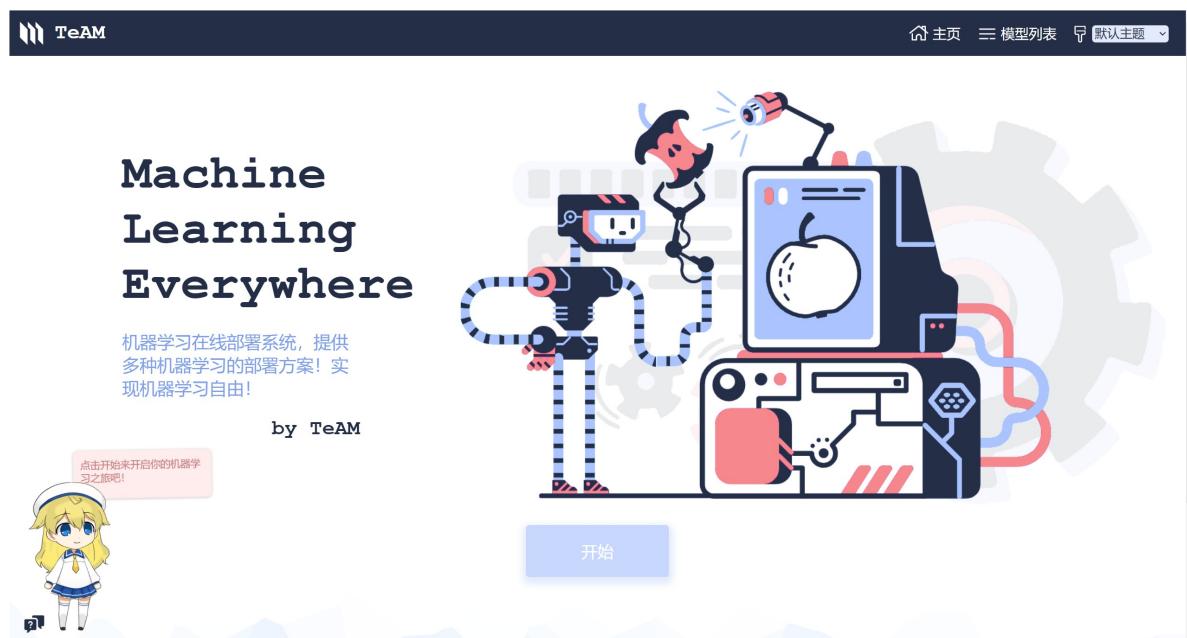
页脚包含大作业团队的一些基本信息，包括电子邮箱、联络电话、友情链接、团队成员。页脚的右下角还有前往项目的清华Git的图标。

## Live2D小助手

为了使用户第一次使用时得到更加清晰的指引，系统配备了一名Live2D“小助手”（见左下角）。当鼠标悬停在不同页面的不同区域时，小助手都会给出相应的提示信息。如果用户已经对系统的使用较为熟练或小助手造成了信息的阻挡，可以点击小助手左下角的按钮关闭。小助手的显示状态会保存到cookie中，这样如果小助手之前被关闭，则下次系统启动时，小助手仍然会处于关闭状态，直到再次点击按钮。

## 页面详情

### 主页



主页的前端地址是“/”。

主页的内容主要是一些欢迎信息，以及一个“开始”按钮。点击该按钮可以进入系统，跳转到模型列表页面。

### 模型列表页面

## 模型列表页面

模型名称	描述	类型	算法	上传时间	删除模型
TestModel1	This is a test model	pmm	sklearn.linear_model._base.LinearRegression	2022/8/19 15:31:06	
TestModel2	This is another test model	onnx	Conv	2022/8/19 15:32:36	



模型列表页面的前端地址是“/model”。

该页面的内容包含一张**模型总表与上传模型按钮**。模型总表显示了系统中目前已导入的模型的信息，包括模型名称、描述、类型、算法、上传时间。表中的每一行的末尾还有删除模型按钮，为了防止用户误触，鼠标悬浮时该按钮会抖动，并且需要双击后才可删除模型。

点击每一行的模型名称可以跳转到该模型对应的**模型详细信息页面**。

点击上传模型按钮可以跳转到**模型上传页面**。

该页面加载时，会向后端地址“/model”发送GET请求，后端会从数据库中获取模型列表页面所需的所有信息并返回。

## 模型上传页面

## 模型导入页面

模型名称	TestModel2
模型描述	This is another test model
模型类型	ONNX
模型文件	<input type="file"/> LeNet.onnx



模型上传页面的前端地址是“/upload”。

该页面的内容是一张表单，表单项包括了**模型名称、模型描述、模型类型、模型文件**，下方有一个提交按钮。

模型名称必须是一个非空字符串，如果未填写的状态下进行上传前端会弹窗提示错误。模型描述是可选的，可以是多行的文本，也可以选择留空。模型类型是一个下拉选择框，包含PMML、ONNX、PKL三项，用户必须选择其中一项。模型文件是一个文件选择框，并且只能上传用户在“模型类型”处选择的类型。

点击上传按钮时，前端会向后端地址“/model”以Form Data的形式发送POST请求，后端收到请求后，会尝试分析模型文件的信息，**若模型文件不合法，则会捕获到错误发送给前端**。会自动分配一个“**模型ID**”（从0开始的正整数），将模型文件保存至后端的models文件夹下，并改名为对应模型ID的名称（后缀不变），方便后续调用，同时会向数据库中添加记录。前端收到回应后，会跳转回**模型列表页面**。

用户点击左上角的返回按钮后，也会返回至**模型列表页面**。

## 模型详细信息页面

The screenshot shows the 'Model Detail Information Page' for 'TestModel2'. At the top, it displays basic model details: name (TestModel2), type (onnx), algorithm (Conv), upload time (2022/8/19 15:34:09), preprocessing (LeNet\_img.py), and description (This is another test model). Below this, there are two tables: 'Input Variable List' and 'Output Variable List'. The 'Input Variable List' table has one row: 'input.1' (tensor(float) [1, 1, 28, 28]). The 'Output Variable List' table has one row: '44' (tensor(float) [1, 10]). At the bottom, three buttons are visible: '进入模型测试' (Enter Model Test), '进入服务列表' (Enter Service List), and '进入预处理加载' (Enter Preprocessing Load).

模型详细信息页面的前端地址是“/model/[model id]”，其中model id是模型ID，若对应的模型不存在，则会提示错误。

在该页面加载时，前端会向后端地址“/model/[model id]”发送GET请求，后端从数据库中查找信息并返回给前端。系统会显示模型名称、模型类型、模型算法、上传时间、预处理项、模型描述。其中**预处理项**在初始时显示为无，需要用户后续上传。

第二个框则显示了该模型的所有输入变量和输出变量，以及对应的变量信息。由于不同格式的模型文件中变量的储存信息不同，因此有些字段可能无法从文件中获取。

下面的三个按钮点击后分别会前往**模型测试页面**、**服务列表页面**和**预处理文件载入页面**。

## 模型测试页面

The screenshot shows the 'Model Testing Page' for 'TestModel2'. It features a 'Input' section where a file named '8.png' is selected, showing a black image of the digit '8'. A note at the bottom left says '(@>o<@)记得要填写所有选项才点击提交!' (Remember to fill in all options before clicking submit!). Below the input is a 'Submit' button. To the right is an 'Output' section displaying a long string of numerical values representing the model's prediction for the input image.

模型测试页面的前端地址是“/test/[model id]”，其中model id是模型ID，若对应的模型不存在，则会提示错误。

该界面加载时，前端会向后端地址“/model/[model id]”发送GET请求，获取当前的模型ID对应的模型名称和输入变量信息。

变量的输入有两种模式：**form模式**和**json模式**，点击当前模式右边的切换图标可进行切换。

当处于**form模式**下，系统会根据当前模型的输入变量生成对应的输入框（又分为文件输入框和文本输入框）。不同的变量类型对应着不同的输入框。

而在**json模式**下，用户可以直接以json格式输入数据（要输入文件可以将文件转换为base64编码）。

若用户想要清空当前的输入，可以双击垃圾桶图标。垃圾桶图标在鼠标悬浮时会抖动。

用户完成输入后，可以点击提交按钮。此时会判断用户输入的合法性（是否非空、是否是json格式），若不合法则报错。通过验证后会向后端地址“/model/[model id]/test”带着当前的输入数据以json格式发送POST请求。后端会根据数据库和模型文件实例化一个模型对象，对输入进行预测，并将结果返回给前端，在本页面的右边的输出框显示。

下图为**form模式**的文本输入框和**json模式**：

The image contains two screenshots of the "Model Test Page".

**Screenshot 1 (Form Mode):**

- Input:** Shows three text input fields for variables CRIM, ZN, and INDUS. A red info box at the bottom left says "(@>o<@)记得要填完所有选项才点击提交哟！".
- Output:** Displays the message "this is output!".

**Screenshot 2 (JSON Mode):**

- Input:** Shows a JSON input field containing a large base64 string. A red info box at the bottom left says "(@>o<@)记得要填完所有选项才点击提交哟！".
- Output:** Displays a large JSON object with numerical values.

Both screenshots include a header with "主页", "模型列表", and "默认主题" buttons, and a footer with contact information: 电子邮箱: yczddgj@126.com, 联络电话: +86-15611462910.

## 预处理文件载入页面

当前预处理脚本文件:

当前预处理脚本描述:

预处理描述

将图像放缩至28x28的大小

预处理文件

选择文件 LeNet\_img.py

(记得要填完所有选项才点击提交哦!)

本网站仅为大作业使用，并无任何商业用途！

由于数据类型的多变性，直接来自文件输入框的图片（或其他输入类型）可能不满足模型的输入格式。因此，必要时用户可以编写**自定义的预处理脚本**，并上传到预处理文件载入界面，载入的文件与模型的输入是绑定的。

该界面的前端地址是“/preprocess/[model id]”，其中model id是模型ID，若对应的模型不存在，则会提示错误。

该页面加载时，会向后端地址“/model/[model id]/preprocess”发送GET请求，获取当前模型的预处理信息，用户可以点击已载入的预处理文件来下载。也可以选择删除已有的预处理文件，这会向后端地址“/model/[model id]/preprocess/delete”发送POST请求，删除当前的预处理文件。用户也可以在下方选择上传新的预处理文件预描述。点击上传后，系统会判断预处理文件是否为空，如不为空则会将预处理文件以FormData的形式POST到后端地址“/model/[model id]/preprocess”，后端程序接收到文件后，会将相关记录添加到数据库中，同时保存该与处理文件到后端文件夹preprocesses文件夹下，文件名会被改为当前的模型ID，并记录文件路径。

在模型进行预测时（无论是模型测试，还是服务的快速返回接口和批量返回接口），系统会检查当前模型是否存在预处理文件，若存在则会将输入数据经过文件中的pre\_process函数处理，然后再输入进模型。

## 服务列表页面

服务名称	创建时间	服务状态	服务次数	平均响应时长	最大响应时长	最小响应时长	删除服务
Service1	2022/8/20 12:20:12	stopped	3	1.505ms	1.753ms	1.177ms	
Service2	2022/8/20 12:20:18	running	4	1.525ms	2.124ms	1.132ms	

新服务名 Service3

(可以添加新的服务哦!)

本网站仅为大作业使用，并无任何商业用途！

电子邮件: yczddgj@126.com  
联系电话: +86-15611462910  
友情链接: 鲍城之都网站  
团队成员:

服务列表页面的前端地址是“/service/[model id]”，其中model id是模型ID，若对应的模型不存在，则会提示错误。

服务列表界面会显示当前模型所对应的所有服务，包括服务名称、创建时间、服务状态、服务次数、平均响应时长、最大响应时长、最小响应时长。

该页面加载时，会向后端地址“/model/[model id]/service”发送GET请求，后端会遍历所有数据库中所有服务的记录和**所有响应的记录**，筛选出当前模型对应的服务和服务对应的响应，**计算服务次数、响应时长相关信息**，并返回给前端。

每一行信息的后面有删除服务按钮，双击后会向后端地址“/model/[model id]/service/[service id]”发送POST请求，并注明将服务删除，后端会找到该服务，并将该服务相关的任务停止并删除，最后删除服务。

每一行的服务状态信息有一个状态切换的按钮，点击可以切换服务状态，这会向后端地址“/model/[model id]/service/[service id]”发送POST请求，并注明是暂停服务还是开始服务。暂停后的服务将不会接受预测任务。

下方可以用来添加新服务，输入新服务的名称，然后点击添加按钮，前端会检查服务名是否非空，而后向后端地址“/model/[model id]/service”发送POST请求，后端会为每个新创建的服务分配**服务ID**（从0开始的正整数），并利用Celery开始对应的服务，同时记录服务相关的信息到数据库中。

点击每一行的服务名称则会跳转到**部署接口页面**。

## 部署接口页面

The screenshot shows the deployment interface page with the following sections:

- Json输入**: A text area containing JSON input data.
- Curl代码**: A text area containing curl command code. A tooltip indicates "一键复制curl代码, 放置Ctrl+C Ctrl+V(<=><=)".
- 输出**: A text area showing the output of the curl command.

部署接口页面的前端地址是“/predict/[model id]/[service id]”，其中model id是模型ID，service id是服务ID，若对应的模型或服务不存在，则会提示错误。

该页面提供一个在前端测试服务的**快速返回接口**的功能。在json输入框内输入合法的json输入串后，点击下方的提交按钮，就会向后端地址“/model/[model id]/service/[service id]/quick”(**快速返回接口**)以当前的json输入发送POST请求。后端接收到对应的POST请求后，会调用当前服务保存的模型进行预测。与模型测试页面不同的是，快速返回接口不会临时实例化一个模型进行测试，而是利用已经创建的服务内部的模型进行测试。运行完毕后，模型会像前端返回模型的输出，前端在右侧的输出框内显示结果。

该页面还提供一个**json输入转curl代码**的功能。在json输入框内输入合法的json输入串后，点击Curl代码右侧的向下的箭头图标，即可生成curl代码。curl代码可以在终端运行，方便生产环境的集成。为方便用户使用，点击复制图标还可以一键对curl代码进行复制，方便用户粘贴到终端。

页面右上角有“前往批量任务列表”按钮，点击该按钮可以跳转到**批量任务列表页面**。

## 批量任务列表页面

The screenshot shows the 'Batch Task List Page'. At the top, there's a header with the TeAM logo, a '主页' (Home) button, a '模型列表' (Model List) button, and a '默认主题' (Default Theme) dropdown. Below the header is a title '批量任务列表页面' (Batch Task List Page), indicating the current service is 'Service2' and the model is 'TestModel2'. A table displays two tasks: '1' and '2', both in the 'finished' state. Below the table is a form for adding new tasks, with a file input field containing 'numbers.zip' and a button labeled '点击添加新任务' (Click to add new task). The background features a large gear icon and a cartoon character on the left.

批量任务列表页面的前端地址是“/batch/[model id]/[service id]”，其中model id是模型ID，service id是服务ID，若对应的模型或服务不存在，则会提示错误。

该页面提供一个访问后端等待返回接口的功能。该页面提供任务管理查看的功能。页面加载时，会向后端地址“/model/[model id]/service/[service id]/task”发送GET请求，获取当前服务的所有任务以及对应的状态。任务的状态分为三种：waiting（等待调度）、running（运行中）、finished（已完成）。

用户可以选择上传文件来添加批量预测的任务，支持的文件格式包括csv和zip，其中csv用来支持文本输入，zip用来支持其他格式的输入（图片等）。点击上传按钮后，会向后端地址“/model/[model id]/service/[service id]/task”以Form Data的形式发送POST请求。后端接收到文件后，会先根据文件的格式解码，处理成模型接受的数据，然后在服务的任务列表中添加一个异步任务。后端同时会分配一个任务ID（从0开始的正整数），返回给前端，前端可以立刻通过任务ID来查看任务的状态，而无需等待任务结束后才收到响应。

点击任务列表中的任务ID即可前往对应的**任务详情页面**。

## 任务详情页面

The screenshot shows the 'Task Detail Page'. At the top, there's a header with the TeAM logo, a '主页' (Home) button, a '模型列表' (Model List) button, and a '默认主题' (Default Theme) dropdown. Below the header is a title '任务详情页面' (Task Detail Page), indicating the current service is 'Service1' and the model is 'TestModel2'. The page displays task details: '当前任务: 1', '任务状态: finished', and '任务结果:'. The results are shown as a JSON array: [ { "filename": "0.png", "result": [ [ [ 6.918611526489258, -7.744033058242501 ] ] ] } ]. The background features a large gear icon and a cartoon character on the left.

任务详情页面的前端地址是“/task/[model id]/[service id]”，其中model id是模型ID，service id是服务ID，task id是任务ID，若对应的模型、服务或任务不存在，则会提示错误。

该页面的功能主要是查看某一个任务的执行情况。页面加载时，会向后端地址“/model/[model id]/service/[service id]/task/[task id]”发送GET请求，后端会调用服务的接口来寻找指定的任务并返回给前端。如果任务状态为finished，则会以json格式显示任务结果，否则不会显示。

## 404页面



当用户在浏览器输入了不属于任何一个页面的地址时，会进入404页面，此时用户可以从导航栏返回[主页](#)或者[模型列表](#)页面。

## 主题功能



为了丰富用户的视觉体验，系统为前端页面提供了5套各具特色的UI主题，分别为：**默认主题、黑白主题、赛博朋克、掘金者、青山**。

用户可以从导航栏的右上角进行主题更换，目前使用的主题会被保存到cookie中，用户下次打开前端页面时主题的选择不会丢失。

主题功能对文本、控件、图标和图片生效。对于文本与空间，更换主题时会更改它们的css颜色值（或背景颜色）；对于图标，主题会对预先保存好的黑色图标进行css的filter处理，使其呈现出不同的契合主题的纯色；而对于图片，我们无法进行简单的动态颜色替换处理，因此我们为每个主题都保存了一套图片用于加载，包括主页图片、页面背景图片、页脚背景图片、404页面图片。

## 后端设计

### 后端接口

本项目采用Flask作为后端框架，后端运行的默认端口是5000。以不同的请求方法访问后端的地址即可调用相应的Flask接口。后端接口采用RESTful API风格设计。

后端各个接口地址与请求方式汇总见下表（MID是模型ID，SID是服务ID，TID是任务ID）：

函数名	后端地址	请求方法	接口说明
getAllModels	/model	GET	获取模型列表
createModel	/model	POST	创建模型
getModelInfo	/model/[MID]	GET	获取模型信息
testModel	/model/[MID]/test	POST	测试模型
deleteModel	/model/[MID]/delete	POST	删除模型
getPreProcess	/model/[MID]/preprocess	GET	获取模型预处理
LoadPreProcess	/model/[MID]/preprocess	POST	加载模型预处理
deletePreProcess	/model/[MID]/preprocess/delete	POST	删除模型预处理
getAllServices	/model/[MID]/service	GET	获取服务列表
createService	/model/[MID]/service	POST	创建服务
changeServiceStatus	/model/[MID]/service/[SID]	POST	更改服务状态 (开启、暂停、 删除)
quickPredict	/model/[MID]/service/[SID]/quick	POST	快速返回接口
batchPredict	/model/[MID]/service/[SID]/task	POST	等待返回接口
getAllTasks	/model/[MID]/service/[SID]/task	GET	获取任务列表
getTaskInfo	/model/[MID]/service/[SID]/task/[TID]	GET	获取任务信息

## 数据库设计

为了精简后端接口的实现，后端单独将数据库相关的操作从后端接口分离出来。后端接口通过调用数据库操作来关系模型、预处理、服务的数据。数据库操作只负责保管模型、预处理、服务的相关静态信息，例如名称、描述、文件路径等。因此，任务的相关实现不被包含在数据库中。本项目采用MongoDB作为后端数据库，可以使用docker来运行MongoDB应用，并暴露端口27017。

数据库由4个list组成：**模型list、预处理list、服务list、响应list**。

模型list存放模型ID、名称、描述、类型、算法、创建时间。

预处理list存放模型ID、文件名、描述、路径、类型。

服务list存放服务ID、名称、创建时间、状态、模型ID。

响应list存放每一次响应的服务ID、开始时间、结束时间、持续时长。

数据库相关函数设计如下表：

函数名	函数描述
getAllModels	从模型list获取所有模型
getModelByID	从模型list查找指定ID模型
addModel	添加模型到模型list
deleteModel	从模型list删除指定ID模型
addPreProcess	添加预处理到预处理list
deletePreProcess	从预处理list删除模型ID对应的预处理
getPreProcessByID	从预处理list获取指定模型ID对应的预处理
getServicesByModel	从服务list获取指定模型ID对应的所有服务
addService	添加服务到服务list
setServiceStatus	在服务list中设定指定服务ID的状态（或删除）
addResponse	添加一次服务响应到响应list

## 模型数据结构设计

后端的模型数据结构采用Model类设计。Model类的作用是解析用户上传的模型文件、实例化为模型测试所需的模型、内置于服务的数据结构中被服务所调用。

下表为成员变量及方法：

Model类成员	类型	描述
model	变量 (object)	模型的核心对象，用于预测
name	变量 (string)	模型的名称
des	变量 (string)	模型的描述
type	变量 (string)	模型的类型
algo	变量 (string)	模型的算法
time	变量 (float)	创建时间
input	变量 (list)	输入变量列表
output	变量 (list)	输出变量列表
pmmInit	方法	PMM类型初始化
onnxInit	方法	ONNX类型初始化
pklInit	方法	PKL类型初始化
predict	方法	使用模型对象进行预测（带预处理）

## 服务与任务数据结构设计

除数据库保存的服务信息外，后端在内存里维护了一个服务列表，用于实现服务相关的操作。服务列表的元素为Service类的对象，可以根据服务ID获取、添加、删除指定的Service类对象。

Service类成员变量及方法如下：

Service类成员	类型	描述
model	变量 (object)	服务内置的模型对象
modelID	变量 (int)	模型对象的ID
pre_processor	变量 (dict)	模型输入预处理的相关信息
id	变量 (int)	服务ID
status	变量 (bool)	服务状态
tasks	变量 (dict)	储存所有的任务信息
proc	变量 (Process)	服务对应的进程
count	变量 (int)	任务数量
predict	方法	使用模型进行快速预测
batch	方法	创建新的批量任务
getResult	方法	获取任务结果
getTaskStatus	方法	获取任务状态
getTasks	方法	获取任务列表
changeStatus	方法	改变服务状态
close	方法	关闭服务

## 参考文献