

# 人工智能之机器学习

## Stacking

主讲人：李老師

## 课程内容

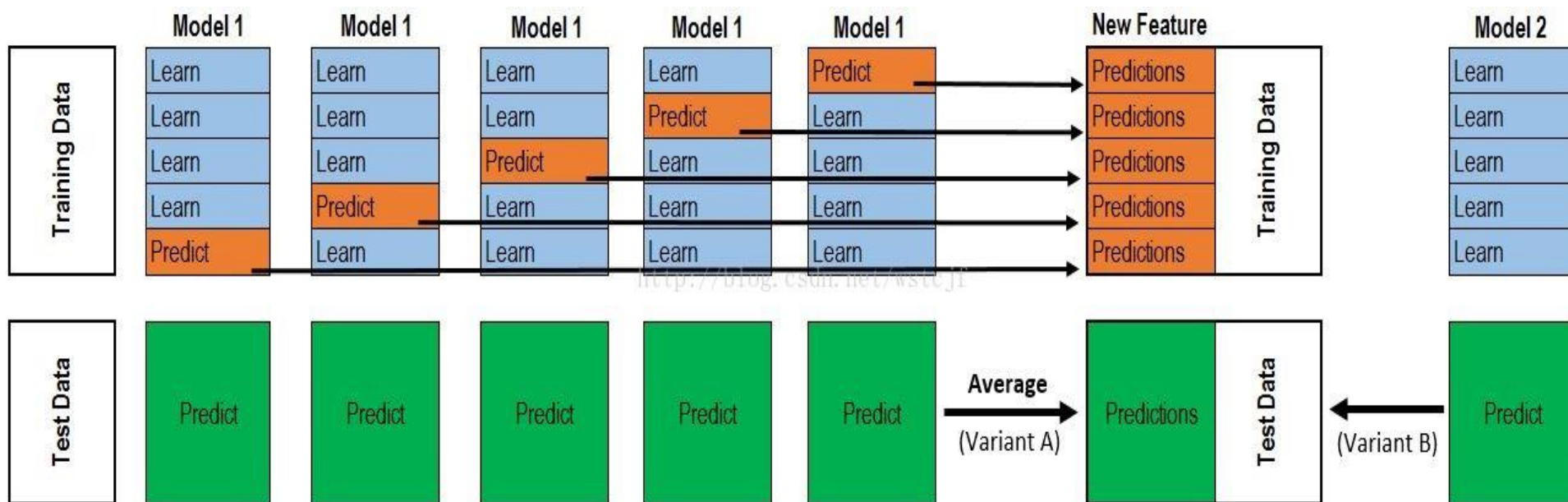
- Stacking概述
- Stacking原理讲解
- Stacking安装
- Stacking项目案例应用

## Stacking概述

- Stacking(有时候也称之为stacked generalization)是指训练一个模型用于组合(combine)其他各个模型。即首先我们先训练多个不同的模型, 然后再以之前训练的各个模型的输出为输入来训练一个模型, 以得到一个最终的输出。
- 如果可以选用任意一个组合算法, 那么理论上, Stacking可以表示前面提到的各种Ensemble方法。然而, 实际中, 我们通常使用单层logistic回归作为组合模型。
- 注意: Stacking有两层, 一层是不同的基学习器(classifiers/regressors), 第二个是用于组合基学习器的元学习器(meta\_classifier/meta\_regressor)

# Stacking原理讲解

- 直观理解



# Stacking原理讲解

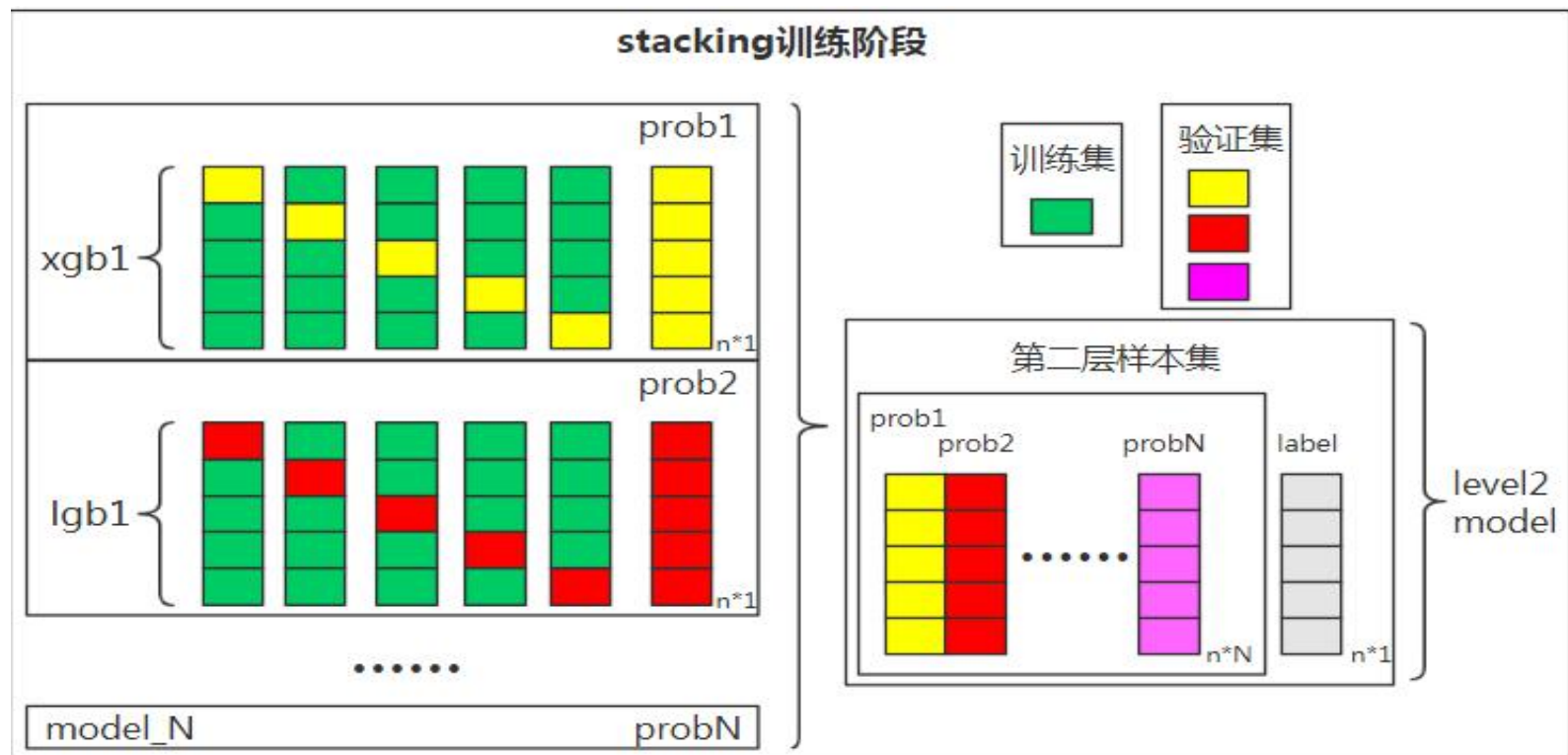
- 图中上半部分是用一个基础模型进行5折交叉验证，如：用XGBoost作为基础模型Model1，5折交叉验证就是先拿出四折作为training learn（蓝色部分），另外一折作为testing predict（橙色部分）。注意：在stacking中此部分数据会用到整个training data。如：假设我们整个training data包含10000行数据，testing data包含2500行数据，那么每一次交叉验证其实就是对training data进行划分，在每一次的交叉验证中training learn将会是8000行，testing predict是2000行。
- 每一次的交叉验证包含两个过程，1. 基于training learn训练模型；2. 基于training learn训练生成的模型对testing predict进行预测。在整个第一次的交叉验证完成之后我们将会得到关于当前testing predict的预测值，这将会是一个一维2000行的数据，记为a1。注意！在这部分操作完成后，我们还要对数据集原来的整个testing data进行预测，这个过程会生成2500个预测值，这部分预测值将会作为下一层模型testing data的一部分，记为b1（绿色部分）。因为我们进行的是5折交叉验证，所以以上提及的过程将会进行五次，最终会生成针对testing data数据预测的5列2000行的数据a1,a2,a3,a4,a5，对testing set的预测会是5列2500行数据b1,b2,b3,b4,b5。

## Stacking原理讲解

- 在完成对Model1的整个步骤之后，我们可以发现 $a_1, a_2, a_3, a_4, a_5$ 其实就是对原来整个training data的预测值，将他们拼凑起来，会形成一个10000行一列的矩阵，记为A1。而对于 $b_1, b_2, b_3, b_4, b_5$ 这部分数据，我们将各部分相加取平均值，得到一个2500行一列的矩阵，记为B1。
- 以上就是stacking中一个模型的完整流程，stacking中同一层通常包含多个模型，假设还有Model2: LR, Model3: RF, Model4: GBDT, Model5: SVM，对于这四个模型，我们可以重复以上的步骤，在整个流程结束之后，我们可以得到新的A2,A3,A4,A5,B2,B3,B4,B5矩阵。
- 在此之后，我们把A1,A2,A3,A4,A5并列合并得到一个10000行五列的矩阵作为新的training data，B1,B2,B3,B4,B5并列合并得到一个2500行五列的矩阵作为新的testing data。让下一层的模型（元学习器），基于他们进一步训练。

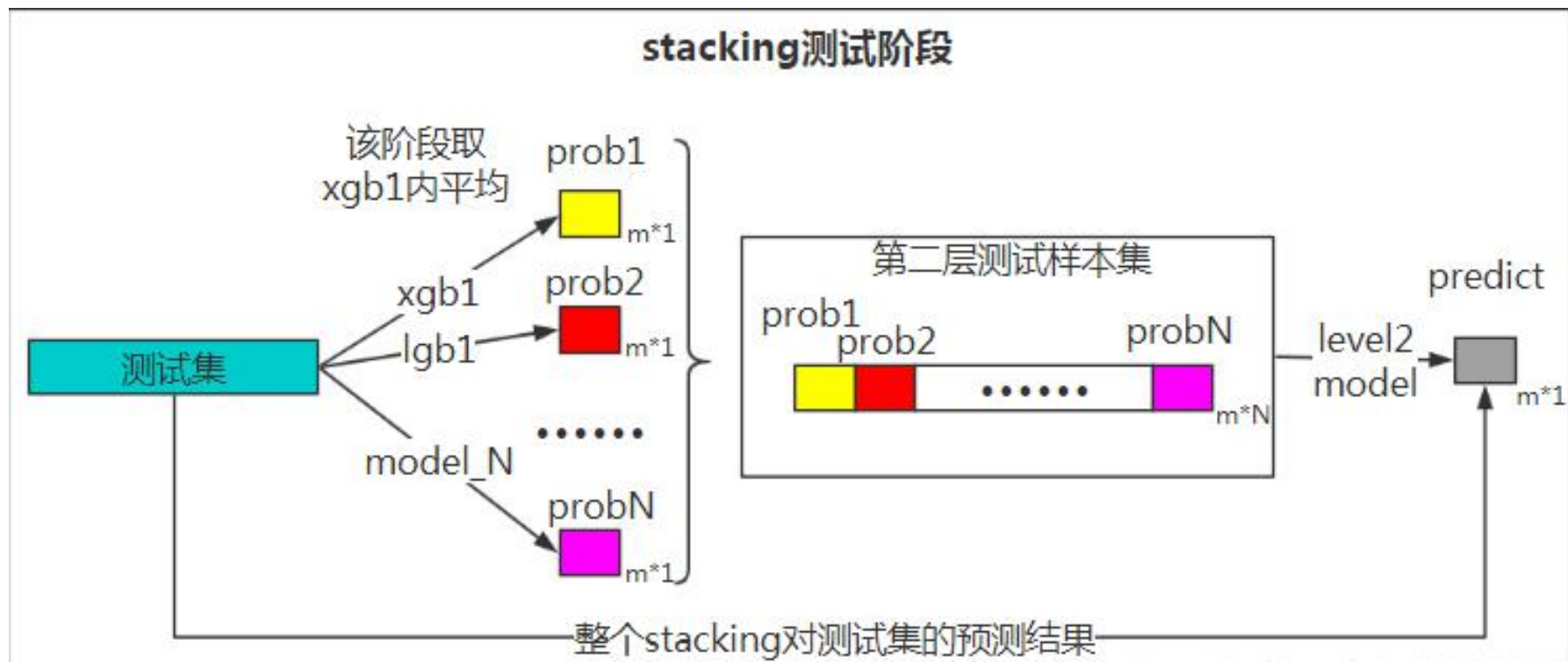
# Stacking原理讲解

- 训练阶段



# Stacking原理讲解

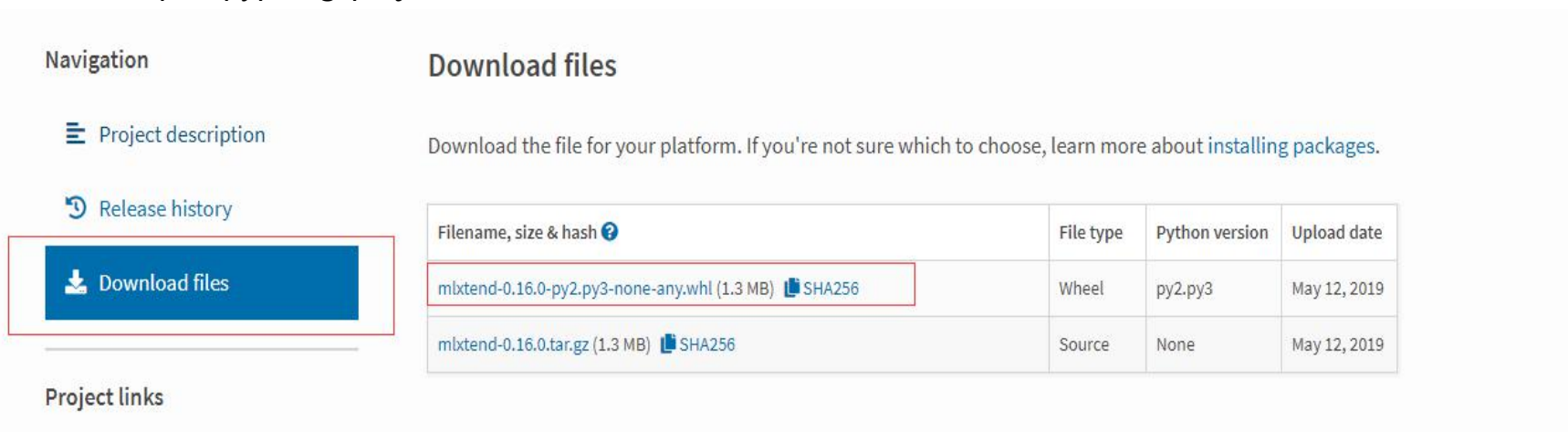
- 预测阶段





# Stacking安装

- Stacking在sklearn库中没有API接口，可以使用mlxtend库
  - <https://pypi.org/project/mlxtend/>
  - <https://rasbt.github.io/mlxtend/>
- 安装方式
  - 下载whl文件安装: <https://pypi.org/project/mlxtend/>



Navigation

- Project description
- Release history
- Download files**

Download files

Download the file for your platform. If you're not sure which to choose, learn more about [installing packages](#).

Filename, size & hash	File type	Python version	Upload date
<a href="#">mlxtend-0.16.0-py2.py3-none-any.whl (1.3 MB)</a> <a href="#">SHA256</a>	Wheel	py2.py3	May 12, 2019
<a href="#">mlxtend-0.16.0.tar.gz (1.3 MB)</a> <a href="#">SHA256</a>	Source	None	May 12, 2019

Project links

- `pip install mlxtend`
- `pip install -i https://pypi.tuna.tsinghua.edu.cn/simple mlxtend`

```
C:\Users\ibf>python
Python 3.6.3 |Anaconda, Inc.| (default, Oct 15 2017, 03:27:45) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import mlxtend
>>> exit()
```

# stacking API

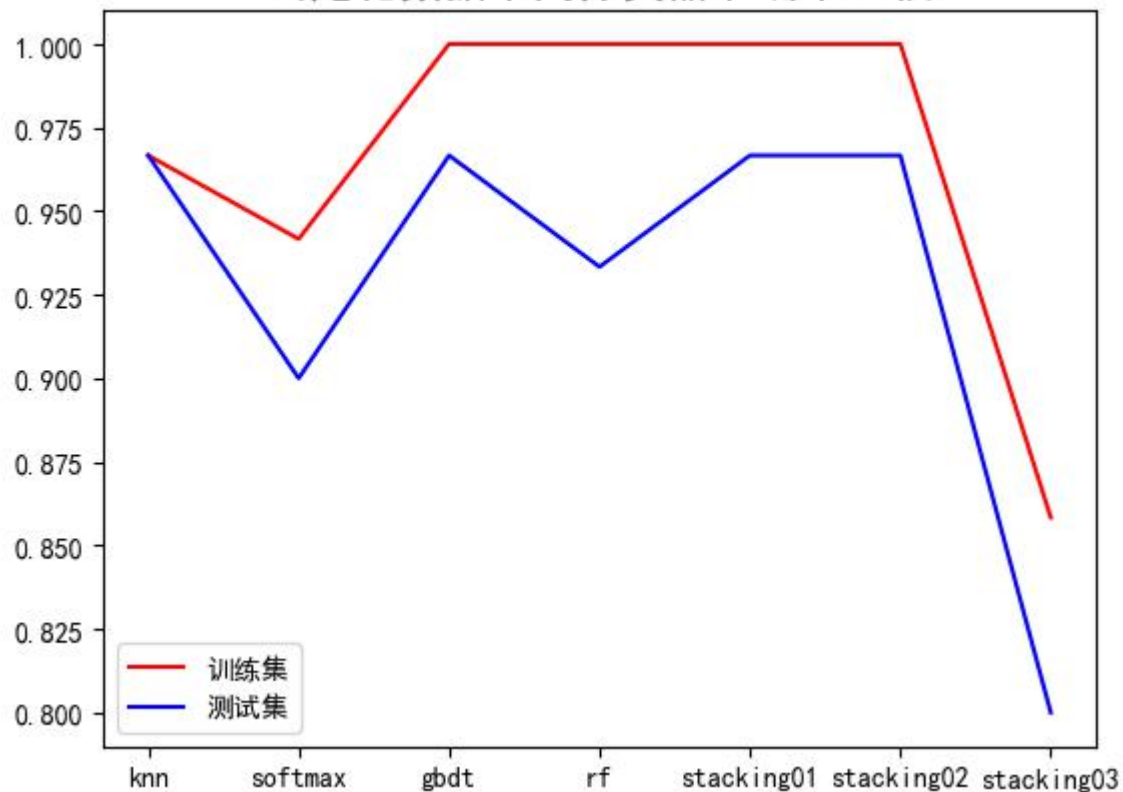
- `StackingClassifier(classifiers, meta_classifier, use_probas=False, average_probas=False, verbose=0, use_features_in_secondary=False)`
- 参数：
  - `classifiers` : 基分类器, 数组形式, `[cl1, cl2, cl3]`. 每个基分类器的属性被存储在类属性 `self.clfs_`.
  - `meta_classifier` : 目标分类器, 即将前面分类器合起来的分类器
  - `use_probas` : `bool` (default: `False`) , 如果设置为`True`, 那么目标分类器的输入就是前面分类输出的类别概率值而不是类别标签
  - `average_probas` : `bool` (default: `False`), 用来设置上一个参数当使用概率值输出的时候是否使用平均值。
  - `verbose` : `int`, optional (default=0)。用来控制使用过程中的日志输出, 当 `verbose = 0`时, 什么也不输出, `verbose = 1`, 输出回归器的序号和名字。 `verbose = 2`, 输出详细的参数信息。 `verbose > 2`, 自动将`verbose`设置为小于等于2的, `verbose -2`.
  - `use_features_in_secondary` : `bool` (default: `False`). 如果设置为`True`, 那么最终的目标分类器就被基分类器产生的数据和最初的数据集同时训练。如果设置为`False`, 最终的分类器只会使用基分类器产生的数据训练。

# stacking API

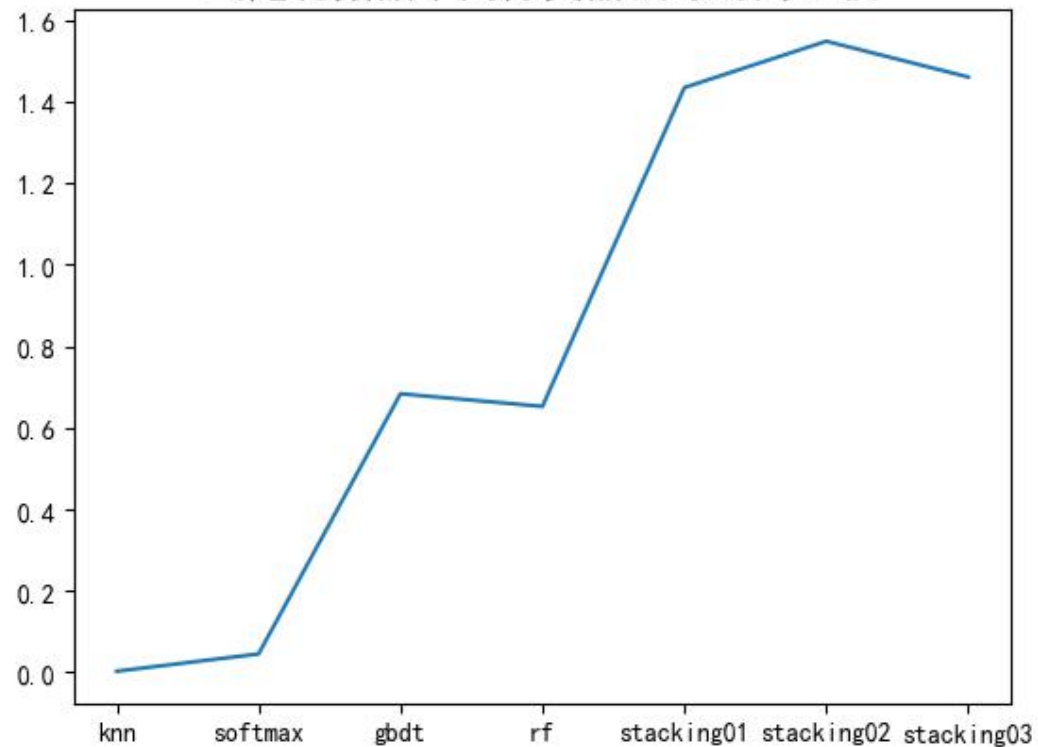
- 属性：
  - `clfs_` : 每个基分类器的属性, list, shape 为 `[n_classifiers]`。
  - `meta_clf_` : 最终目标分类器的属性
- 方法：
  - `fit(X, y)`
  - `fit_transform(X, y=None, fit_params)`
  - `get_params(deep=True)`, 如果是使用sklearn的GridSearch方法, 那么返回分类器的各项参数。
  - `predict(X)`
  - `predict_proba(X)`
  - `score(X, y, sample_weight=None)`, 对于给定数据集和给定label, 返回评价accuracy
  - `set_params(params)`, 设置分类器的参数, `params`的设置方法和sklearn的格式一样
- 注: 回归的API `StackingRegressor(regressors, meta_regressor)` 基本类似

# Stacking项目案例应用

鸢尾花数据不同分类器准确率比较



鸢尾花数据不同分类器训练时间比较



# Stacking项目案例应用

