

机器学习 II

Machine Learning II: sklearn

01

工作流程

02

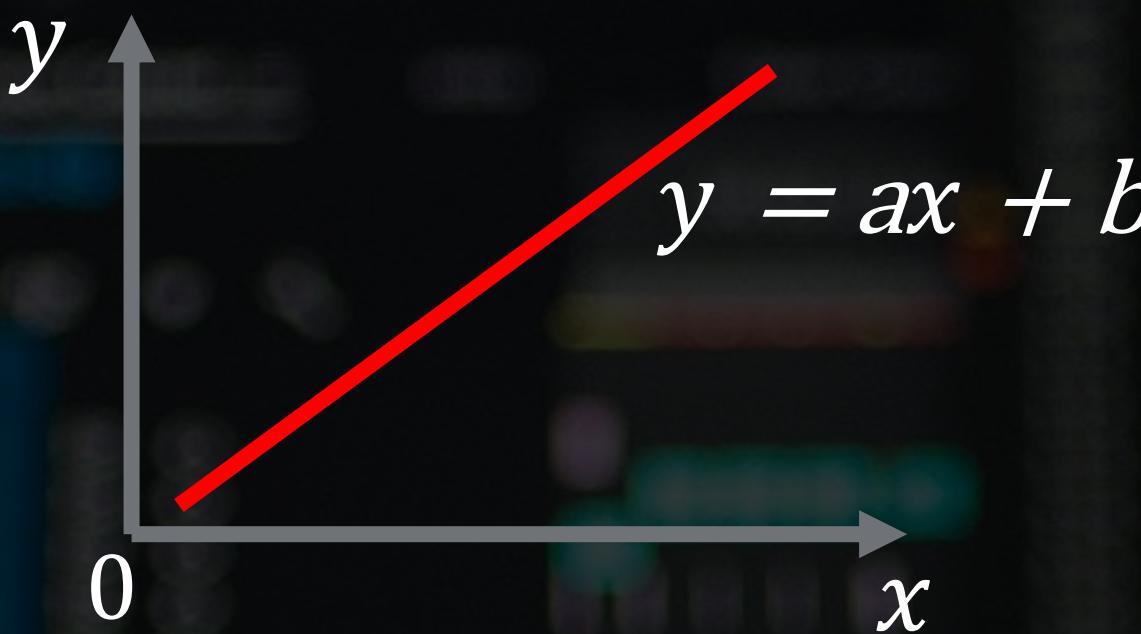
数据探索

3.1

多元线性回归

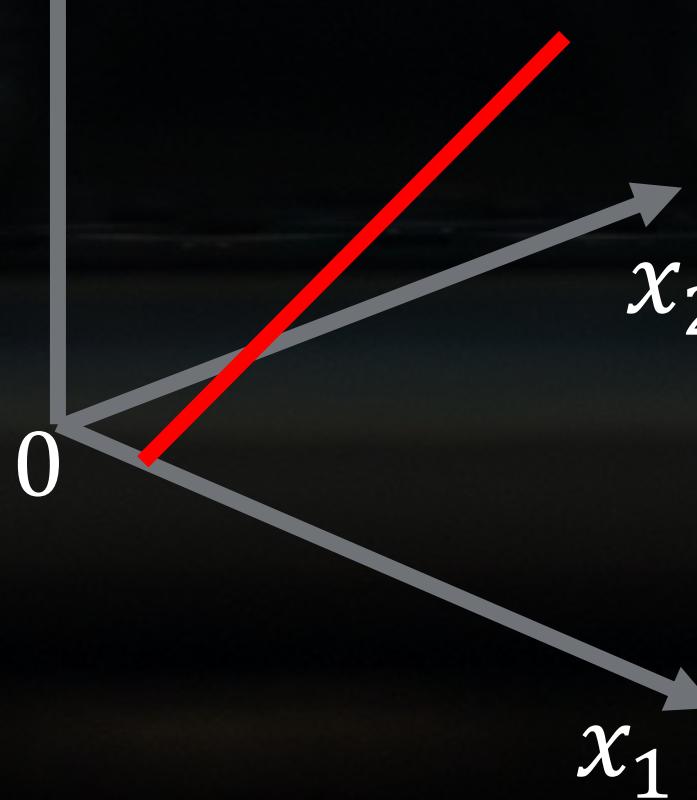
多元线性回归：模型

- 一元线性回归：



$$y = a_1 x_1 + a_2 x_2 + a_3 x_3 + \dots + a_n x_n + b$$

- 多元线性回归：



多元线性回归：损失函数

一元线性回归损失函数：

- $\frac{1}{N} \sum_{i=1}^N (y_i - (ax_i + b))^2$



多元线性回归损失函数：

- $\frac{1}{N} \sum_{i=1}^N (y_i - (a_1 x_1 + a_2 x_2 + \dots + a_n x_n + b))^2$

多元线性回归：矩阵求解

- 假设 $x_0 = 1$ $a_0 = b$, :

$$y = a_1 x_1 + a_2 x_2 + a_3 x_3 + \dots + a_n x_n + b$$



$$y = a_0 x_0 + a_1 x_1 + a_2 x_2 + \dots + a_n x_n$$

多元线性回归：矩阵求解

公式的向量转化：

- 将 $x_0, x_1 \dots x_n$ 表示成为一个长度为 $n + 1$ 的向量 X
- 将参数 $a_0, a_1 \dots a_n$ 表示成为一个长度为 $n+1$ 的向量 A

$$y = a_1 x_1 + a_2 x_2 + a_3 x_3 + \dots + a_n x_n + b$$



$$y = A^T X$$

多元线性回归：矩阵求解

假设建模数据集有m条样本：

- 将自变量 $[x_{00}, x_{01} \dots x_{0n}] \dots [x_{m0}, x_{m1} \dots x_{mn}]$ 表示成为一个长度为 $m * (n + 1)$ 的矩阵X
- 将 $y_1, y_2 \dots y_m$ 表示成为一个长度为m的列向量Y
- 将参数 $a_0, a_1 \dots a_n$ 表示成为一个长度为n+1的列向量A

多元线性回归损失函数：

$$E(a_1, a_2 \dots a_n) = \frac{1}{N} \sum_{i=1}^N (y_i - (a_1 x_1 + a_2 x_2 + \dots + a_n x_n + b))^2$$



$$E(A) = (Y - XA)^T (Y - XA)$$

多元线性回归：矩阵求解

多元线性回归损失函数：

$$E(A) = (Y - XA)^T (Y - XA)$$

- 对参数向量A求导：

$$\frac{\partial E(A)}{\partial A} = 2X^T(Y - XA) = 0$$

- 解出A：

$$A = (X^T X)^{-1} X^T Y$$

推导过程：<https://zhuanlan.zhihu.com/p/33899560>

3.2

惩罚模型

惩罚模型

惩罚（正则化）定义：

通过在模型损失函数中增加一个正则项（惩罚项）来控制模型的复杂度

惩罚项：一般来说都是一个随着模型复杂度增加而增加的单调递增函数

惩罚模型在此处键入公式。

惩罚项（正则化）的形式：

假设一个模型的损失函数为：

$$\frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i))$$

则加了惩罚项的损失函数为

$$\frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i)) + \lambda J(f)$$

优化目标则变成：

$$\min\left(\frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i)) + \lambda J(f)\right)$$

惩罚模型

优化公式：

$$\min\left(\frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i))\right)$$

优化目标：

- 求解参数使得模型误差整体最小

正则化的优化公式：

$$\min\left(\frac{1}{N} \sum_{i=1}^N L(y_i, f(x_i)) + \lambda J(f)\right)$$

正则化的优化目标：

- 求解参数使得模型的误差最小同时
模型的复杂度最低

惩罚模型

惩罚项（正则化）的目的：通过降低模型的复杂度，从而防止过拟合，提高模型的泛化能力

解释一：

- 奥卡姆剃刀原理 (*Occam's Razor*)：“如无必要，勿增实体 (*Entities should not be multiplied unnecessarily*)”
- 解释：能够用简单的方法达到很好的效果，就没有必要使用复杂的方法
- 原理推广：如果简单的模型就能够达到很好的预测效果，就没有必要选择复杂的模型

解释二：

- 在模型中使用更多的自变量，一般情况下都会提升模型在训练数据集上的表现，但同时也会提高模型的复杂度、降低模型在验证集上的泛化能力，造成过拟合。

惩罚模型

常用的惩罚项（正则项）：

以线性回归模型的损失函数为例：

假设线性回归模型需要求解的参数为列向量A，数据集中有N个样本

- L1正则系数：lasso回归

$$E(A) = \frac{1}{N} \sum_{i=1}^N (y_i - (Ax_i))^2 + \lambda ||A||_1$$

- L2正则系数：ridge回归

$$E(A) = \frac{1}{N} \sum_{i=1}^N (y_i - (Ax_i))^2 + \lambda ||A||_2^2$$

惩罚模型

常用的惩罚项（正则项）特性：

L1正则系数：lasso回归：

- L1是模型各个参数的绝对值之和
- L1可以将特征参数约束到0，因此L1会趋向于产生少量的特征，而其他的特征都是0
- L1也因此具有特征筛选的功能（被筛除的特征参数为0）
- L1通过融入少量的特征来防止过拟合

L2正则系数：ridge回归：

- L2是模型各个参数的平方和的开方值。
- L2只能减小特征参数值，让参数接近0，但不能将参数约束到0
- L2通过减少特征的参数值来防止过拟合

3.3

Logistic回归模型

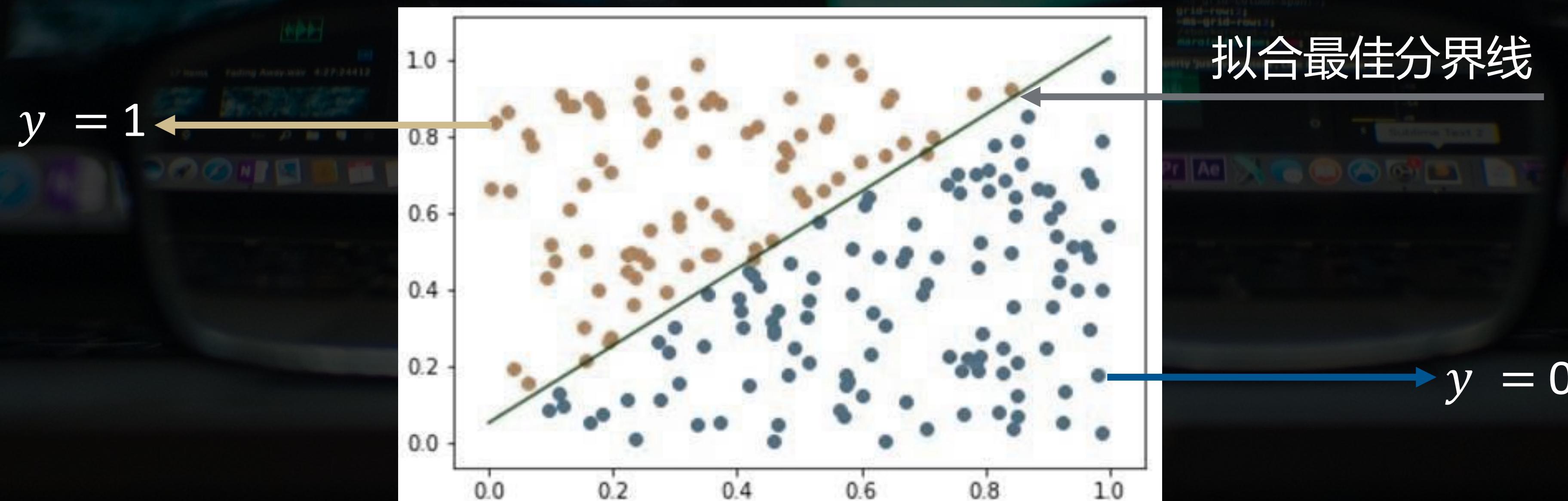
Logistic回归模型

Logistic回归：一个被命名为回归的分类模型

Logistic回归模型

- 主体思想：

通过对数据的分类边界线建立回归公式，从而实现分类



Logistic回归模型

- 线性回归回顾: $y = ax + b$
- 线性回归y值: 连续数值型变量

问题: 如何使用线性回归模型输出离散类别变量?

Logistic回归模型

神奇的激活函数：将连续的数值转化成0或1的输出

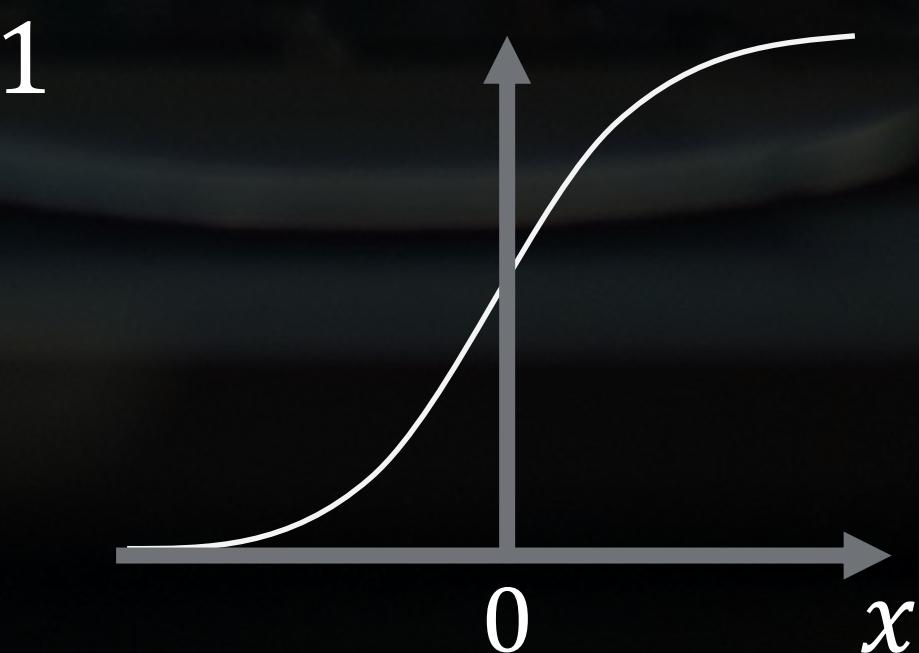
Heaviside函数：

$$\sigma(z) = \begin{cases} 0, & z < \text{threshold} \\ 1, & z \geq \text{threshold} \end{cases}$$



Sigmoid函数：

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



Logistic回归模型

神奇的激活函数：将连续的数值转化成0或1的输出

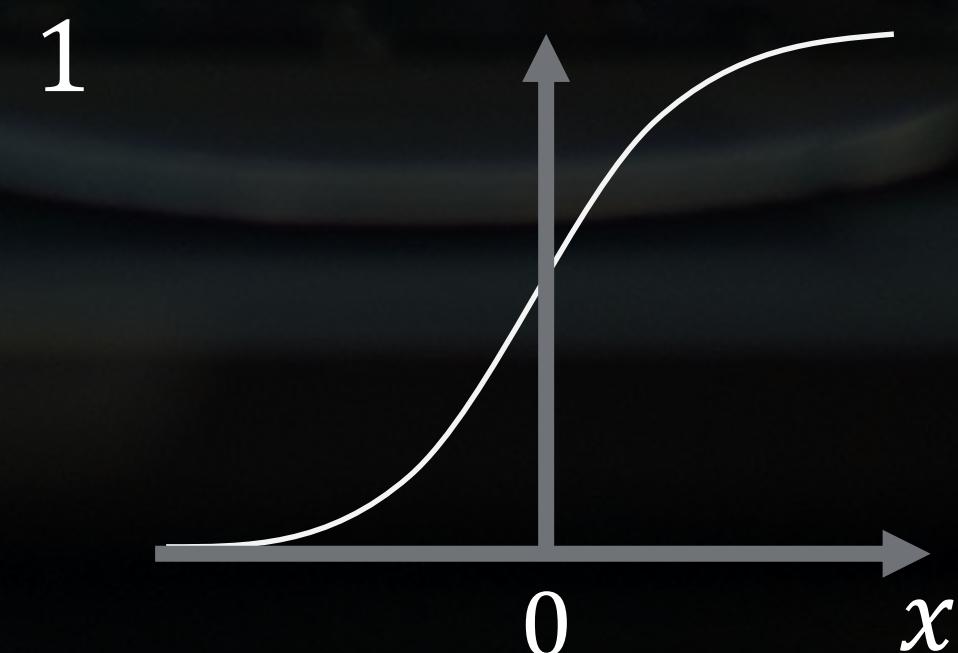
Heaviside函数：

- 0到1的跳跃过程不平滑



Sigmoid函数：

- 0到1的渐变过程平滑



Logistic回归模型

Logistic回归模型实现过程：

1. 将样本特征值与回归系数相乘
2. 再将所有特征值与回归系数的乘积相加
3. 最后将加和代入sigmoid函数
4. 输出一个范围在0-1之间的值
5. 结果大于0.5的样本归入1类， 小于0.5的归入0类

Logistic回归模型

Logistic回归模型公式：

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

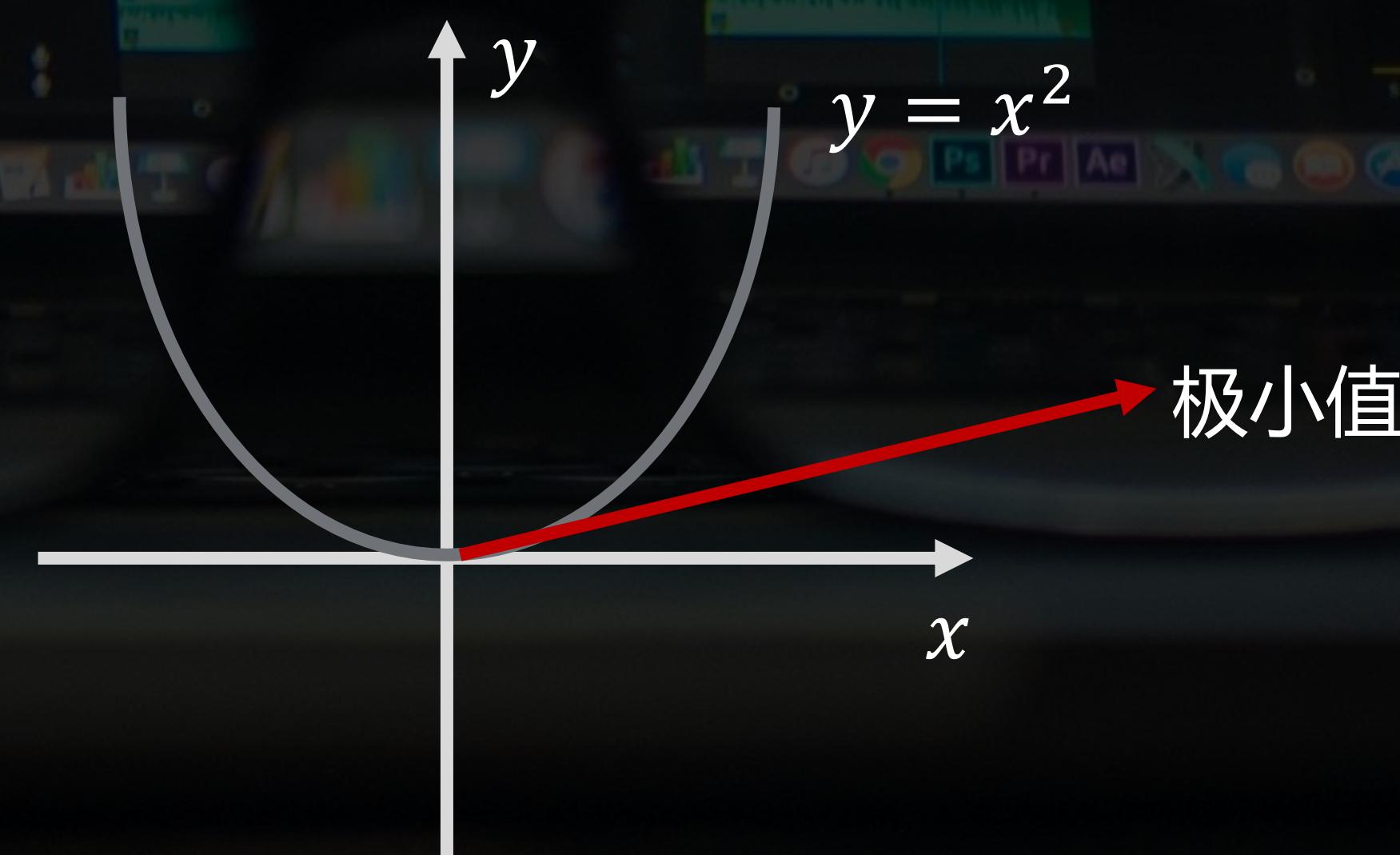
$$z = ax + b$$

逻辑回归：损失函数

回顾：线性回归损失函数：

$$\frac{1}{N} \sum_{i=1}^N (y_i - (ax_i + b))^2$$

回顾：线性回归损失函数最优解最优解图像：



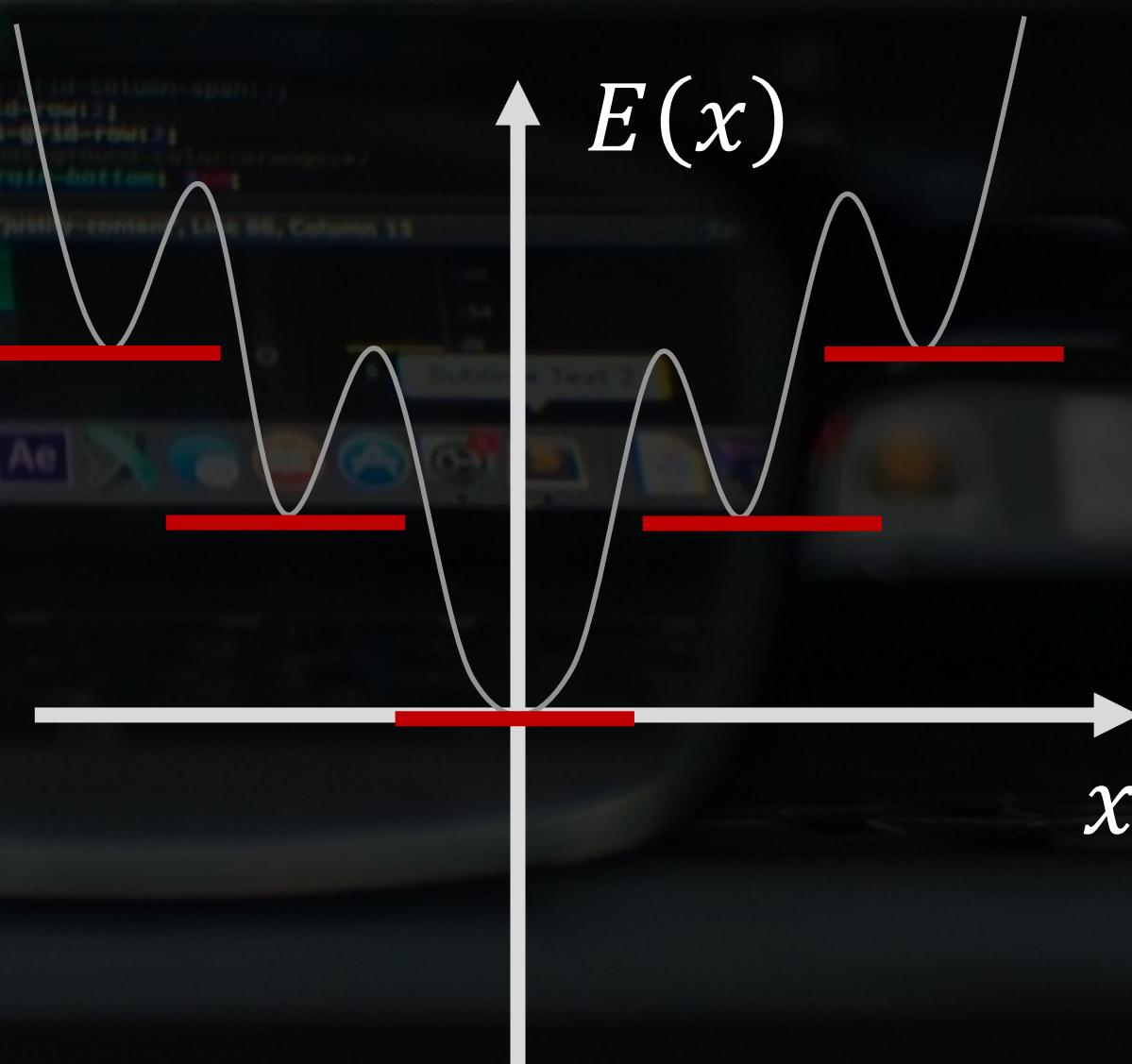
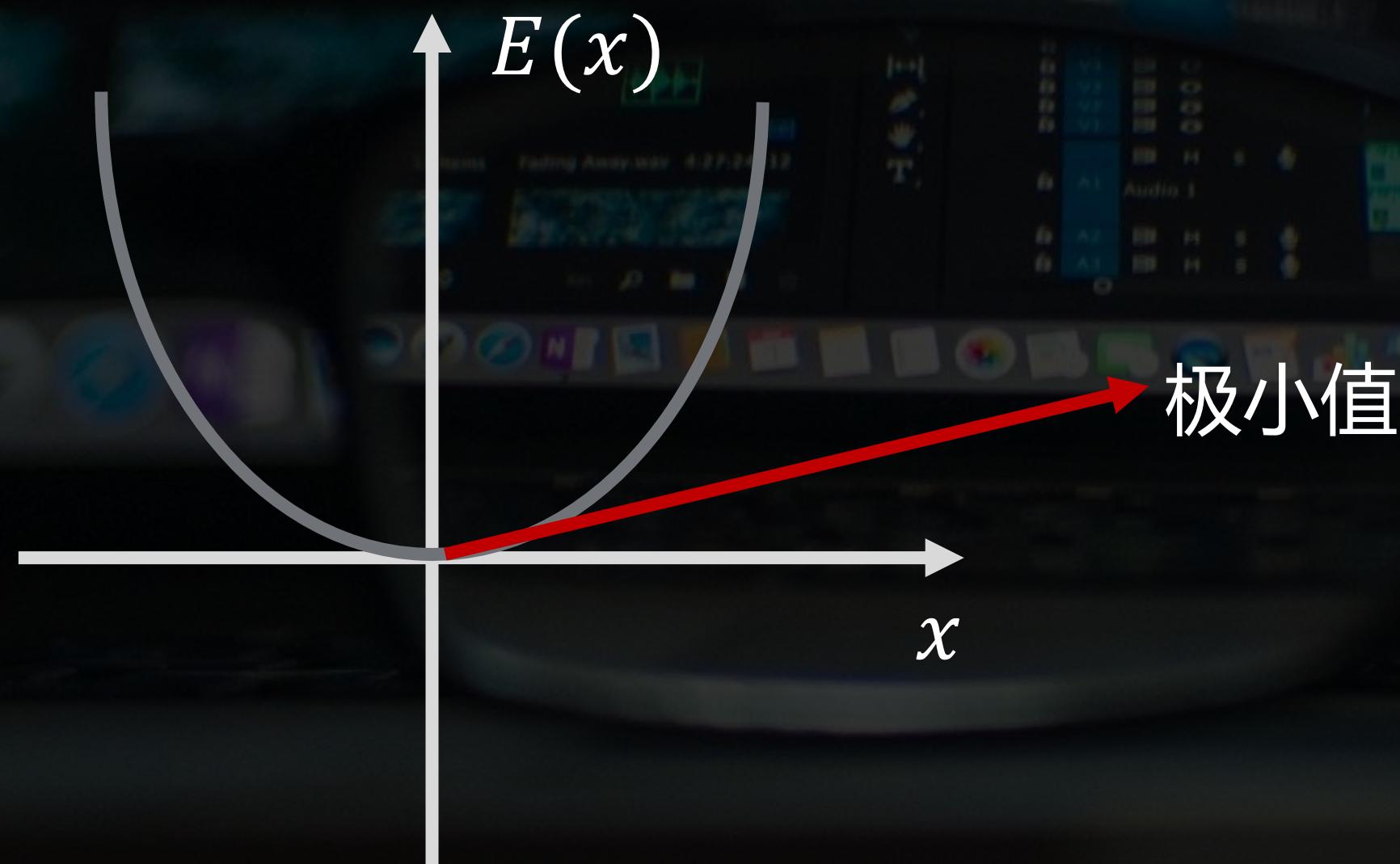
逻辑回归：损失函数

问题：逻辑回归的损失函数是否是如下公式呢？

$$\frac{1}{N} \sum_{i=1}^N \left(y_i - \frac{1}{1 + e^{-(ax_i + b)}} \right)^2$$

逻辑回归：损失函数

思考：逻辑回归的损失函数图像是否还是二次函数？



逻辑回归：损失函数

逻辑回归的损失函数：对数似然损失函数

$$E(\sigma(z)) = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\sigma(z)) + (1 - y_i) \log(1 - \sigma(z))]$$

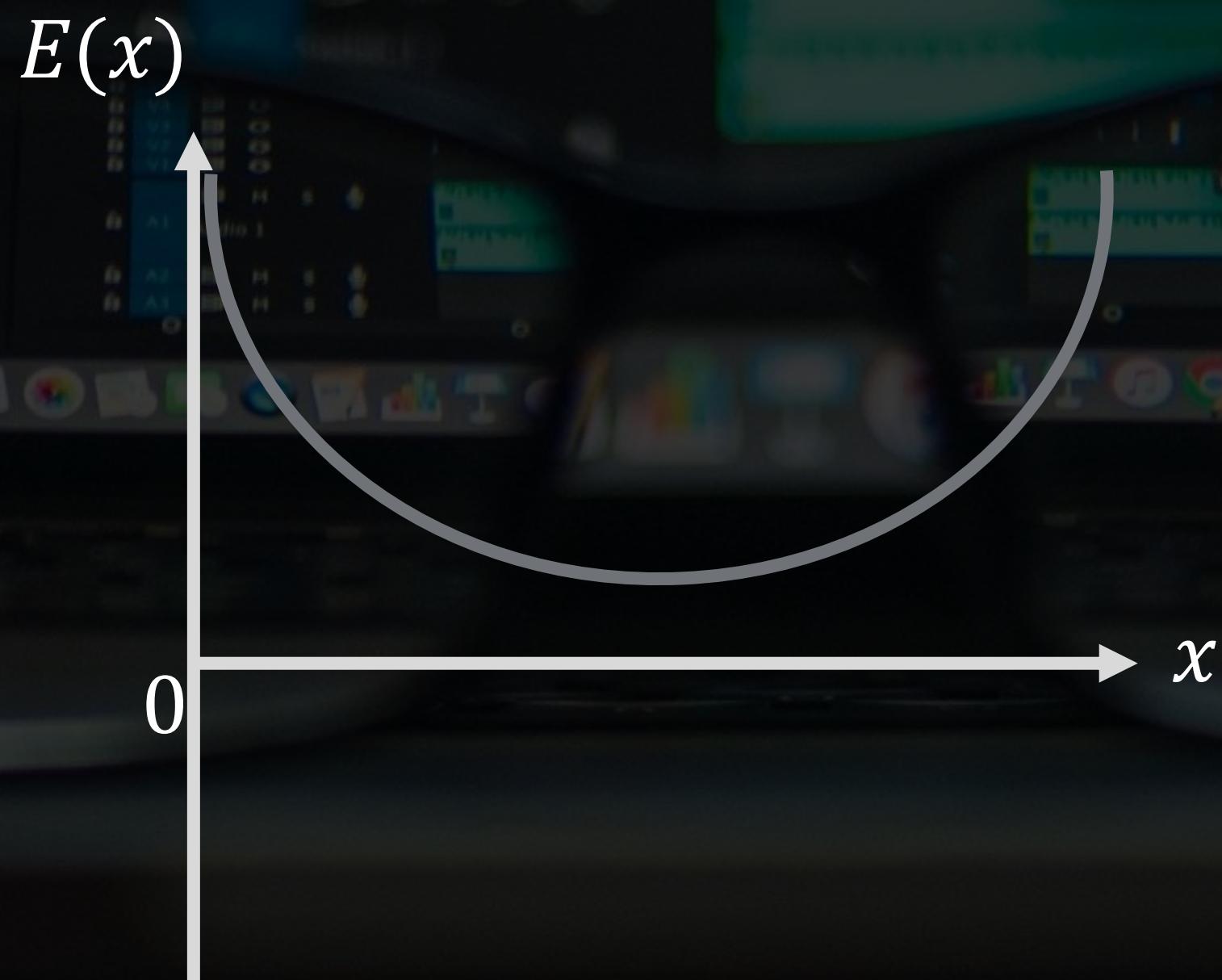
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$z = ax + b$$

推导过程：<https://zhuanlan.zhihu.com/p/44591359>

逻辑回归：损失函数

$$E(a, b) = -\frac{1}{N} \sum_{i=1}^N [y_i \log\left(\frac{1}{1+e^{-(ax_i+b)}}\right) + (1-y_i) \log\left(1 - \frac{1}{1+e^{-(ax_i+b)}}\right)]$$



梯度下降法

梯度下降法：

- 无约束多元函数极值求解方法
- 一种常用的机器学习参数求解法
- 通过迭代得到最小化的损失函数所对应的模型参数

基本思路：

- 在求解目标函数 $E(a)$ 的最小值时， a 沿着梯度下降的方向不断变化求解最小值。



梯度下降法

什么是梯度：

假设优化目标是求解函数 $E(a)$ 的最小值

- 参数 a 的梯度为函数 $E(a)$ 的偏导数： $\frac{\partial E(a)}{\partial a}$
- 因此 a 的迭代公式为：

$$a = a - \alpha \frac{\partial E(a)}{\partial a}$$

* α ：步长

什么是步长：

- 步长是梯度下降迭代的速度控制器
- 步长太小：收敛速度慢
- 步长太大：可能跳过函数最小值，导致发散



逻辑回归的参数求解

回顾：逻辑回归的损失函数

$$E(a, b) = -\frac{1}{N} \sum_{i=1}^N [y_i \log\left(\frac{1}{1+e^{-(ax_i+b)}}\right) + (1-y_i) \log\left(1 - \frac{1}{1+e^{-(ax_i+b)}}\right)]$$

参数求解：梯度下降法：

循环{

$$a := a - \alpha \frac{\partial E(a,b)}{\partial a},$$

$$b := b - \alpha \frac{\partial E(a,b)}{\partial b}$$

}

如何理解梯度下降：<https://zhuanlan.zhihu.com/p/36902908>

04

数据预处理、特征工程模型评价

数据预处理与特征工程

概念：数据预处理与特征工程泛指对训练数据集进行特征增加、删除、变换的方法

目标：通过对训练数据的处理变换，提高模型的训练表现和泛化能力。

类别：

- 特征变换：预处理、标准化、纠偏
- 特征增加与删减：特征降维与变量扩展

数据预处理 与特征工程

更多资源：

<https://zhuanlan.zhihu.com/p/51131210>
<https://zhuanlan.zhihu.com/p/33665963>

模型评价体系

回顾：

模型 (model) : 规律和经验

学习 (learning) : 从数据中总结规律的过程

误差 (error) : 衡量模型准确性的指标

训练集 (教材教辅) : 训练模型的数据集

验证集 (模拟考卷) : 测试机器学习模型泛化能力的数据集

应用数据 (高考) : 模型实际应用场景中的特征集

模型评价体
系

模型评价体系

拓展：

回归模型应该使用哪些评价指标？

分类模型使用的评价指标有何异同？

如何有效划分训练集与验证集，从而能有效地检验模型的经验误差与泛化误差？

模型评价体
系

更多资源：

<https://zhuanlan.zhihu.com/p/36326966>

<https://zhuanlan.zhihu.com/p/39551438>

05

其他机器学习算法

• 5.1 •

CART分类及回归树

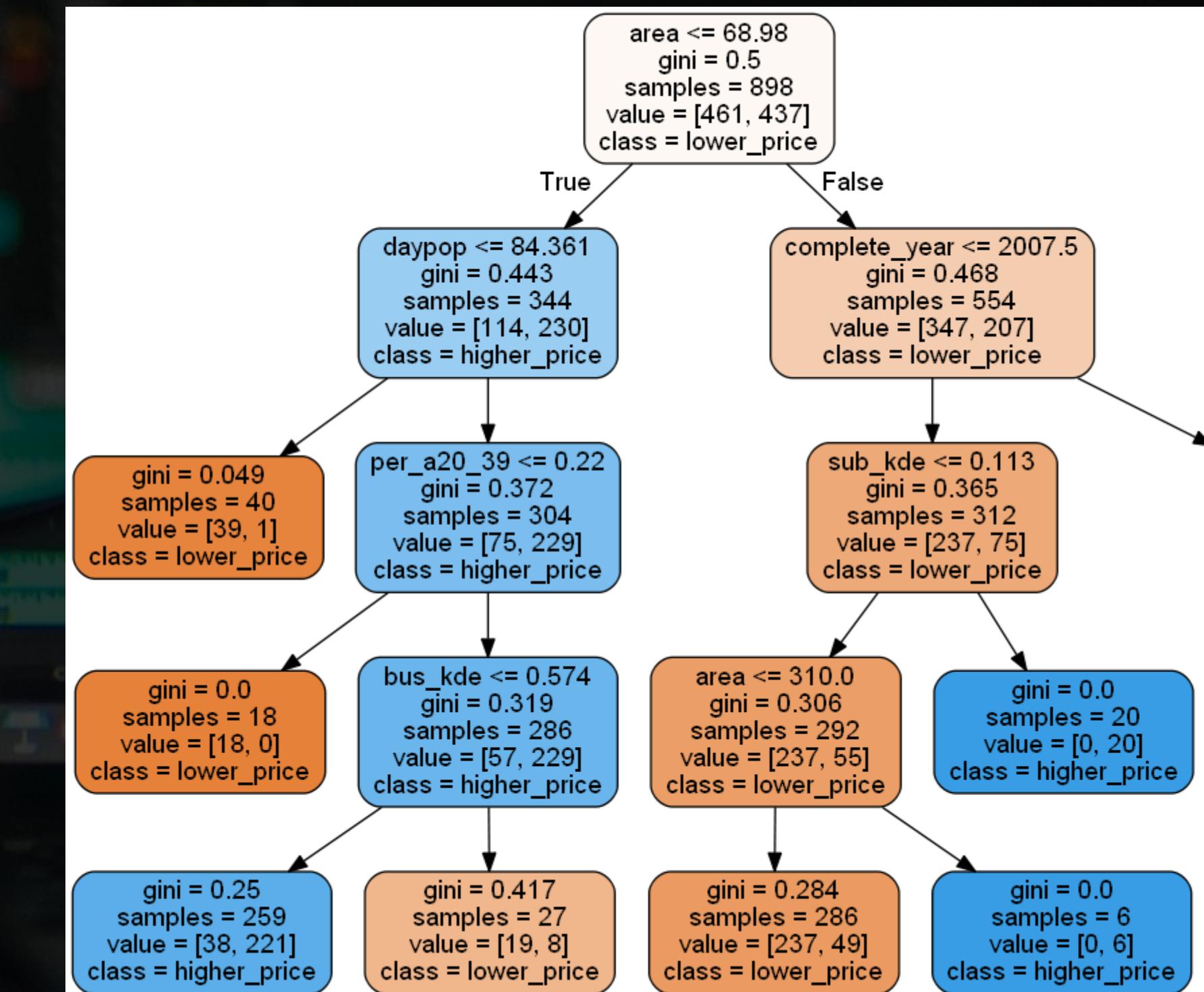
决策树模型

概念：基于树形结构来拟合自变量与应变量之间的关系

划分算法：选择最佳划分特征及特征中最佳划分点位置的算法

类别：

- ID3：信息增益判断
- C4.5：信息增益率判断
- CART：GINI系数判断



CART基尼系数法

算法流程：

- 1) 迭代计算每个特征的每个二分切点gini系数
- 2) 选取gini最小的特征及对应切分点为最佳分裂点
- 3) 进行一次样本划分
- 4) 对划分后的两部分样本重复以上迭代过程，逐步向下分裂
- 5) 所有样本被被分到叶节点中
- 6) 结束

CART分类
及回归树

CART分类与回归树

问题：树模型是否是越分叉越多，结构越复杂越好呢？

剪枝策略

剪枝目的：降低模型复杂度，防止过拟合

预剪枝：

在构建树的过程中，先计算当前的因此分裂是否能带来模型泛化能力的提升，如果不能，则不继续生长

后剪枝：

让算法生成一颗完整的决策树之后，然后从最底层向上计算，如果对分裂点剪枝，模型的泛化能力提升，则进行剪枝。

决策树剪枝策略

5.2

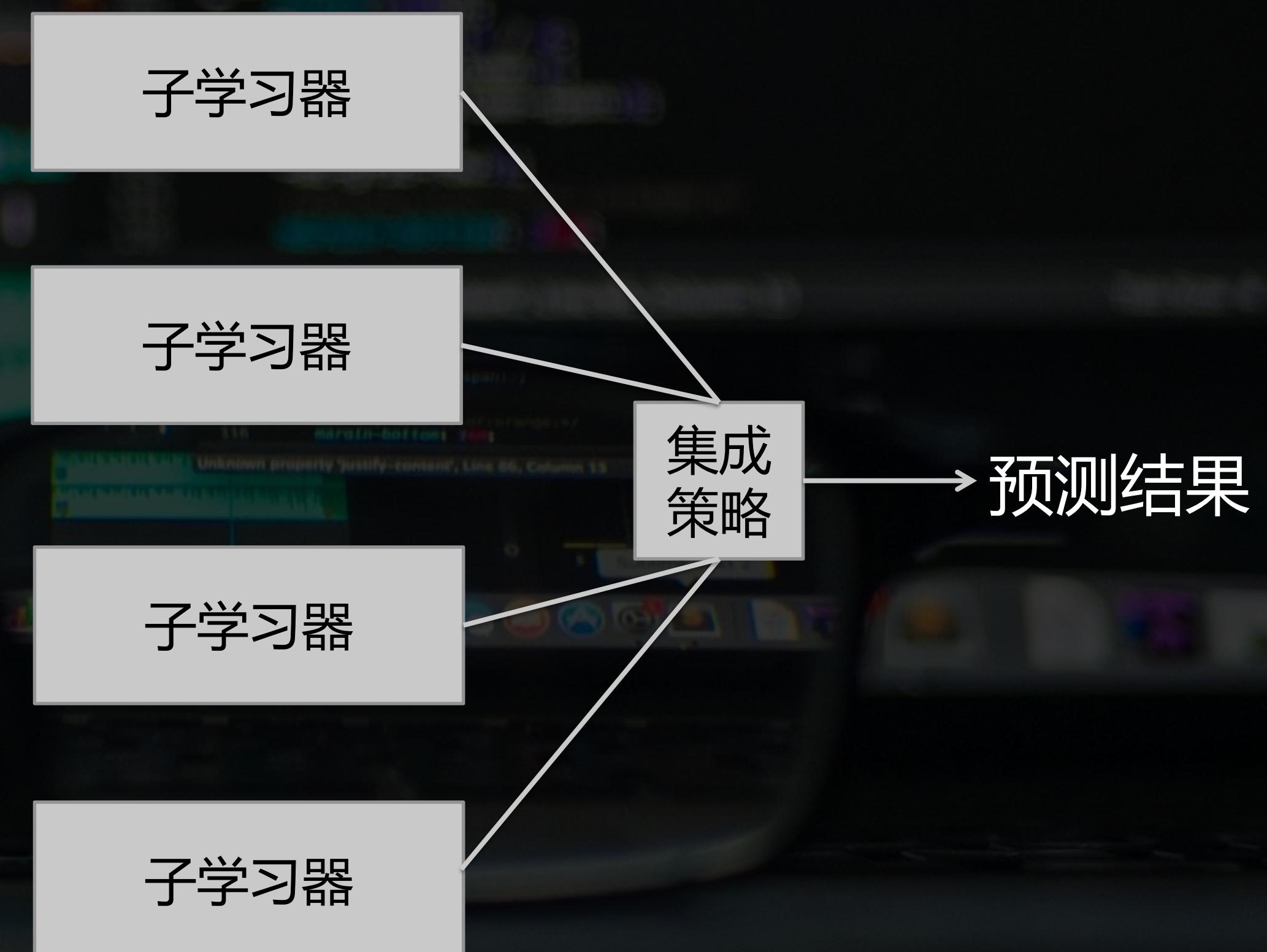
集成学习

集成学习

概念：通过构建并结合多个模型来共同完成学习任务

流程：

- 1) 构建多个子学习器
- 2) 使用某种集成策略将模型集成
- 3) 完成学习任务

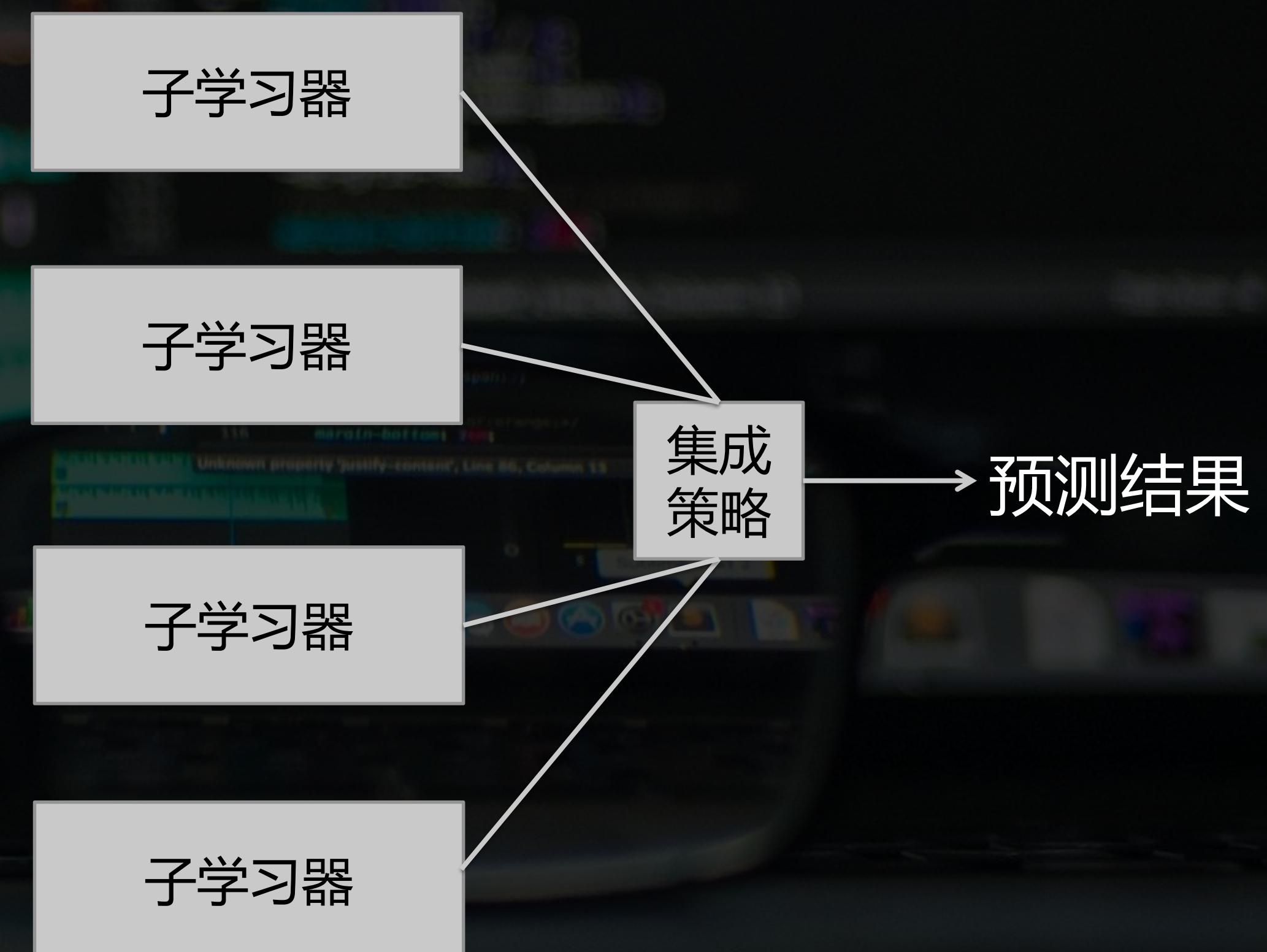


集成学习

目的：通过集成，提高多个子学习器的模型泛化能力

子学习器筛选原则：

- 1) 每个子学习器都要有一定的准确性
- 2) 子学习器之间要保持相对独立性和多样性



集成学习策略

Bagging：并行式集成学习

基本原理：

同时训练多个子学习机，分别对 y 进行预测，最后所有子学习机以投票的形式（分类）或者均值的形式（回归），返回集成的预测结果

子学习器构建策略：

对样本按一定比例有放回的抽样，抽出 m 个样本子集，然后构建 m 个子学习器分别在 m 个样本子集上进行训练

集成策略：

投票法或均值法

集成学习策
略

集成学习策略

Bagging：并行式集成学习

代表模型：随机森林

- 子学习器：决策树模型
- 子学习器构建方法：按一定比例同时对样本和特征进行有放回抽样，抽样出 m 个特征和样本都存在差异的样本子集,再在这 m 个子集上训练 m 个决策树模型。
- 集成方法：分类问题采用投票法返回预测结果；回归采用均值法返回预测结果

集成学习策
略

集成学习策略

Boosting：提升式集成学习

基本原理：

先训练一个子学习机，再计算子学习机的误差或残差，并以此作为下一个学习机的输入，之后不断迭代重复整个过程，使得模型损失函数不断减小。

Gradient Boosting梯度提升算法：

根据当前模型损失函数的负梯度信息来训练新加入的子学习器，然后将所有训练好的子学习器以累加的形式混合到最终模型中

集成学习策
略

集成学习策略

Boosting：提升式集成学习

集大成者：XGBOOST

- 基本思想：gradient boosting梯度提升
- 防止过拟合：加入L1和L2正则系数
- 子学习器：决策树模型（cart）、线性回归、线性分类器
- 集成策略：在传统梯度提升模型的基础上，融入随机森林模型对子学习器训练样本和特征进行随机取样的策略
- 优化方法：同时使用损失函数一节、二阶导数信息，加快优化速度
- 工程优化：提高训练效率，支持并行计算

集成学习策
略