# 算法概念与公式

## 随机森林回归器 (RandomForestRegressor)

随机森林回归器 (RandomForestRegressor) 是一种集成学习方法，通过组合多个决策树（通常是回归树）来提高模型的预测性能和稳定性。每棵决策树都是在一部分训练数据上独立构建的，这部分数据是从训练集中通过有放回抽样（即 Bootstrap 采样）得到的。最终的预测结果是所有决策树预测值的平均值。

随机森林具有良好的抗过拟合能力，尤其适合处理高维数据集和复杂的非线性关系。其优势在于通过集成多个弱学习器（即单个决策树）的预测，减少了单个模型可能带来的过拟合风险，并能够处理缺失数据和多样性很大的特征数据。

### 计算公式:

随机森林的基本思想是组合多个回归树的预测结果。对于回归问题，随机森林的预测值是所有决策树预测值的平均值。

假设有 M 棵决策树构成的随机森林，对于输入样本 x，第 m 棵树的预测值为 $h_m(x)$。那么，随机森林的最终预测值 $\hat{y}$ 计算公式如下：

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} h_m(\mathbf{x})$$

M: 决策树的数量。M 代表了森林中的决策树的总数，也就是所谓的"树的数量"（n_estimators 参数）。

$h_m(x)$: 第 m 棵决策树对输入样本 x 的预测值。每棵树都是独立训练的，它在子数据集（由 Bootstrap 方法抽取的）上构建，并根据树结构进行预测。

ŷ: 随机森林对输入样本 x 的最终预测值。这个值是所有 M 棵树的预测值的平均值，即各个树的预测值的简单算术平均。

## 随机梯度下降回归器(SGDRegressor)

随机梯度下降回归器(SGDRegressor)是一种基于梯度下降的线性回归模型。它通过逐步更新模型参数以最小化损失函数来学习模型参数。由于每次更新只使用一个样本或一小批样本，因此它在处理大规模数据集时非常高效。

### 计算公式:

在线性回归中，目标是找到一组参数，使得输入特征的线性组合能够尽可能准确地预测目标值。

$$\hat{y} = \mathbf{w} \cdot \mathbf{x} + b$$

其中，损失函数通常使用均方误差（Mean Squared Error, MSE）：

$$L = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

参数更新公式为：

$$\mathbf{w} := \mathbf{w} - \eta \frac{\partial L}{\partial \mathbf{w}}$$
$$b := b - \eta \frac{\partial L}{\partial b}$$

w：模型的权重向量，对应于每个输入特征的权重。

b：模型的截距，即偏置项。

X：输入特征向量。

ŷ：模型的预测值。

y：实际的目标值。

m：样本数量。

L：损失函数，用于度量预测值与实际值之间的差异。

η：学习率，控制每次参数更新的步长。

$\frac{\partial L}{\partial w}$：损失函数关于权重向量的梯度。

$\frac{\partial L}{\partial b}$：损失函数关于截距的梯度。

## 最小绝对收缩和选择算子(Lasso)

Lasso（Least Absolute Shrinkage and Selection Operator）是一种线性回归方法，通过在损失函数中加入 L1 正则化项来实现特征选择和模型的稀疏性。它倾向于产生较少特征具有非零系数的模型，从而能够有效地选择特征。

### 计算公式:

$$L = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2 + \alpha\sum_{j=1}^{n}|w_j|$$

w：模型的权重向量。

b：模型的截距。

α：正则化强度，控制正则化项的影响程度。

m：样本数量。

n：特征数量。

$\hat{y}$：模型的预测值。

y：实际的目标值。

## 决策树回归器(DecisionTreeRegressor)

决策树回归器（DecisionTreeRegressor）是一种通过构建决策树进行回归预测的方法。决策树通过递归地分割数据集，形成一棵树，树的每个叶节点代表一个预测结果。

### 计算公式:

在决策树回归中，每个叶节点的预测值为该叶节点内所有样本目标值的平均值：

$$\hat{y} = \frac{1}{|N|} \sum_{i \in N} y_i$$

N：叶节点中的样本集合。

$\hat{y}$：叶节点的预测值，即叶节点中所有样本目标值的平均值。

$y_i$：第 i 个样本的目标值。

## 弹性网络回归(ElasticNet)

ElasticNet 是一种结合了 L1 和 L2 正则化的线性回归方法，能够同时进行特征选择和参数收缩。它在 Lasso 和 Ridge 回归之间实现了平衡。

### 计算公式:

$$L = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2 + \alpha \left( \frac{1-\rho}{2} \sum_{j=1}^{n} w_j^2 + \rho \sum_{j=1}^{n} |w_j| \right)$$

w：模型的权重向量。

b：模型的截距。

α：正则化强度。

ρ：权衡 L1 和 L2 正则化的比例。

m：样本数量。

n：特征数量。

$\hat{y}$：模型的预测值。

y：实际的目标值。

## 支持向量回归（SVR）

支持向量回归（Support Vector Regression, SVR）是一种基于支持向量机（SVM）的回归方法。它通过引入不敏感损失函数，在一定的误差范围内进行拟合。

### 计算公式:

损失函数通常为：

$$L = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\max(0, |y_i - \hat{y}_i| - \epsilon)$$

w：模型的权重向量。

b：模型的截距。

C：正则化参数，控制训练样本中每个样本的权重。

ϵ：不敏感损失阈值，控制模型的容忍误差。

m：样本数量。

$\hat{y}$：模型的预测值。

y：实际的目标值。

## 梯度提升回归器(GradientBoostingRegressor)

梯度提升回归器（Gradient Boosting Regressor）是一种集成学习方法，通过逐步构建决策树来最小化损失函数。每一步都试图纠正前一步的误差。

### 计算公式:

梯度提升的更新公式为：

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \nu f_t(x_i)$$

t：迭代次数。

ν：学习率，控制每棵树对最终模型的贡献。

$f_t$：第 t 次迭代的回归树。

$\hat{y}_i^{(t)}$：第 t 次迭代后第 i 个样本的预测值。

## 岭回归(Ridge Regression)

岭回归（Ridge Regression）是一种线性回归方法，通过在损失函数中加入 L2 正则化项来防止过拟合。它通过对大权重施加惩罚来控制模型的复杂度。

### 计算公式:

$$L = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2 + \alpha \sum_{j=1}^{n} w_j^2$$

w：模型的权重向量。

b：模型的截距。

α：正则化强度。

m：样本数量。

n：特征数量。

$\hat{y}$：模型的预测值。

y：实际的目标值。

## AdaBoostRegressor

AdaBoostRegressor 是一种通过组合多个弱回归器来提高预测性能的集成学习方法。它采用自适应的方式，通过调整样本的权重来逐步优化模型性能。每个弱回归器的权重由其在前一轮中的预测误差决定，误差较大的样本在后续的回归器中会被赋予更高的权重，从而使模型更加关注难以预测的样本。

### 计算公式:

AdaBoost 的基本思想是组合多个弱回归器的预测结果。假设有 TTT 个弱回归器，每个回归器的预测值为 $h_t(x)$，其对应的权重为 $\alpha_t$。那么，AdaBoost 回归器的最终预测值 $\hat{y}$ 的计算公式如下：

$$\hat{y} = \sum_{t=1}^{T} \alpha_t h_t(x)$$

T: 弱回归器的数量。代表了模型中回归器的总数，也就是所谓的 "弱学习器的数量"（n_estimators 参数）。

$h_t(x)$: 第 t 个弱回归器对输入样本 x 的预测值。每个回归器都是独立训练的，并根据样本权重进行预测。

$\alpha_t$: 第 t 个弱回归器的权重。它反映了该回归器在组合预测中的重要性，通常由该回归器的误差率决定。

$\hat{y}$: AdaBoost 回归器对输入样本 x 的最终预测值。这个值是所有 T 个弱回归器的加权预测值的总和。

## 最小角回归(Lars)

最小角回归 Lars（Least Angle Regression）是一种逐步回归算法，适用于高维数据集的特征选择。它通过逐步引入特征来构建模型，确保每一步都选择与当前残差最相关的特征。Lars 的结果与 Lasso 相似，但计算效率更高。

### 计算公式:

Lars 算法的核心思想是逐步选择特征，并在每一步调整回归系数。具体公式为：

$$\mathbf{w} := \mathbf{w} + \gamma \mathbf{u}$$

w: 模型的权重向量。表示当前模型的参数，随着特征的引入逐步调整。

γ: 步长。表示每次调整的幅度，决定了模型参数的更新幅度。

u: 方向向量。表示当前选择的特征方向，使得模型沿着该方向调整参数。

## 极端梯度增强(XGBoost)

极端梯度增强 XGBoost（Extreme Gradient Boosting）是一种高效的梯度提升算法，具有强大的处理能力和灵活性。它在传统梯度提升的基础上进行了多项优化，如加权量化直方图、并行处理和缓存优化等，从而显著提高了模型的训练速度和预测性能。

### 计算公式:

XGBoost 的预测值是通过多个回归树的加权和来计算的。假设有 T 个回归树，每个树的预测值为 $f_t(x)$，其对应的权重为 $\omega_t$。那么，XGBoost 的最终预测值 $\hat{y}$ 的计算公式如下：

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \nu f_t(x_i)$$

t: 迭代次数。表示当前训练的回归树的序号。

v: 学习率。控制每棵树对最终模型的贡献，防止过拟合。

$f_t$: 第 t 次迭代的回归树。表示当前训练的树对输入样本 x 的预测值。

$\hat{y}_i^{(t)}$: 第 t 次迭代后第 iii 个样本的预测值。表示经过 t 棵树后模型对样本 $x_i$ 的预测结果。

## K 近邻回归(KNeighborsRegressor)

KNeighborsRegressor（K 近邻回归）是一种基于距离度量的回归方法。它通过计算输入样本与训练样本之间的距离，选择最近的 kkk 个邻居进行预测。这些邻居的目标值的平均值作为最终的预测结果。

### 计算公式:

KNeighbors 回归的预测值是通过最近 k 个邻居的目标值的平均值来计算的。假设最近邻居的目标值为 $y_i$，那么 KNeighbors 回归的最终预测值 $\hat{y}$ 的计算公式如下：

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i$$

k: 最近邻样本的数量。表示参与预测的邻居样本数量。

$y_i$: 第 i 个最近邻样本的目标值。表示邻居样本的实际目标值。

$\hat{y}$: KNeighbors 回归对输入样本的最终预测值。这个值是所有 k 个最近邻样本目标值的平均值。

## 线性回归(LinearRegression)

线性回归（LinearRegression）是一种最简单的回归方法，通过最小化残差平方和来拟合直线。它假设输入特征与目标值之间存在线性关系，通过计算输入特征的加权和来预测目标值。

### 计算公式:

线性回归的预测值是输入特征的加权和加上截距。假设输入特征为 x，权重向量为 w，截距为 b，那么线性回归的最终预测值 $\hat{y}$ 的计算公式如下：

$$\hat{y} = \mathbf{w} \cdot \mathbf{x} + b$$

w: 模型的权重向量。表示每个输入特征的权重。

b: 模型的截距。表示模型的偏置项。

x: 输入特征向量。表示待预测样本的特征。

$\hat{y}$: 线性回归对输入样本的最终预测值。这个值是输入特征的加权和加上截距。

## 贝叶斯岭回归 BayesianRidge

贝叶斯岭回归（BayesianRidge）是一种将贝叶斯方法应用于岭回归的模型，通过引入先验分布来估计模型参数。它通过最大化后验分布来优化模型，使得模型参数不仅考虑数据的拟合效果，还考虑先验知识。

### 计算公式:

贝叶斯岭回归的核心思想是通过最大化后验分布来估计模型参数。假设先验分布为 $p(w,\alpha,\lambda)$，后验分布为 $p(w,\alpha,\lambda|X,y)$，

那么贝叶斯岭回归的目标是：$w,\alpha,\lambda \sim p(w,\alpha,\lambda|X,y)$

w: 模型的权重向量。表示每个输入特征的权重。

α: 噪声精度的先验。表示目标值的噪声程度。

λ: 权重精度的先验。表示权重的正则化强度。

X: 输入特征矩阵。表示训练数据的特征。

y: 目标值向量。表示训练数据的目标值。

## Huber 回归(Huber Regressor)

Huber 回归（Huber Regressor）是一种对异常值不敏感的回归方法。它结合了均方误差（MSE）和绝对误差（MAE）的优点，通过引入 Huber 损失函数来实

现鲁棒回归。当误差较小时，Huber 损失函数表现为平方误差，当误差较大时，表现为线性误差，这使得 Huber 回归在处理数据中的异常值时具有较好的鲁棒性。

### 计算公式:

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2, & |y - f(x)| \le \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, & |y - f(x)| > \delta \end{cases}$$

这个公式是 Huber 损失函数的公式，用于在回归问题中计算损失。Huber 损失函数结合了均方误差（MSE）和绝对误差（MAE）的优点，在处理异常值时具有鲁棒性。

L_δ(y, f(x)) :表示损失函数，其中δ是一个超参数。

y :是真实值。

f(x) :是模型的预测值。

|y − f(x)| :表示预测值与真实值之间的绝对误差。

δ :是一个阈值参数，用于决定使用哪种损失计算方式。

当 |y − f(x)| ≤ δ 时，使用平方误差: (1/2)(y − f(x))^2

当 |y − f(x)| > δ 时，使用一个修改后的线性损失: δ|y − f(x)| − (1/2)δ^2

## 直方图梯度提升回归器(HistGradientBoostingRegressor)

直方图梯度提升回归器(HistGradientBoostingRegressor)是一种通过直方图近似来加速训练过程的梯度提升算法。它将连续特征离散化为直方图，从而减少计算量并提高训练速度。该算法适用于处理大规模数据集，具有较高的效率和良好的预测性能。

### 计算公式:

直方图梯度提升的更新公式与普通的梯度提升类似，每一步的预测值为前一步预测值与当前回归树预测值的加权和：

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \nu f_t(x_i)$$

t: 迭代次数，表示当前训练的回归树的序号。

v: 学习率，控制每棵树对最终模型的贡献，防止过拟合。

$f_t$: 第 t 次迭代的回归树，表示当前训练的树对输入样本 x 的预测值。

$\hat{y}_i^{(t)}$: 第 t 次迭代后第 i 个样本的预测值，表示经过 t 棵树后模型对样本 $x_i$ 的预测结果。

## 极度随机树回归器(ExtraTreesRegressor)

极度随机树回归器（ExtraTreesRegressor）是一种集成学习方法，通过构建多棵完全随机的决策树来提高预测性能。与随机森林不同，极度随机树在节点划分时使用了完全随机的特征和阈值，从而增加了模型的多样性。

### 计算公式:

极度随机树的预测值是通过所有决策树的预测值的平均值来计算的。假设有 MMM 棵决策树，每棵树的预测值为 $h_m(x)$，那么极度随机树的最终预测值 $\hat{y}$ 的计算公式如下：

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} h_m(x)$$

M: 决策树的数量，表示森林中的决策树的总数。

$h_m(x)$: 第 m 棵决策树对输入样本 x 的预测值，每棵树都是独立训练的，并根据树结构进行预测。

$\hat{y}$: 极度随机树对输入样本 x 的最终预测值，这个值是所有 M 棵树的预测值的平均值。

## 多层感知器回归器(MLPRegressor)

多层感知器回归器（MLPRegressor）是一种基于神经网络的回归方法，具有一个或多个隐藏层。它通过多层神经元的连接和非线性激活函数来学习复杂的函数关系，从而进行回归预测。

### 计算公式:

MLP Regressor 的预测值是通过多层神经网络的计算得到的。假设输入特征为 x，权重矩阵为 $W_1$ 和 $W_2$，偏置向量为 $b_1$ 和 $b_2$，激活函数为 f，那么 MLP Regressor 的最终预测值 $\hat{y}$ 的计算公式如下：

$$\hat{y} = f(\mathbf{W}_2 f(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2)$$

W1 和 W2: 权重矩阵，表示每层神经元之间的连接权重。

b1 和 b2: 偏置向量，表示每层神经元的偏置项。

f: 激活函数，用于引入非线性，通常使用 ReLU、Sigmoid 等函数。

x: 输入特征向量，表示待预测样本的特征。

$\hat{y}$: MLP Regressor 对输入样本的最终预测值，这个值是通过多层神经网络的计算得到的。