1. SMOTE 算法

SMOTE 对随机过采样算法进行了改进，使用插值方法进行新样本的人工合成，对少数类别进行了扩充，其算法流程如下：

Step 1：设少数类中的样本个数为 T，对其中的每一个样本x，计算它到其它少数类样本的距离，并取其k 近邻，记为 $x_{i,j}, j \in \{1, \ldots, k\}$。

Step 2：从 k 近邻中随机选择 1 个样本$x_{i,j}$，再生成一个范围为 0-1 的随机数 ε，新生成的样本为：

$$x_{i,new} = x_i + \varepsilon\,(x_{i,j} - x_i)$$

Step 3：根据样本不平衡比例设置生成倍率 N，对于每一个少数类样本 x_i，都重复步骤 $2N$次，从而将少数类样本扩大 N倍。

2. Tf-idf 算法

$$\begin{pmatrix} \text{This is a pencil} \\ \text{This is a car} \\ \text{This is a banana} \end{pmatrix}$$

	$This$	is	a	$pencil$	car	$banana$
$document\ 1$	1	1	1	1	0	0
$document\ 2$	1	1	0	0	1	0
$document\ 3$	1	1	0	0	0	1

下面计算 TF 值和 IDF 值，对于“this”这个单词，TF 是词在各文档中的频率

$$\mathrm{TF}('\ \mathrm{this}'\ ,\mathrm{d1}) = 0.25$$

$$\mathrm{TF}('\ \mathrm{this}'\ ,\mathrm{d2}) = 0.25$$

$$\mathrm{TF}('\ \mathrm{this}'\ ,\mathrm{d3}) = 0.25$$

IDF 是包含词的文档概率，所以

$$\mathrm{IDF}\left('\ \mathrm{this}'\ ,\mathrm{D}\right) = \log(3/3) = 0$$

$$\mathrm{TF} - \mathrm{IDF}\left('\ \mathrm{this}'\ ,\mathrm{d1}\right) = 0.25 \times 0 = 0$$

3. 模型评估

首先是模型评估的第一部分，基于评价指标的评估。对于分类问题，很多评价指标由混淆矩阵引出，下**表 1** 展示了混淆矩阵的结构：

表 1：混淆矩阵结构

	预测正例	预测负例
真实正例	True Positive	False Positive
真实负例	False Negative	True Nagative

根据混淆矩阵定义准确率，精确率（查准率），召回率（查全率），F_β分数，ROC 曲线，AUC 为：

$$\text{准确率(Accuracy)} = \frac{TP+TN}{TP+FP+FN+TN}$$

$$\text{精确率(Precision)} = \frac{TP}{TP+FP}$$

$$\text{召回率(Recall)} = \frac{TP}{TP+FN}$$

$$F_\beta = (1+\beta^2)\frac{precision \times recall}{(\beta^2 \times precision)+recall}$$

$$= \frac{(1+\beta^2)\times TP}{(1+\beta^2)\times TP+\beta^2 \times FN+FP}$$

$$TPR(True\ Positivee\ Rate) = \frac{TP}{TP+FN}$$

$$FPR(False\ Positivee\ Rate) = \frac{FP}{FP+TN}$$

ROC 曲线以 FPR 作为横坐标，TPR 作为纵坐标绘制，AUC 定义为 ROC 曲线下的面积，AUC 值越大，分类器表现越好。AUC 对非均衡样本的评估很稳定，所以被广泛应用。Kappa 系数也是基于混淆矩阵衡量精度的指标，范围在-1 到 1 之间，但通常落在 0 到 1 之间，可分为五个级别：当取值在 0-0.2 时，判定为极低(slight)，0.2-0.4 一般(fair)，0.4-0.6 中等(moderate)，0.6-0.8 高度一致(substantial)，0.8-1 几乎完全一致(almost perfect)。其计算公式如下：

$$\text{Kappa} = \frac{p_0 - p_e}{1-p_e}$$

其中 p_0 表示准确度 Accuracy，p_e 定义如下：

$$p_e = \frac{r_1 \times p_1 + r_2 \times p_2 + \cdots + r_c \times p_c}{n \times n}$$

每一类真实样本个数为r_i，预测出的每一类样本个数为p_i。

对于回归的评价一般基于误差，下面是各指标的计算公式：

$$\text{MSE}(\text{均方误差}) = \frac{1}{N}\sum_{t=1}^{N}(real_t - predict_t)^2$$

$$\mathrm{RMSE}(\text{均方根误差}) = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(real_t - predict_t)^2}$$

$$\mathrm{MAE}(\text{平均绝对误差}) = \frac{1}{N}\sum_{i=1}^{N}|real_t - predict_t|$$

$$\mathrm{RMSLE}(\text{均方根对数误差}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1) - \log(r_i+1))^2}$$

其中 p_i 为预测量，r_i 为真实值，n 表示样本数。

$$\mathrm{R}^2(\text{可决系数}) = 1 - \frac{RSS}{TSS}$$

$$Adjusted\ R^2(\text{可决系数}) = 1 - \frac{RSS/(n-k-1)}{TSS/(n-1)}$$

其中

$$TSS = \sum {y_i}^2 = \sum (Y_i - \bar{Y})^2$$

$$ESS = \sum {\hat{y}_i}^2 = \sum (\hat{Y}_i - \bar{Y})^2$$

$$RSS = \sum {e_i}^2 = \sum (Y_i - \hat{Y}_i)^2$$

$\hat{y}_i$为回归预测值与真实值的平均之间的差距，e_i是回归预测值与真实值之间的差距，y_i是真实值与真实值的平均之间的差距，n 为样本个数，k 为变量个数。

RMSE 是很常见的衡量工具，相比于 MAE，RMSE 强调对大误差的惩罚，减轻对小误差的惩罚，而且 MSE 与 RMSE 可以求导，所以同样作为了很多算法的损失函数。RMSLE 则对欠拟合的惩罚大于过拟合，但相比于 RMSE，由于其的对数机制，使得评价指标不会受到大误差的影响。可决系数是很好的指标，其优势在于易解释性。调整后的 R 方对变量的个数做了惩罚，解决了 R 方随着变量增多而增加的问题。

对于聚类问题的评价分为有监督与无监督，有监督聚类可以用分类的指标衡量，对于没有给定标签的聚类结果有紧密性（各点到聚类中心的平均距离），间隔性（各聚类中心之间的距离），戴维森堡丁指数（综合评判类内与类间）3 个指标，下面是指标的计算公式。

$$\text{Compactness(CP)} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{|\Omega_i|}\sum_{x_i\in\Omega_i}||x_i - w_i||$$

CP 指标没有考虑类间效果，CP 越低，类内聚类距离越近。

$$\text{Separation} = \frac{2}{k^2-k}\sum_{i=1}^{k}\sum_{j=i+1}^{k}||w_i - w_j||_2$$

SP 指标没有考虑类内效果，仅计算各聚类中心两两之间的平均距离，SP 越高意味着聚类距离越远。

$$\text{Davis} - \text{Bouldin} = \frac{1}{k}\sum_{i=1}^{k}\max\left(\frac{\overline{C_\iota} + \overline{C_J}}{||w_i - w_j||_2}\right)$$

DB 指标因为使用欧式距离，所以对于环状类分布的聚类评测很差，DB 越小则类内距离越小，类间距离越大。

4. 广义线性模型

这一部分主要就广义线性模型做出说明，讨论最小二乘估计，加入 L1 正则化的 Lasso 算法，加入 L2 正则化的 Ridge 算法，弹性网络和 Logistic 回归。对于一个线性模型 $\mathrm{y}^{(i)} = \theta^T X^{(i)} + \varepsilon^{(i)}$，假定 $\varepsilon^{(i)}(1 \le \mathrm{i} \le \mathrm{m})$ 独立且服从均值为 0 方差为 σ^2 的分布，所以可得：

$$p\left(\varepsilon^{(i)}\right) = \frac{1}{\sqrt{2\pi\sigma}}exp\ (-\frac{(\varepsilon^{(i)})^2}{2\sigma^2})$$

因为$\varepsilon^{(i)} = \mathrm{y}^{(i)} - \theta^T X^{(i)}$，所以有

$$p\left(\mathrm{y}^{(i)}\right) = \frac{1}{\sqrt{2\pi\sigma}}exp\ (-\frac{(\mathrm{y}^{(i)} - \theta^T X^{(i)})^2}{2\sigma^2})$$

下面使用最大似然估计求解 θ

$$\begin{aligned}\mathrm{L}(\theta) &= \prod_{i=1}^{m} p(\mathrm{y}^{(i)}|x^{(i)};\theta)\\ &= \prod_{i=1}^{m}\frac{1}{\sqrt{2\pi\sigma}}exp\ (-\frac{(\mathrm{y}^{(i)} - \theta^T X^{(i)})^2}{2\sigma^2})\end{aligned}$$

$$l(\theta) = log\ (L(\theta))$$

$$l(\theta) = mlog\frac{1}{\sqrt{2\pi\sigma}} - \frac{1}{2\sigma^2}\sum_{i=1}^{m}(\mathrm{y}^{(i)} - \theta^T X^{(i)})^2$$

下面就是对 $l(\theta)$ 求最大，由于除去求和部分其余变量均为定值，所以问题转化为求解$J(\theta)$（误差函数）的最小值。

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(\mathrm{y}^{(i)} - \theta^T X^{(i)}\right)^2$$
$$= \frac{1}{2}(X\theta - y)^T(X\theta - y)$$

$$\nabla_\theta J(\theta) = \nabla_\theta\left(\frac{1}{2}(X\theta - y)^T(X\theta - y)\right)$$
$$= X^T X\theta - X^T y$$

由此可以得到最小二乘算法下的结果：

$$\theta = (X^T X)^{-1}X^T y$$

为了使得模型更具有普遍意义，令$\mathrm{h}_\theta = f(\theta^T X + \varepsilon)$，在最小二乘意义下，$f$ 为最简单的函数，等于自变量本身，但是对于后面提到的 Logistic 回归，f 就变为e^x。虽然模型本质并没有改变，但是此时的模型已经不再是线性的了，所以称之为广义线性回归。

对前面的损失函数加上 L1 正则，可得 Lasso 回归：

$$J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y) + \lambda\sum_{j=1}^{n}|\theta_j|$$

对前面的损失函数加上 L2 正则，可得 Ridge 回归：

$$J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y) + \lambda\sum_{j=1}^{n}(\theta_j)^2$$

对其同时加上 L1 与 L2 正则，得到弹性网络。由于此时不能直接解出结果，所以对于此问题的求解可以使用 min-batch 梯度下降算法，$y^{(i)} - h_\theta\left(x^{(i)}\right)$ 为损失函数的梯度。

$$\theta_{i+1} = \theta_i + \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}))x^{(i)}$$

对于 Logistic 回归，设：

$$\mathrm{P}(y^{(i)} = 1 | x^{(i)}; \theta) = h_\theta(x) = p$$

$$\mathrm{P}(y^{(i)} = 0 | x^{(i)}; \theta) = h_\theta(x) = 1 - p$$

$$\mathrm{P}(y^{(i)} | x^{(i)}; \theta) = \left(h_\theta(x^{(i)})\right)^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

$$\mathrm{L}(\theta) = \prod_{i=1}^{m} \mathrm{P}(y^{(i)} | x^{(i)}; \theta)$$

设 $p_i = \frac{1}{1+e^{-f_i}}$，可得

$$l(\theta) = \sum_{i=1}^{m} \ln \left[(\frac{1}{1+e^{-f_i}})^{y_i} (\frac{1}{1+e^{-f_i}})^{1-y_i} \right]$$

同理，可以使用梯度上升算法计算 θ 。

对于广义线性模型，需要注意特征之间是否存在共线性关系，以及是否有异方差性存在。

下面举例说明 Logistic 回归的应用，假设我们需要对一个银行客户群体进行信用卡违约预测，根据其的消费记录，存取款时间，信用卡还款记录等信息可以构建特征，然后使用这些特征来描述单个客户，但是这样得到的仅仅是不同特征加权的数值结果，虽然有比较意义，但是无法分析，所以 Logistic 回归告诉我们可以通过寻找合适位置的 Sigmoid 函数对数值结果做映射，使之成为可以理解的概率，这就是 Logistic 回归。事实上，银行确实广泛使用 Logistic 回归来处理违约概率的问题。

对于 Logistic 回归的优点：其算法较为简单，易于理解；计算成本不高；噪点影响不大，模型鲁棒性较好；以概率形式输出结果，可以做排序；对少数类别分类问题表现较好；对于其缺点：容易欠拟合。Logistic 回归在业界应用广泛，经常作为分类算法的首选尝试模型，但由于其分类准确度不够高，一般

使用神经网络和树模型（GBDT；LightGBM 等）替代，或者作为集成模型中的基学习器或 Meta 学习器。

5. 树模型

5.1 决策树

树模型是机器学习中最重要的模型之一，由于其出色的表现和可解释性被工业界广泛运用，其中决策树算法是最基础的树模型。为了更好的理解决策树，这一部分的内容先从一个例子说起。假定我们认为 OECD 原油库存量和 OECD 原油消费量是决定伦敦期货交易所 BRENT 原油未来价格走势的关键，但是我们并不知道库存量处于什么样的水平时其处于高位，消费量处于什么样的水平时其处于低位。如果无法对高低做出预判，就没有办法根据供需对未来价格的走势进行预测。这时决策树告诉我们，当 OECD 原油库存大于 a，且消费量小于 b 时，将价格未来走势判断为下降；当 OECD 原油库存小于 a，且消费量大于 b 时判断为上升，这样一套准则的正确率最高，我们就可以按照这样一种逻辑进行交易。这就是决策树最简单的运行机制，而算法本身解决的问题是寻找合适的 a 和 b，并判断应该先用 OECD 库存还是消费量作为判断条件，本质是使用贪心算法寻优，并用高维分段函数对问题进行描述。

决策树算法的本质不同于广义线性回归，不再假定分布之后使用最大似然方法对参数进行估计，而是使用基于信息熵（Entropy）的贪心算法。信息熵是衡量信息量的指标，下面举一个例子来描述其构建思路。当了解到一件事情发生的概率是 99.99%时，如交易日股票市场的集合竞价时间为 9:15-9:25，其所带来的信息量很小，因为其每天都在发生；但一件事情的概率只有 1%甚至更低时，如俄罗斯政府违约，其所包含的信息量是巨大的。所以如果需要定义一个值越大代表信息强度越大的量，其至少是与概率成反比的，假设定义为 $-\log(p)$ 。对于一个随机变量 X_i，假定其中每个事件发生的概率是 p_i，则其对应的信息强度为$-\log(p_i)$，对随机变量中每个事件的信息强度按照概率加权，可以得到信息熵的表达式：

$$H(X) = -\sum_{i=1}^{n} p_i log p_i$$

通过信息熵可以定义每个特征的信息强度，对于决策树如何根据信息熵进行分支，需要引入信息熵变化这一概念，也就是信息熵增益。这里我们希望将整个决策树构建的过程转化为让信息强度下降最快的过程，因为当信息强度很小时，其根节点对应的事件几乎是确定发生的。信息增益的定义如下：

$$Info_A(D) = \sum_{j=1}^{n} \frac{|D_j|}{|D|} info(D_j)$$

$$gain(A) = info(D) - Info_A(D)$$

这里用一个例子来说明信息增益的计算，表中数据非真实，仅供说明需要：

表 2：市场冲击成本历史观察表

交易单号	交易金额	市场流动性	市场活跃程度	冲击成本
001	<100 万	2.3	活跃	约 20bp
002	100-1000 万	2.7	稳定	约 25bp
003	>1000 万	3.2	稳定	约 25bp
004	100-1000 万	1.9	活跃	约 20bp
005	>1000 万	3.1	活跃	约 25bp
006	<100 万	2.4	不活跃	约 25bp

根据表中数据，总计 6 个交易单，冲击成本约为 20bp 的 3 个，约为 25bp 的 3 个，计算得到信息熵为：

$$Info(D) = -\frac{3}{6} log\left(\frac{3}{6}\right) - -\frac{3}{6} log\left(\frac{3}{6}\right) = log\ (2)$$

然后计算交易金额这一属性的信息量，小于 100 万的 2 个，其中冲击成本约为 20bp 的 1 个，约为 25bp 的 1 个；100-1000 万的 2 个，其中冲击成本约为 20bp 的 1 个，约为 25bp 的 1 个；大于 1000 万的 2 个，其中冲击成本约为 25bp 的 2 个，计算得到信息熵为：

$$Info_{交易金额}(D) = \frac{2}{6}\left(-\frac{1}{2}\log\left(\frac{1}{2}\right) - \frac{1}{2}\log\left(\frac{1}{2}\right)\right)$$
$$+\frac{2}{6}\left(-\frac{1}{2}\log\left(\frac{1}{2}\right) - \frac{1}{2}\log\left(\frac{1}{2}\right)\right) + \frac{2}{6}\left(-\frac{2}{2}\log\left(\frac{2}{2}\right)\right) = \frac{2}{3} log\ (2)$$

所以，信息增益为：

$$Gain(\text{交易金额}) = log(2) - \frac{2}{3}log(2) = \frac{1}{3}log(2)$$

决策树构建的准则就是选择信息增益最大的特征进行分支，以最快的速度到达根节点，但是当单个节点分支过多时，如在上述问题中加入交易单号这一特征，会发现交易单号会被作为第一个特征进行分支，因为这样只需要一次分支操作就可以使得信息熵降低为 0，但是这明显会导致过拟合问题，所以我们希望特征本身的信息熵不能过大，更改使用 Gain(feature)/Entropy(feature)作为分支标准，这样构建决策树的算法被称为 C4.5，前述方法被称为 ID3。还有一种基于 Gini 系数作为分支的方法，称为 CART 树，对 C4.5 作了进一步改进，相比较而言，CART 树是较优的做法，事实上 Sklearn 中所用的模型就是 CART。

对于决策树算法的优点：其理解较为简单且易于可视化；能够同时接受数值和类别型的变量；可以基于决策树构建表现更好的集成学习模型；对于决策树的缺点：其模型较不稳定，数据集的轻微改变可能会导致树结构的变化；容易欠拟合与过拟合；参数较多，调参复杂；训练时间较长；业界对决策树模型的运用非常广泛，但准确来说主要运用的是决策树的集成模型，如 GBDT 和 LightGBM。

6. 集成模型

在决策树算法中，比较常用的集成学习模型有随机森林，GBDT，XGBoost 与 LightGBM，目前最流行的是 LightGBM，其在运行速度和准确度上都较其它更好。对于集成学习方式，有三种常用方法，一是 Bagging 组合，二是 Boosting 组合，三是 Stacking 组合。对于这三种集成方法，MLSolver 框架中已经集成，方便将各种机器学习基学习器进行融合，下面主要详细介绍三种集成学习方法。

最简单的模型融合方式是 Bagging 方法，对于分类问题采用基学习器投票的方式，对于回归问题采用取平均或以某种方式作加权平均的方式。由于 Bagging 算法中所有基学习器可以并行，所以速度很快。

Boosting 方法是对 Bagging 算法的改进，其核心是在梯度方向上构造基学习器，逐渐减小损失，下面介绍其原理。给定输入和输出构成的若干训练样本，$(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$，目标是找到函数 F(x)，使得损失函数 L(y,F)的损失值最小，常见损失函数定义为：

$$\mathrm{L(y,F)} = \frac{1}{2}(y - F(x))^2$$

假定最优函数为$F^*(x)$，即：

$$F^*(x) = argmin_F E_{(x,y)}[L(y, F(x))]$$

将$F(x)$表示为一族函数的加权：

$$\mathrm{F(x)} = \sum_{i=1}^{M} \gamma_i f_i(x) + C$$

对于常函数$\mathrm{F}_0(\mathrm{x})$：

$$\mathrm{F}_0(\mathrm{x}) = \mathrm{argmin}_c \sum_{i=1}^{n} L(y_i, c)$$

以贪心思路拓展得到$\mathrm{F}_m(\mathrm{x})$：

$$\mathrm{F}_m(\mathrm{x}) = \mathrm{F}_{m-1}(\mathrm{x}) + \mathrm{argmin}_{f \in H} \sum_{i=1}^{n} L(y_i, \mathrm{F}_{m-1}(x_i) + f(x_i))$$

将样本带入基函数得到：

$$\mathrm{F}_m(\mathrm{x}) = \mathrm{F}_{m-1}(\mathrm{x}) + \gamma_m \sum_{i=1}^{n} \Delta_f L(y_i, \mathrm{F}_{m-1}(x_i))$$

使用线性搜索求最优步长γ_m：

$$\gamma_m = \mathrm{argmin}_\gamma \sum_{i=1}^{n} L(y_i, \mathrm{F}_{m-1}(x_i) - \gamma \Delta_f L(y_i, \mathrm{F}_{m-1}(x_i)))$$

Stacking 算法是使用基学习器训练出的结果作为输入，然后使用 Meta 学习器进行再训练，Stacking 集成模型应用非常广泛，下**图 7** 以回归为例展示了 Stacking 方法的集成过程。

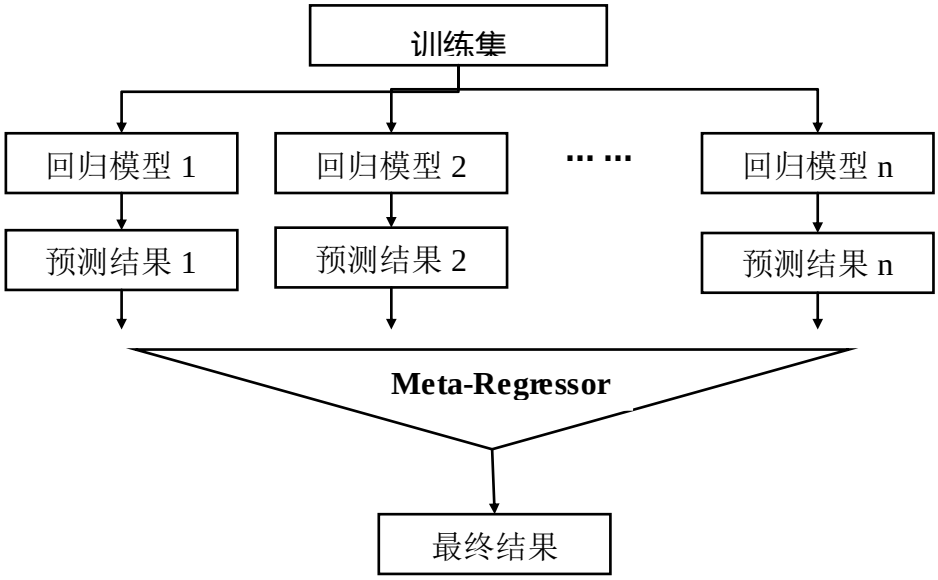

图 7：Stacking 集成学习算法流程

7. 支持向量机

支持向量机算法是机器学习中最重要的算法之一，在深度学习网络被发现之前，其一直在机器学习领域占据统治地位。近些年，由于神经网络的崛起和 SVM 的一些缺陷，业界已经很少使用支持向量机处理问题，但是其在小规模低维数据集上的出色表现使之仍有很强的使用价值，下面对 SVM 的原理进行讨论。

SVM 的本质是在高纬空间中寻找一个超平面使之可以将数据集分割，从而达到分类的目的。将超平面定义为：

$$\mathrm{y} = \mathrm{w}^T \Phi(\mathrm{x}) + \mathrm{b}$$

相应的决策函数为：

$$\mathrm{f}(\mathrm{x}) = \mathrm{sign}(\mathrm{w}^T \Phi(\mathrm{x}) + \mathrm{b})$$

其中 $\Phi(\mathrm{x})$ 是投影函数，方便将 x 映射到更高的维度上去。若问题本身线性可分，则 $\Phi(\mathrm{x}) = x$。当$\mathrm{y}(\mathrm{x}_i) > 0$，则$\mathrm{y}_i = 1$；若$\mathrm{y}(\mathrm{x}_i) < 0$，则$\mathrm{y}_i = -1$。由此可得：

$$\mathrm{y}_i y(\mathrm{x}_i) > 0$$

对上式进行等比例缩放w，使得两类点的函数值都满足$|\mathrm{y}| \geq 1$可得：

$$\frac{y_i y(x_i)}{||w||} = \frac{y_i(w^T \Phi(x) + b)}{||w||}$$

根据最大间隔分离超平面这一目标可得：

$$\arg_{w,b}\max\{\frac{1}{||w||}\min_i[y_i(w^T\Phi(x_i)+b)]$$

对目标函数进行简化，可得新目标函数：

$$\arg_{w,b}\max\frac{1}{||w||}$$

其约束为：$y_i(w^T\Phi(x)+b)\geq 1, i=1,2,3\ldots n$

下面只需采用拉格朗日算法对拉格朗日的对偶函数求解即可得到最大分割的超平面，本质是对下面的问题进行求解：

$$\min_a \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}a_i a_j y_i y_j\left(\Phi(x_i)\cdot\Phi(x_j)\right)-\sum_{i=1}^{n}a_i$$

$$\text{s.t.}\ \sum_{i=1}^{n}a_i y_i=0$$

$$a_i\geq 0, i=1,2,3\ldots n$$

计算可得a_i，并知：

$$w=\sum_{i=1}^{N}a_i y_i\Phi(x_i)$$

$$b=y_i-\sum_{i=1}^{N}a_i y_i(\Phi(x_i)\cdot\Phi(x_j))$$

对拉格朗日的对偶函数加上松弛因子，然后再求解，即可得到带松弛因子的 SVM，这主要为了避免过拟合的问题。

下面通过一个具体的例子来理解 SVM 算法。

假定对市场环境进行建模，有两个特征，一是收益率波动水平，有 4 个级别，级别越高其波动状态越剧烈；二是成交量变化波动水平，同样有 4 个级别，级别越高其波动状态越剧烈；下面需要根据这两个特征对市场环境进行分类，有如下 3 个训练样本点：

表 3：市场环境历史观察表

观察时点	收益率波动	成交量波动	市场环境
1	3	3	+
2	4	3	+
3	1	1	−

根据 SVM 的优化目标函数：

$$\min_a \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{n} a_i$$

$$= \frac{1}{2}(18{a_1}^2 + 25{a_2}^2 + 2{a_3}^2 + 42a_1a_2 - 12a_1a_3 - 14a_2a_3) - a_1 - a_2 - a_3$$

$$\text{s.t.}\, a_1 + a_2 + a_3 = 0, a_i \geq 0, i = 1,2,3$$

将 $a_1 + a_2 = a_3$ 带入目标函数，得到关于 a_1, a_2 的函数：

$$s(a_1, a_2) = 4{a_1}^2 + \frac{13}{2}{a_2}^2 + 10a_1a_2 - 2a_1 - 2a_2$$

对a_1, a_2求偏导并令其为 0，易知$s(a_1, a_2)$在点 $(1.5, -1)$ 处取极值。而该点不满足条件$a_2 \geq 0$，所以，最小值在边界上。

当$a_1 = 0$时，最小值$s(0, 2/13) = -2/13 = -0.154$，当$a_2 = 0$时，最小值$s(1/4, 0) = -1/4 = -0.25$。所以可得$s(1/4, 2/13)$ 时最小，此时 $a_3 = a_1 + a_2 = 1/4$。

根据w, b 的公式可得，$w_1 = w_2 = 0.5, b = -2$

因此，分离超平面为：

$$\frac{1}{2}x_1 + \frac{1}{2}x_2 - 2 = 0$$

分离决策函数为：

$$f(x) = \text{sign}\left(\frac{1}{2}x_1 + \frac{1}{2}x_2 - 2\right)$$

对于 SVM 算法的优点：SVM 有很好的泛化能力，因为其本身的优化目标在于结构最优而不是经验最优；SVM 在小样本数据上表现很好；对高维数据适应度很好；基于核函数构建，有很大的灵活性；SVM 的缺点有：SVM 只提供分类结果，并不提供可能性的估计；SVM 模型较复杂，不容易解释；参数较多不容易调参。

8. 朴素贝叶斯

朴素贝叶斯（Naïve Bayes, NB）是一种应用贝叶斯公式的有监督学习算法。其基于两个假设：

（1）特征独立性：一个特征出现的概率与其它特征条件独立，这一假设主要为了简化贝叶斯公式的计算。

$$P(x_i | y, x_1, \ldots\ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i | y)$$

（2）特征均衡性：每个特征同等重要，这一假设是由于贝叶斯公式本身并没有对特征进行加权处理。

根据贝叶斯公式：

$$P(y|x_1,\dots\dots,x_n)=\frac{P(y)P(x_1,\dots\dots,x_n|y)}{P(x_1,\dots\dots,x_n)}$$

依据所有给定特征值，可以判读 y 为预测值的概率有多大，从而完成分类问题。根据假设一，可以化简贝叶斯公式为：

$$P(y|x_1,\dots\dots,x_n)=\frac{P(y)\prod_{i=1}^{n}P(x_i|y)}{P(x_1,\dots\dots,x_n)}$$

在给定样本的前提下，$P(x_1,\dots\dots,x_n)$为常数，所以

$$P(y|x_1,\dots\dots,x_n)\propto P(y)\prod_{i=1}^{n}P(x_i|y)$$

从而得到对每个样本的分类依据：

$$\hat{y}=argmax_y P(y)\prod_{i=1}^{n}P(x_i|y)$$

下面来举例说明贝叶斯分类器的应用，假设有一个样本为 1000 个交易日，每个交易日对应着一个信息向量$X=(x_1,\dots\dots,x_n)$包含用于判断当天的市场情绪的信息。建立贝叶斯分类器的目的是给定第 1001 个交易日，确定其市场情绪的强度，是震荡 C1 还是平稳 C2。根据贝叶斯公式：

$$P(C1|X)=\frac{P(X|C1)\times P(C1)}{P(X)}$$

$$P(C2|X)=\frac{P(X|C2)\times P(C2)}{P(X)}$$

其中 $P(x_1|C1)$表示在市场平稳的前提下，x_1这一特征出现的概率，使用先验分布估计；$P(x_1)$ 表示在所有样本中，x_1这一特征出现的概率；$P(C1)$ 表示在所有样本中市场震荡出现的概率。

朴素贝叶斯中对于$P(y)$的估计使用点估计的方法，对于条件概率分布$P(x|y)$中参数的估计方法采用最大似然估计。最大似然估计（Maximum

Likelihood Estimation, MLE）在已知数据服从某一分布的前提下，找出一组参数来确定分布，使得模型产生出观测数据的概率最大。如假设已知数据服从二项分布，则需要使得所有数据产生于这个分布的概率最大，由于假设样本独立同分布，所以可以将需要最大化的概率简化为每个样本服从这一分布的概率相乘，由此构建出最大似然函数，将问题转化为求解最大似然函数在值最大的情况下的参数求解问题。

根据上述所示，使用贝叶斯模型之前需要从两个维度检验：一是特征之间是否相互独立，二是特征的条件概率分布，用于选择合适的模型。

对于贝叶斯算法的优点，一是对结果的使用概率进行评价，不完全接受或否决；二是对缺失数据不敏感；三是模型简单，参数很少，易于理解；四是训练速度很快；五是适用于增量训练。对于算法的缺点，一是基于的前提假设使得其不能很好适用于很多数据集；二是不同的先验分布对数据的要求不一样。

9. 聚类分析

聚类分析是很常用的算法，如银行对客群的聚类分析，对市场环境的聚类判断等。常见的聚类算法有 KNN 算法（有监督学习）与 K-means 算法（无监督学习），相对于其它机器学习算法，其逻辑非常简单。K-Means 聚类算法主要分为 4 个步骤：

Step 1：在全样本中初始化聚类中心 K 个。

Step 2：计算每个样本点到聚类中心的距离，对每个样本，依据距离聚类中心最近的准则标记到离该点最近的类中去。

$$C^{(i)} = argmin_k ||x^{(i)} - \mu_{k^{(i)}}||^2$$

其中$C^{(i)}$表示类别标签，$\mu_{k^{(i)}}$表示样本点 $x^{(i)}$ 的聚类中心。

Step 3：计算每个聚类中心所有点坐标的平均值，将聚类中心移动到该点坐标上去。

$$\mu_k = \frac{1}{n}[x^{(k_1)} + x^{(k_2)} + \cdots + x^{(k_n)}] \in R^n$$

Step 4：反复执行第 2 和 3 布，直至损失函数收敛，或迭代次数到达设定界限，其损失函数为：

$$J\left(C^{(i)},\dots,C^{(i)},\mu_k,\dots,\mu_k\right)=\frac{1}{m}\sum_{i=1}^{m}||x^{(i)}-\mu_{k^{(i)}}||^2$$

KNN 算法的思想与 K-means 类似，算法步骤如下：

Step 1：计算待分类数据与所有样本数据之间的距离；

Step 2：按照距离的递增关系排序；

Step 3：选择距离最小的 K 个点；

Step 4：计算前 K 个点所在类别出现的概率；

Step 5：返回前 K 个点中出现频率最高的类别作为测试数据的预测分类；

对于 K-means 算法，有两个问题需要解决，一是初始的聚类中心如何选择；二是如何确定分类数。对于第一个问题，K-means++对算法做了改进，下面是 K-means++的确定初始点的过程：

Step1：从数据集中选取一个样本作为初始聚类中心

Step2：首先计算每个样本与当前已有聚类中心之间的最短距离，即与最近的一个聚类中心的距离，用 D(x)表示；接着计算每个样本被选为下一个聚类中心的概率：

$$\frac{D(x)^2}{\sum_{x\in X}D(x)^2}$$

最后按照轮盘赌的方法选出下一个聚类中心。

Step3：重复第二步直到选出 K 个聚类中心

对于有特定需求的实际场景，分类数目的确定根据情况而定，如对市场环境的聚类，根据需求可以分为震荡与平缓，则 K 的取值为 2。但是有些问题并不知道应该聚类的个数，肘方法（Elbow Method）可以解决这一问题，首先绘制损失函数 J 关于分类数目 K 的函数图，J 值会随着 K 的增加而逐渐减小，最后趋向于平缓，对曲线做二次拟合，选择二阶导数为 0 的点，对其就近取整数，即可得到应该分类的数目。

K-means 与 KNN 由于算法简单，所以容易理解和解释，但也有很多缺点：一是此算法只有在类内平均值有定义的情况下才可以使用，所以不能适应很多场景，如：特征中包含无序类别型变量；二是当样本数目非常多时，由于其需要计算每个样本点到聚类中心点的距离，所以计算开销非常大；三是对异常点很敏感；四是对样本值缩放很敏感；五是其容易陷入局部最优，即使数据规整；