朴素贝叶斯（native Bayes）

$$P(Y|X) = \frac{P(X,Y)}{P(X)} = \frac{P(Y)P(X|Y)}{\sum_Y P(Y)P(X|Y)}$$

将输入 x 分到后延概率最大的类 y

$$y = argmax_{c_k} P(Y = c_k) \prod_{j=1}^{n} P(X_j = x^{(j)}|Y = c_k)$$

注意点：

1 朴素贝叶斯对条件概率分布做了条件独立性的假设。

$$P(X = x|Y = c_k) = P\left(X^{(1)} = x^{(1)},\cdots,X^{(n)} = x^{(n)}\middle|Y = c_k\right)$$

$$= \prod_{j=1}^{n} P(X^{(j)} = x^{(j)}|Y = c_k)$$

相关知识点：

1 先验概率与后验概率

事情还没有发生,要求这件事情发生的可能性的大小,是先验概率.

事情已经发生,要求这件事情发生的原因是由某个因素引起的可能性的大小,是后验概率.

先验概率是指根据以往经验和分析得到的概率，如全概率公式，它往往作为“由因求果”问题中的“因”出现。后验概率是指在得到“结果”的信息后重新修正的概率，如贝叶斯公式中的，是“执果寻因”问题中的“因”。先验概率与后验概率有不可分割的联系，后验概率的计算要以先验概率为基础。

2 利用极大似然估计计算的时候可能出现概率值为 0 的情况，会影响到后验概率的结果，使分类产生偏差。因此采取增加λ的方式，称为拉普拉斯平滑。

优点：

1 因为独立性的假设，模型包含的条件概率的数量大为减少，因此高效，且易于实现。

2 对小规模的数据表现很好，适合多分类任务，适合增量式训练。

缺点：

1 因为独立性的假设，牺牲了一定的分类准确率，分类的性能不一定高。

2 对输入数据的表达形式很敏感。

KNN(k-nearest neighbor)

注意点：

1 K 值的减小意味着整体模型变得复杂，容易发生过拟合。K 值变大意味着整体模型变得简单。 K 值通常选取一个比较小的数据，采用交叉验证法来选取最优的 K 值。常用的分类决策规则是多数表决，对应于经验风险最小化。
2 KNN 的基本做法是：对给定的训练实例点和输入实例点，首先确定输入实例点的 k 个最近邻训练实例点，然后利用这 k 个训练实例点的类的多数来预测输入实例点的类。
3 KNN 模型对应于基于训练数据集对特征空间的一个划分。KNN 中，当训练集，距离度量，k 值和分类决策规则确定后，其结果唯一确定。
4 KNN 的实现需要考虑如何快速搜索 k 个最近邻点。kd 树是一种便于对 k 维空间中的数据进行快速检索的数据结构。kd 树是二叉树，表示对 k 维空间的一个划分，其每个节点对应于 k 维空间划分的一个超矩形区域。利用 kd 树可以省去对大部分数据点的搜索，从而减少搜索的计算量。kd 树的平均计算复杂度是 O(logN).

相关知识点：

优点：

1 思想简单，理论成熟，既可以用来做分类也可以用来做回归。
2 可用于非线性分类。
3 训练时间复杂度为 O(n).
4 准确度高，对数据没有假设，对 outlier 不敏感。

缺点：

1 计算量大
2 样本不平衡（即有些类别的样本数量很多，而其他样本的数量很少）
3 需要大量的内存

决策树（decision tree）

熵（entropy）的定义：表示随机变量不确定性的度量。熵越大，随机变量的不确定性越大。

$$P(X = x_i) = p_i$$

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

$$H(Y|X) = \sum_{i=1}^{n} p_i H(Y|X = x_i)$$

如果对数以 2 为底，那么熵的单位是比特（bit），如果以为底，那么单位是纳特（nat）。

信息增益（information gain）表示得知特征 A 的信息而使得训练数据集 D 的信息的不确定性减少的程度。信息增益打的特征具有更强的分类能力。

$$g(D, A) = H(D) - H(D|A)$$

注意点：

1 因为从可能的决策树中直接选取最有决策树是 NP 完全问题。现实中采用启发式方法学习次优的决策树。决策树的学习算法包含 3 部分，特征性选择，树的生成和树的剪枝、常用的算法有 ID3，C4.5 和 CART。

2 特征选择的目的在于选取对训练数据能够分类的特征。特征选择的关键是其准则。常用的准则如下：

（1） 样本集合 D 对特征 A 的信息增益（ID3）

$$g(D, A) = H(D) - H(D|A)$$

$$H(D) = -\sum_{k=1}^{k} \frac{|C_k|}{|D|} log_2 \frac{|C_k|}{|D|}$$

$$H(D|A) = \sum_{n=1}^{n} \frac{|D_i|}{|D|} H(D_i)$$

其中，H(D)是数据集 D 的熵，H(D_i)是数据集D_i的熵，H(D|A)是数据集 D 对特征 A 的条件熵。D_i是 D 种特征 A 取第 i 个值的样本子集，C_k是 D 中属于第 k 类的样本子集。|D|表示其样本容量，即样本个数。n 是特征 A 取值的个数，K 是类的个数。

（2） 样本集合 D 对特征 A 的信息增益比（C4.5）

$$g_R(D, A) = \frac{g(D, A)}{H(D)}$$

其中，g(D,A)是信息增益，H(D)是数据集 D 的熵。

（3） 样本集合 D 的基尼指数（CART） CART 的决策树是二叉树

$$Gini(D) = 1 - \sum_{k=1}^{k} (\frac{|C_k|}{|D|})^2$$

特征 A 条件下集合 D 的基尼指数：

$$Gini(D, A) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

3 决策树的生成。通常使用信息增益最大，信息增益比最大或者基尼指数最小作为特征选择的准则。决策树的生成往往通过计算信息增益或其他指标，从根节点

开始，递归地产生决策树。这相当于用信息增益或其他准则不断的选取局部最优的特征，或将训练集分割为能够基本正确分类的子集。
4 决策树的剪枝。由于生成的决策树存在过拟合的问题，需要对它进行剪枝，以简化学到的决策树。决策树的剪枝，往往从已生成的树上减掉一些叶节点或叶节点以上的子树，并将其父节点或根节点作为新的叶节点，从而简化生成的决策树。决策树的减枝分为预剪枝和后减枝，预剪枝通过验证集的方式实现，后减枝可以通过验证集或者损失函数来实现。

相关知识点：

优点：

1 计算量简单，可解释性强，比较适合处理有缺失属性值的样本，能够处理不相关的特征。

缺点：

1 容易过拟合（后续出现了随机森林，减小了过拟合的现象）

逻辑回归

线性回归

感知机（perceptron）

感知机是根据输入实例的特征向量 x 对其进行二类分类的线性分类模型：

$$f(x) = \text{sign}(w \cdot x + b)$$

$$\text{sign}(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

感知机模型对应于输入空间（特徵空间）中的分离超平面

$$w \cdot x + b = 0$$

注意点：

1 感知机学习算法是基于随机梯度下降法的对损失函数的最优化算法，有原始形式和对偶形式。原始形式中，首先任意选取一个超平面，然后利用梯度下降法不断极小化目标函数。在这个过程中一次随机选取一个误分类点使其梯度下降。
2 感知机的损失函数是根据误分类点到超平面的总距离

$$\min_{w,b} L(w,b) = -\sum_{x_i \in M} y_i (w \cdot x_i + b)$$

其中 M 是误分类点的集合。这个损失函数就是感知机学习的经验风险函数。
3 当训练数据集线性可分时，感知机学习算法是收敛的。感知机算法在训练数据集上的误分类次数 k 满足不等式：

$$k \leq \left(\frac{R}{\gamma}\right)^2$$

当训练数据集线性可分时，感知机学习算法存在无穷多个解，其解由于不同的初值或不同的迭代顺序而可能有所不同。
4 采用随机梯度下降法进行优化，具体的实现要看书。
5 采用对偶形式的原因

关于感知机学习算法的对偶形式

我们假设样本点(x_i, y_i)在更新过程中被使用了n_i次。因此，从原始形式的学习过程可以得到，最后学习到的w和b可以分别表示为：

$$w = \sum_{i=1}^{N} n_i \eta \, y_i x_i \tag{1}$$

$$b = \sum_{i=1}^{N} n_i \eta \, y_i \tag{2}$$

考虑n_i的含义：如果n_i的值越大，那么意味着这个样本点经常被误分。什么样的样本点容易被误分？很明显就是离超平面很近的点。超平面稍微一点点移动，这个点就从正变为负，或者从负变为正。如果学过SVM就会发现，这种点很可能就是支持向量。

代入式(1)和式(2)到原始形式的感知机模型中，可得：

$$f(x) = sign(w \cdot x + b) = sign(\sum_{j=1}^{N} n_j \eta \, y_j x_j \cdot x + \sum_{j=1}^{N} n_j \eta \, y_j) \tag{3}$$

此时，学习的目标就不再是w和b，而是n_i，$i = 1, 2, \ldots, N$。

相应地，训练过程变为：
1. 初始时$\forall n_i = 0$。
2. 在训练集中选取数据(x_i, y_i)
3. 如果 $y_i(\sum_{j=1}^{N} n_j \eta \, y_j x_j \cdot x_i + \sum_{j=1}^{N} n_j \eta \, y_j) \leq 0$，更新：$n_i \leftarrow n_i + 1$
4. 转至2直至没有误分类数据。

可以看出，其实对偶形式和原始形式并没有本质区别，那么对偶形式的意义在哪里呢？从式(3)中可以看出，样本点的特征向量以内积的形式存在于感知机对偶形式的训练算法中，因此，如果事先计算好所有的内积，也就是Gram矩阵，就可以大大地加快计算速度。

相关知识点：

优点：

缺点：

1 只能数据集线性可分时才能使用，只能用于二分类。

支持向量机（SVM）

注意点：

1 支持向量机最简单的情况是线性可分支持向量机，或硬间隔支持向量机。构建它的条件是训练数据线性可分。其学习策略是最大间隔法。可以表示为凸二次规划问题，其原始最优化问题为

$$\min_{w,b} \frac{1}{2}||w||^2$$

$$st. \quad y_i(w \cdot x_i + b) - 1 \geq 0, i = 1,2,\cdots,N$$

求的最优化问题的解为w^*, b^*,得到线性可分支持向量机，分离超平面是

$$w^* \cdot x + b = 0$$

分类决策函数是

$$f(x) = sign(w^2 \cdot x + b^2)$$

最大间隔法中，函数间隔与几何间隔是重要的概念。
线性可分支持向量机的最优解存在且唯一。位于间隔边界上的实例点为支持向量。最优分离超平面有支持向量完全决定。
二次规划问题的对偶问题是

相关知识点：

优点：

缺点：

神经网络

GBDT 和 XGBOOST

随机森林

kmeans

EM