Model	理论	步骤	代价函数	优劣	备注
Logistic Regression	假设数据服从伯努利分布，通过极大化似然函数的方法，运用梯度下降来求解参数		$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$ $\theta_j := \theta_j + \alpha(y^i - h_\theta(x^i))x_j^i$	1.实现简单 2.分类时计算量非常小，速度很快，存储资源低	logistic regression
				1.容易欠拟合，一般准确度不太高 2.只能处理二分类问题，且必须线性可分	使用 softmax
SVM	1.拉格朗日乘子法 2.对偶问题 3.二次规划 4.SMO	1.优化目标函数 2.转换成拉格朗日形式 3.使用对偶理论转换目标函数 4.对 w,b 求导 $\mathcal{L}(\mathrm{w},\mathrm{b},\alpha) = 0.5 * w^T w + \sum_{n=1}^{N}\alpha_n\left(1 - y_n(w^T z_n + b)\right)$ $\mathrm{st.}\ \alpha_n \geq 0$ $\theta_p(w,b) = \max_{w,b,\alpha\geq 0}\mathcal{L}(\mathrm{w},\mathrm{b},\alpha) = \max_{w,b,\alpha\geq 0} 0.5 * w^T w + \sum_{n=1}^{N}\alpha_n(1 - y_n(w^T z_n + b))$ $\min_w 0.5 * w^T w = \min_w \theta_p(w,b) = \min_w \max_{w,b,\alpha\geq 0}\mathcal{L}(\mathrm{w},\mathrm{b},\alpha)$		1.可用于线性、非线性分类，也可回归 2.低泛化误差 3.容易解释 4.计算复杂度低	
				1.对参数和核函数的选择比较敏感 2.原始的 SVM 只擅长处理二分类问题	
KNN	投票表决	1.假设有一个带有标签的样本数据集（训练样本集），其中包含每条数据与所属分类的对应关系。 2.输入没有标签的新数据后，将新数据的每个特征与样本集中数据对应的特征进行比较。 a.计算新数据与样本数据集中每条数据的距离。 b.对求得的所有距离进行从小到大排序 c.取前 k（k 一般小于等于 20）个样本数据对应的分类标签。 3.求 k 个数据中出现次数最多的分类标签作为新数据的分类。		1.理论简单，可分类可回归，无需估计参数，无续训练 2.特别适合多分类问题，KNN 比 SVM 表现好。 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	
				1.计算量大，需要大量内存 2. 当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的 K 个邻居中大容量类的样本占多数。该算法只计算“最近的”邻居样本，某一	KD-Tree

				类的样本数量很大，那么或者这类样本并不接近目标样本，或者这类样本很靠近目标样本。无论怎样，数量并不能影响运行结果。	
KD-Tree	KD-Tree.md				
Decision-Tree	1.信息增益 2.信息增益率 3.Gini 系数			1.计算简单，可解释性强，比较适合处理有缺失属性的样本，能够处理不相关的特征 1.容易过拟合	随机森林
	ID3			1.切分过于迅速 2.不能直接处理连续型特征	
	C4.5			C4.5 只能做分类	
	Cart			CART 可以回归分析也可以分类	
树回归		对每个特征: 对每个特征值: 将数据集切分成两份（小于该特征值的数据样本放在左子树，否则放在右子树） 计算切分的误差 如果当前误差小于当前最小误差，那么将当前切分设定为最佳切分并更新最小误差 返回最佳切分的特征和阈值		优点：可以对复杂和非线性的数据建模。 缺点：结果不易理解。	
朴素贝叶斯	$\mathrm{P}(c_i\|\mathrm{w}) = \mathrm{P}(\mathrm{w}\|c_i)P(c_i)/P$			1.对小规模的数据表现良好，适合多分类任务，适合增量式训练 1.对输入数据的表达形式很敏感	
Boosting		先从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续受到更多关注，然后基于调整后的样本分布训练下一个基学习器；如此重复进行，直到基学习器达到事先指定的值 T，最终将这 T 个基学习器进行加权结合。		1.低泛化误差； 2.容易实现，分类准确率较高，没有太多的参数可调整 1.对 outlier 比较敏感	

Linear Regression	用梯度下降法对最小二乘法形式的误差函数进行优化		普通线性回归 $\sum_{i=1}^{m}(y_i-\theta^T x_i)^2$ $w=(X^TX)^{-1}X^Ty$	1.实现简单，计算简单	
			局部加权线性回归 $\sum_{i=1}^{m}w_i(y_i-\theta^T x_i)^2$ $w=(X^TWX)^{-1}X^TWy$	1.不能拟合非线性数据	
K-means	基于划分	1. 创建 k 个点作为起始质心（通常是随机选择） 2. 当任意一个点的簇分配结果发生改变时 2.1 对数据集中的每个数据点 2.1.1 对每个质心 2.1.2 计算质心与数据点之间的距离 2.1.3 将数据点分配到距其最近的簇 2.2 对每一个簇，计算簇中所有点的均值并将均值作为质心		1.算法简单、快速 2.对处理大数据集，该算法是相对可伸缩的和高效率的 3.时间复杂度近于线性，为 O(nkt)，适合挖掘大规模数据集。 4.当簇是密集、球状、团状且簇与簇之间区别明显时，聚类效果好	k-means.md k-means++:初始的聚类中心之间的相互距离要尽可能的远
				1.对初值敏感 2.不适合发现非凸面形状的簇，或者大小差别很大的簇 3.对噪声、孤立点数据敏感，少量的该类数据能够对平均值产生极大影响。 4.不断调整新的聚类中心，因此计算量非常大时算法的时间开销也非常大。	
Agnes	基于层次聚类自底向上聚合策略	1.先对仅含一个样本的初始聚类簇和相应的距离矩阵进行初始化； 2.然后不断合并距离最近的聚类簇，并对合并得到的聚类簇的距离矩阵进行更新 3.上述过程 1，2 不断重复，直到达到预设的聚类簇数。			
Dbsacn	基于密度聚类			1.将足够高密度的区域划分成簇，并能在具有噪声的空间数据库中发现任意形状的簇 2.在大规模数据库上更好的效率	
Wave Cluster、STING	基于网格的方法				

EM、SOM、COBWEB	基于模型的聚类				
GBDT	一种迭代的决策树算法，该算法由多棵决策树组成，所有树的输出结果累加起来就是最终答案。	其核心就在于，每一棵树是从之前所有树的残差中来学习的。			
EM	似然估计	E 步：选取一组参数，求出在该参数下隐含变量的条件概率值； M 步：结合 E 步求出的隐含变量条件概率，求出似然函数下界函数（本质上是某个期望函数）的最大值。 重复上面 2 步直至收敛。			
异常检测		将特征的每一维看成是相互独立的高斯分布，根据异常样本拟合每个特征的(u_j, σ_j^2)，然后在新的样本计算 P(x)，如果小于某阈值 ε，则认为 Anomaly			anomaly detection
Apriori	1.如果某个项集是频繁的，那么它的所有子集也是频繁的。 更常用的是它的逆否命题，即 2. 如果一个项集是非频繁的，那么它的所有超集也是非频繁的。	1.依据支持度找出所有频繁项集。 首先会生成所有单个元素的项集列表。接着扫描数据集来查看哪些项集满足最小支持度要求，那些不满足最小支持度的集合会被去掉。然后，对剩下来的集合进行组合以生成包含两个元素的项集。接下来，再重新扫描交易记录，去掉不满足最小支持度的项集。该过程重复进行直到所有项集都被去掉。 2.依据置信度产生关联规则。 **情况一**，当 frozenset 的长度为 2： 假设 freqSet={1,2}，则 H1={1},{2}； 计算置信度： freqSet-H1.element -> freqSet，即{2}->{1,2}、{1}->{1,2} **情况二**，当 frozenset 的长度大于 2： 1.假设 freqSet={1,2,3,4}时，则 H1={1},{2},{3},{4}； 2.首先计算置信度： freqSet-H1.element -> freqSet，即{1,2,3}->{1,2,3,4}、{2,3,4}->{1,2,3,4}、..； 3.递归计算频繁项集		1.简单，易理解 2.数据要求低 1.在每一步产生候选集时循环产生的组合过多，没有排除不应该参与组合的元素。 2.每次计算项集的支持度时，都对数据库中的全部记录进行了一遍扫描比较，如果是一个大型的数据库时，这种扫描会大大增加计算机的 I/O 开销。	FP-growth

		3.a 首先通过 aprioriGen 计算 Hm+1，H2={1,2}、{3,4}、.. 3.b 计算置信度： freqSet-H1.element -> freqSet ， 即 {1,2}->{1,2,3,4} 、{1,3}->{1,2,3,4}、..; 3.c 当 Len(Hm+1[0])+1<Len(freqSet)，则(3.a)；否则结束			
FP-growth		**首先，构建 FP 树**; 步骤一：1.遍历所有的数据集合，计算所有项的支持度; 2.丢弃非频繁的项; 3.基于支持度降序排序所有(元)项(不是项集)，生成到统计; 4.将所有数据集合中的每个集合按照得到的顺序(3)重新排序; 5.排序完成后，丢弃每个集合末尾非频繁的项 步骤二：6.读取每个集合插入到 FP 树，同时用一个头部链表数据结构维护不同集合的相同项，FP 树头结点为 null **其次,从 FP 树中挖掘频繁项集；** 7. 对头部链表进行降序排列 8. 对头部链表节点从小到大遍历，得到条件模式基 A1，同时获得一个频繁项集。 9. A1 条件模式基继续构造条件 FP 树(从(5)得到的集合中的数据构造)， 得到频繁项集，和之前的频繁项组合起来，这是一个递归遍历头部链表生成 FP 树的过程，递归截止条件是生成的 FP 树的头部链表为空。		1. 因为 FP-growth 算法只需要对数据集遍历两次，所以速度更快。 2. FP 树将集合按照支持度降序排序，不同路径如果有相同前缀路径共用存储空间，使得数据得到了压缩。 3. 不需要生成候选集。 4. 比 Apriori 更快。 1. FP-Tree 第二次遍历会存储很多中间过程的值，会占用很多内存。 2. 构建 FP-Tree 是比较昂贵的。	第一次遍历，得到所有频繁一项集的的计数。然后删除支持度低于阈值的项，将 1 项频繁集放入头部链表，并按照支持度降序排列。 第二次遍历，将读到的原始数据剔除非频繁 1 项集，并按照支持度降序排列。
BP 网路		1) 模式顺传播 ，输入模式由输入层经中间层向输出层的“模式顺传播”过程 2) 误差逆传播 ，网络的希望输出与网络实际输出之差的误差信号由输出层经中间层向输入层逐层修正连接权的“误差逆传播”过程 3) 记忆训练 ，由“模式顺传播”与“误差逆传播”的反复交替进行的网络“记忆训练”过程 4) 学习收敛 ，网络趋向收敛即网络的全局误差趋向极小值的“学习收敛”过程			
协同过滤		1）收集用户偏好 2）找到相似的用户或者物品			

		3）计算并推荐			
K-fold cross-validation		1.初始采样分割成 K 个子样本，一个单独的子样本被保留作为验证模型的数据，其他 K-1 个样本用来训练。 2.交叉验证重复 K 次，每个子样本验证一次，平均 K 次的结果或者使用其它结合方式，最终得到一个单一估测。		同时重复运用随机产生的子样本进行训练和验证，每次的结果验证一次，10 折交叉验证是最常用的	
留一验证		只使用原本样本中的一项来当做验证资料， 而剩余的则留下来当做训练资料。 这个步骤一直持续到每个样本都被当做一次验证资料。 事实上，这等同于和 K-fold 交叉验证是一样的，其中 K 为原本样本个数。			

$$\mathrm{H(X,Y)} = -\sum_{x,y} p(x,y)\log p(\mathrm{x,y})$$

$$\mathrm{H(X,Y)} - \mathrm{H(X)} = -\sum_{x,y} p(x,y)\log \mathrm{p}(y|x)$$

log[ ]

多元 GBDT 分类算法

$$p_k(\mathrm{x}) = \exp\left(f_k(\mathrm{x})\right) \Big/ \sum_{l=1}^{K} \exp(f_l(\mathrm{x}))$$

$$\mathrm{Loss} = \log\left[\prod_{i=1}^{n}\prod_{k=1}^{k} p_k(x_i)^{y_{ik}}\right]$$

$$\mathrm{L}\left(\{y_k, p_k(\mathrm{x})\}_1^k\right) = -\sum_{k=1}^{K} y_k \, log \, p_k(\mathrm{x})$$

$$h_m(x) = \sum_{j=1}^{J} c_{mj} I(x \epsilon R_{mj})$$

样本 k 负梯度误差

$$r_k = \partial \mathrm{L}\left(\{y_k, p_k(\mathrm{x})\}_1^k\right) \Big/ \partial f_k(x) = \partial \left[-\sum_{k=1}^{K} y_k \, log\left(\exp\left(f_k(\mathrm{x})\right) \Big/ \sum_{l=1}^{K} \exp(f_l(\mathrm{x}))\right)\right] \Big/ \partial f_k(x)$$

$$= \partial\left[-\sum_{k=1}^{K} y_k \left(log\ \exp\left(f_k(\mathrm{x})\right) - \log\sum_{l=1}^{K}\exp(f_l(\mathrm{x}))\right)\right] \Big/ \partial f_k(x)$$

$$= \partial[-\sum_{k=1}^{K} y_k f_k(\mathrm{x}) + \sum_{k=1}^{K}(y_k \log\sum_{l=1}^{K}\exp(f_l(\mathrm{x})))] \Big/ \partial f_k(x)$$

$$= \partial[-\sum_{k=1}^{K} y_k f_k(\mathrm{x})] \Big/ \partial f_k(x) + \partial[\sum_{k=1}^{K}(y_k \log\sum_{l=1}^{K}\exp(f_l(\mathrm{x})))] \Big/ \partial f_k(x)$$

$$= -y_k + \sum_{k=1}^{K} y_k * \exp(f_k(\mathrm{x})) \Big/ \sum_{l=1}^{K}\exp(f_l(\mathrm{x}))$$

$$= -y_k + 1 * p_k(\mathrm{x})$$