

数据分析方法

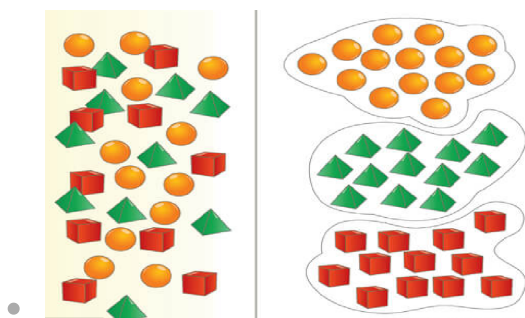
概述

- 数据挖掘与机器学习
 - 数据挖掘：从大量的数据中通过算法搜索隐藏于其中信息的研究
 - 机器学习：
 - 分类：根据输入数据，判别这些数据隶属于哪个类别
 - 预测：根据输入数据，计算一个输出值
 - T 代表任务(task)
 - P 代表任务 T 的性能(performance)
 - E 代表经验(experience)
 - 示例：
 - T (任务)
 - 定义: 任务是需要完成的具体目标或问题。在数据分析和机器学习的上下文中，任务通常指的是要解决的特定问题，比如预测、分类、聚类等。
 - 示例:
 - 预测学生是否能够考上研究生。
 - 预测下节课有多少学生旷课。
 - 学生根据兴趣聚成几个社团。
 - P (性能)
 - 定义: 性能是用来衡量任务完成质量的指标。在机器学习中，性能通常通过各种统计指标来评估模型的效果。
 - 常见性能指标:
 - 分类任务: 准确率、召回率、F1分数、ROC曲线、AUC值等。
 - 回归任务: 均方误差 (MSE)、平均绝对误差 (MAE)、决定系数 (R^2) 等。

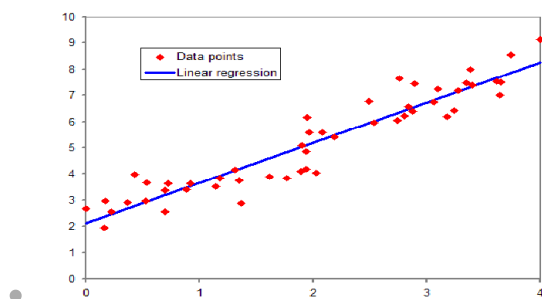
- 聚类任务: 轮廓系数、Davies-Bouldin指数等。
- 分类任务适用于需要将数据标记为**离散类别**的场景。
- 回归任务适合需要**预测连续数值**的场景。
- 聚类任务用于发现数据中的**自然分组或模式**，而无需事先定义标签。
- E (经验)
 - 定义: 经验是指在执行任务过程中获得的知识和技能。它包括对数据的理解、模型的选择、特征工程的技巧等。
 - 重要性:
 - 经验可以帮助数据科学家更好地理解数据集的特性，选择合适的算法，进行有效的特征选择和数据预处理。
 - 经验也能提高模型的调优能力，使得最终结果更加可靠和准确。
 - 分类任务: 需要对数据进行深入分析，了解影响数据的关键因素，并进行特征工程以提升模型性能。
 - 回归任务: 需要收集和分析过去的的数据，识别影响事件的因素，并考虑时间性和趋势性变化。
 - 聚类任务: 需要对影响数据的因素进行预处理，可能需要进行特征选择或降维，以提高聚类结果的可解释性和准确性。

- 数据分析的任务

- 分类: 预测的是离散值



- 回归: 预测的是连续值



- 测试(testing): 学得模型后可以采用样本对其进行预测
- 测试样本(testing sample): 被预测的样本
- 泛化(generalization): 学得模型适用于新样本的能力
- 模型评估方法
 - 留出法 (hold-out) : 将数据集D划分为两个互斥的集合, 其中一个集合作为训练集S, 另一个作为测试集 T
 - 交叉验证法 (Cross Validation) : 划分为K个大小相似的互斥子集, k-1个子集的并集作为训练集, 余下的子集作为测试集
 - 留一法 (Leave-One-Out, 简称LOO) : 每个子集只有一个样本数据
 - 优点:它不受随机样本划分的影响, 因为每个子集都会包含一个样本, 所以留一法的评估结果最准确。
 - 缺点:数据集较大时, 训练m个模型计算开销太大。
 - 自助法 (Bootstrapping) : 有放回抽样
 - 总结:

	采样方法	与原始训练数据集的分布是否相同	相比原始训练数据集的容量	是否适用小数据集	是否适用大数据集	是否存在估计偏差
留出法	分层抽样	否	变小	否	是	是
交叉验证法	分层抽样	否	变小	否	是	是
自助法	放回抽样	否	不变	是	否	是

回归

线性回归

- 目标：寻找直线 $y = \beta_0 + \beta_1 x$ 使得所有样本点尽可能落在它的附近
- 对 β_0 的偏导数： $\frac{\partial S}{\partial \beta_0} = -2 \sum_{i=1}^n (y_i - (\beta_0 + \beta_1 x_i)) = 0$
- 对 β_1 的偏导数： $\frac{\partial S}{\partial \beta_1} = -2 \sum_{i=1}^n x_i (y_i - (\beta_0 + \beta_1 x_i)) = 0$
 - $\triangleright \beta_1 = \frac{n \sum_{i=1}^n (x_i y_i) - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n (x_i^2) - (\sum_{i=1}^n x_i)^2}$
 - $\triangleright \beta_0 = \bar{y} - \beta_1 \bar{x}$
- 均方误差: $MSE = E(\omega_1, \omega_0) = \frac{\sum_{i=1}^n (y_i - \widehat{y_i})^2}{n} \widehat{y_i} = f(x_i)$

多元线性回归

- $Y = b_0 + b_1 X_1 + b_2 X_2 + \dots$
- 如果将样本矩阵 x_{ij} 记为矩阵 A , 将参数矩阵记为向量 β , 真实值记为向量 Y , 上述线性方程组可以表示为:

$$\bullet \quad \begin{vmatrix} 1 & x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ 1 & x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{m1} & x_{m2} & x_{m3} & \cdots & x_{mn} \end{vmatrix} \cdot \begin{vmatrix} \beta_0 \\ \beta_1 \\ \cdot \\ \beta_n \end{vmatrix} = \begin{vmatrix} y_0 \\ y_1 \\ \cdot \\ y_n \end{vmatrix}$$

- 即 $A\beta = Y$
- 解得: $\beta = (A^T A)^{-1} A^T Y$

```
from sklearn.datasets import load_boston
from sklearn.model_selection import cross_val_predict
from sklearn import linear_model
import matplotlib.pyplot as plt

lr = linear_model.LinearRegression()#导入线性回归
y = load_boston().target#导入Boston的回归值
# predicted返回预测结果
predicted = cross_val_predict(lr, load_boston().data, y,
cv=10)#十折交叉验证
fig, ax = plt.subplots()
```

```
ax.scatter(y, predicted, edgecolors=(0, 0, 0))
ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k-', lw=4)
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.show()
```

Logistic回归

- $z_i = x_i\theta, \theta = \hat{w}$
- $p_{\theta}(x_i) = \frac{e^{z_i}}{1+e^{z_i}} = \frac{1}{1+e^{-z_i}} = \frac{1}{1+e^{-x_i\theta}}$
 - 当 $p_{\theta}(x_i) < 0.5$ 时, 即 $z_i < 0$ 时, 则令 $y_i=0$ (预测为负)
 - 当 $p_{\theta}(x_i) > 0.5$ 时, 即 $z_i > 0$ 时, 则令 $y_i=1$ (预测为正)
 - $p_{\theta}(x_i)$ 值越小分类为0 (负) 概率越高, 反之分类为1 (正) 概率越大。
- 模型写成矩阵模式, 矩阵X的大小为 $n \times (d+1)$:
 - $p_{\theta}(X) = \frac{1}{1+e^{-X\theta}}$

```
from sklearn.datasets import load_iris
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

iris = load_iris()
X = iris.data
Y = iris.target
x_train, x_test, y_train, y_test = train_test_split(X, Y,
test_size = 0.3)
#数据集按7:3随机划分为训练集与测试集
lr = LogisticRegression(penalty='l2', solver='newton-
cg', multi_class='multinomial')
lr.fit(x_train, y_train)
print("训练集的准确率: %.3f" %lr.score(x_train, y_train))
print("测试集的准确率: %.3f" %lr.score(x_test, y_test))
```

参数解释:

`penalty='l2'`: 惩罚项, 选择使用 L2 正则化 (也称为岭回归)

`solver='newton-cg'`: 求解器, 使用 Newton-CG 算法来优化逻辑回归模型。

`multi_class='multinomial'`: 多分类策略, 使用多项式逻辑回归来处理多分类问题。

- 例子:
- 数据集:

学习时间 (小时)	参加复习班	是否通过 (1=通过, 0=未通过)
2	0	0
3	0	0
4	1	1
5	1	1
6	1	1

- 模型构建
 - 逻辑斯谛回归模型的形式为:
 - $P(y = 1|x) = \sigma(z) = \frac{1}{1+e^{-z}}$
 - 其中, $(z = \beta_0 + \beta_1 x_1 + \beta_2 x_2)$ 。
- 在这个例子中:
 - (x_1) : 学习时间 (study_hours)
 - (x_2) : 参加复习班 (attend_class)
- 设定初始参数
 - 假设我们初始设置参数为:
 - $(\beta_0 = -4)$
 - $(\beta_1 = 1)$
 - $(\beta_2 = 2)$

- 计算概率

- **第1条数据** (学习时间=2, 参加复习班=0) :

- $z = -4 + 1 \times 2 + 2 \times 0 = -2$

- $P(y = 1|x) = \frac{1}{1+e^2} \approx 0.1192$

- **第2条数据** (学习时间=3, 参加复习班=0) :

- $z = -4 + 1 \times 3 + 2 \times 0 = -1$

- $P(y = 1|x) = \frac{1}{1+e^1} \approx 0.2689$

- **第3条数据** (学习时间=4, 参加复习班=1) :

- $z = -4 + 1 \times 4 + 2 \times 1 = 0$

- $P(y = 1|x) = \frac{1}{1+e^0} = 0.5$

- **第4条数据** (学习时间=5, 参加复习班=1) :

- $z = -4 + 1 \times 5 + 2 \times 1 = 1$

- $P(y = 1|x) = \frac{1}{1+e^{-1}} \approx 0.7311$

- **第5条数据** (学习时间=6, 参加复习班=1) :

- $z = -4 + 1 \times 6 + 2 \times 1 = 2$

- $P(y = 1|x) = \frac{1}{1+e^{-2}} \approx 0.8808$

- 预测结果

- 根据计算的概率, 我们可以将预测的概率与阈值 (通常为0.5) 进行比较, 得到预测结果:

- 第1条数据: $0.1192 < 0.5 \rightarrow$ 预测为 0 (未通过)

- 第2条数据: $0.2689 < 0.5 \rightarrow$ 预测为 0 (未通过)

- 第3条数据: $0.5 = 0.5 \rightarrow$ 预测为 1 (通过)

- 第4条数据: $0.7311 > 0.5 \rightarrow$ 预测为 1 (通过)

- 第5条数据: $0.8808 > 0.5 \rightarrow$ 预测为 1 (通过)

- 最终的预测结果为:

学习时间 (小时)	参加复习班	实际是否通过	预测是否通过
2	0	0	0
3	0	0	0

学习时间 (小时)	参加复习班	实际是否通过	预测是否通过
4	1	1	1
5	1	1	1
6	1	1	1

非线性回归

- 最简单的非线性函数：一元多项式函数
 - 2次： $y = ax^2 + bx + c$ (抛物线)
 - n次： $y = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + b$
- 残差的计算公式： $RSS = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2$
- 过拟合问题：高方差，泛化能力差
 - 处理方式：
 - 减少特征数量
 - 正则化
 - L1正则化的模型建叫做Lasso回归
 - L2正则化的模型叫做Ridge回归（岭回归）
 - Lasso:
 - $L(\omega) = \|\omega\|_1$
 - $\|\omega\|_1 = \sum_{i=1}^n |\omega_i|$
 - $J = J_0 + \alpha \|\omega\|_1 = J_0 + \alpha \sum_w |w|$
 - 岭回归（Ridge regression）：
 - $L(\omega) = \|\omega\|_2$
 - $\|\omega\|_2 = \sqrt{\sum_{i=1}^n \omega_i^2}$
 - $J = J_0 + \alpha \|\omega\|_2 = J_0 + \alpha \sum_w w^2$

回归模型的性能度量

- 平均绝对误差(Mean absolute error, MAE)
 - $MAE = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n}$

- $\hat{y}_i = f(x_i)$
- MAE又被称为L1范数损失，表示预测值与观察值之间的绝对误差的平均值
- 均方误差(Mean squared error MSE)
 - $MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$
 - MSE又被称为L2范数损失，表示预测值与观察值之间的误差平方的平均值
- 均方根误差 (RMSE)
 - $RMSE = \sqrt{MSE}$
- 决定系数: R^2 (R-Square)
 - $R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (\bar{y} - y_i)^2}$
 - 分母为标签Y的方差，分子为MSE
 - 缺点：随着样本数量的增加 R^2 值会随之增加，所以无法定量的说明准确程度。
- 校正决定系数: $Adjusted R^2$ (Adjusted R-Square)
 - $Adjusted R^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1}$
 - n:样本数量
 - p:特征数量
 - Adjusted R^2 抵消了样本数量对 R^2 的影响，可以定量的说明准确程度。

分类

分类性能评估

- 混淆矩阵:
 - 真正(True Positive , TP): 被模型预测为正的正样本。
 - 假正(False Positive , FP): 被模型预测为正的负样本。
 - 假负(False Negative , FN): 被模型预测为负的正样本。
 - 真负(True Negative , TN): 被模型预测为负的负样本。

真实数据	预测结果	
	正样本	负样本
正样本	TP	FN
负样本	FP	TN

- 准确率 (Accuracy) :

- 所有被分类正确的点在所有点中的概率

- $Accuracy = \frac{(TP+TN)}{(TP+FN+FP+TN)}$

- 精确率 (Precision) :

- 表现为预测出是正的里面有多少真的为正的的概率

- $Precision = \frac{TP}{TP+FP}$

- 召回率 (Recall) :

- 正确预测的正例数 / 实际正例总数, 也称查全率

- $Recall = \frac{TP}{TP+FN}$

- F1 score:

- F值是精确率和召回率的调和值

- $\frac{2}{F1} = \frac{1}{Precision} + \frac{1}{Recall}$

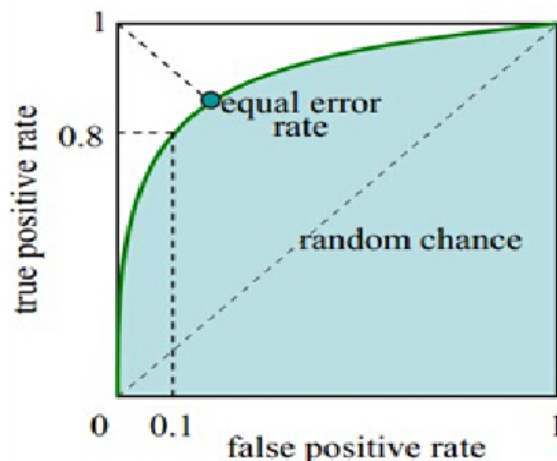
- ROC曲线:

- TPR: 在所有实际为阳性的样本中, 被正确地判断为阳性的比率。

- $TPR = \frac{TP}{TP+FN}$

- FPR: 在所有实际为阴性的样本中, 被错误地判断为阳性的比率。

- $FPR = \frac{FP}{TN+FP}$



- AUC (Area Under Curve) :

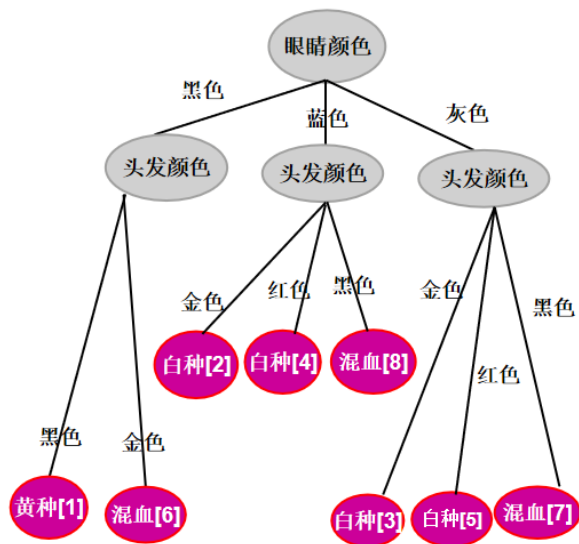
- ROC曲线下的面积(ROC的积分)
- $AUC = 1$, 是完美分类器
- $0.5 < AUC < 1$, 优于随机猜测。
- $AUC = 0.5$, 模型没有预测价值
- $AUC < 0.5$, 比随机猜测还差

决策树

- 与决策树相关的重要算法包括: CLS, ID3, C4.5, CART
- CLS:

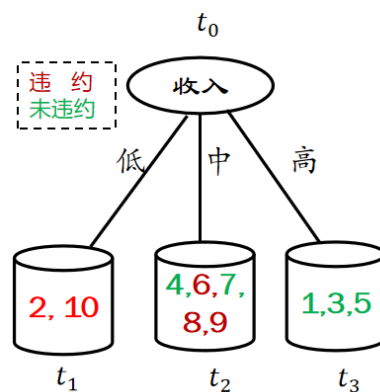
人员	眼睛颜色	头发颜色	所属人种
1	黑色	黑色	黄种人
2	蓝色	金色	白种人
3	灰色	金色	白种人
4	蓝色	红色	白种人
5	灰色	红色	白种人
6	黑色	金色	混血
7	灰色	黑色	混血
8	蓝色	黑色	混血

- 转化成决策树:



• ID3算法

- 挑选使信息增益最大的特征进行分裂
- 从根节点开始，选择“某”属性特征
 - 节点不纯度的度量：
 - 信息熵 (Entropy)
 - Gini指数 (Gini index)
 - 误分率 (Misclassification error)
 - 不纯度越小越好
 - 节点 t 的信息熵为：
 - $Entropy(t) = -\sum_{c=1}^c p(c|t) \log_2 p(c|t)$
 - 当样本均匀分布在每一个类中时，熵为 $\log_2 C$ ，说明不纯度大；
 - 当所有的样本属于同一个类时，熵为0，说明不纯度小。



$$Entropy(t_1) = -\frac{2}{2} \log_2 \frac{2}{2} - \frac{0}{2} \log_2 \frac{0}{2} = 0;$$

$$Entropy(t_2) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.971;$$

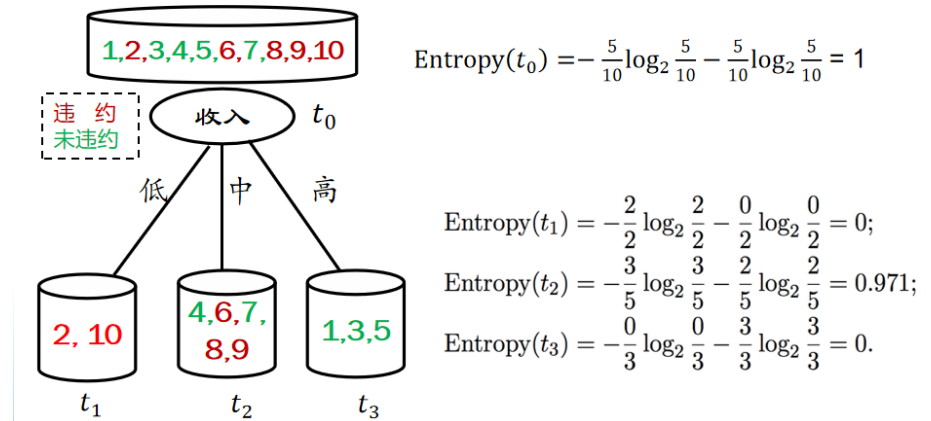
$$Entropy(t_3) = -\frac{0}{3} \log_2 \frac{0}{3} - \frac{3}{3} \log_2 \frac{3}{3} = 0.$$

- 信息增益 (Information Gain) :

- 节点分裂前后的信息熵的下降值

- $InfoGain = Entropy(t_0) - \sum_{k=1}^k \frac{n_k}{n} Entropy(t_k)$

- n_k 是子节点 t_k 的样本数量, n 是父节点 t_0 的样本数量



- $InfoGain(收入) =$

$$Entropy(t_0) - (\frac{2}{10} \times 0 + \frac{5}{10} \times 0.971 + \frac{3}{10} \times 0)$$

- Gini指数

- 节点 t 的Gini指数

- $Gini(t) = 1 - \sum_{c=1}^C [p(c|t)]^2$

- 当样本均匀分布在每一个类中, Gini指数为 $1 - \frac{1}{C}$, 说明不纯度大
- 当样本都分布在同一个类中, Gini指数为0, 说明不纯度小

- 误分率

- 当按照多数类来预测当前节点样本的类别时, 被错误、分类的数据的比例

- 节点 t 的误分率为:

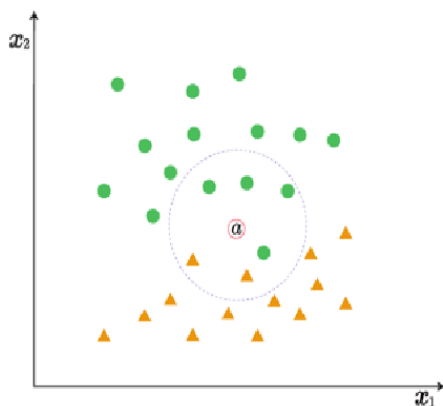
- $Error(t) = 1 - \max(p(1|t), p(2|t), \dots, p(C|t))$

- 当样本均匀分布在每一个类中, 误分率为 $1 - \frac{1}{C}$, 不纯度最大
- 当样本分布在同一个类中, 误分率为0, 不纯度最小

- 缺点：倾向于分裂成很多的小节点（节点的样本数较小），容易造成过拟合
- C4.5算法
 - 采用信息增益率，避免分裂成过多的子节点
 - $SplitInfo = - \sum_{k=1}^k \frac{n_k}{n} \log_2(\frac{n_k}{n})$
 - $InfoGainRatio = \frac{InfoGain}{SplitInfo}$
- 决策树剪枝：减少过拟合，提高决策树的泛化能力
- 预剪枝：在构造树的同时停止信息量较少，也就是信息增益或信息增益比较小的分枝。
- 后剪枝：先生成树，再进行剪枝。
- 两种剪枝策略对比：
 - 后剪枝决策树通常比预剪枝决策树保留了更多的分支；
 - 后剪枝决策树的欠拟合风险很小，泛化性能往往优于预剪枝决策树；
 - 后剪枝决策树训练时间开销比未剪枝决策树和预剪枝决策树都要大的多。

K-近邻算法

- 预测方法：



- @ 的6个邻居样本中，有4个正类，2个负类，因此@的标签为正类
- 流程：
 - 确定 k 的大小和距离计算方法；

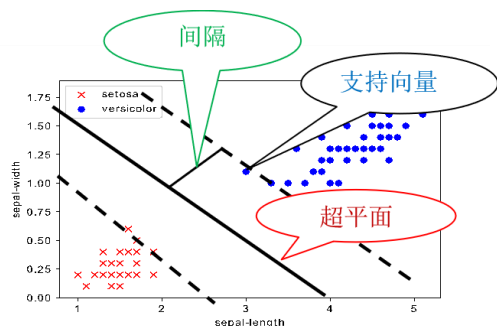
- 遍历数据集，采用距离公式计算距离，获取前 k 个与目标数据距离最小的样本；
 - 根据 k 个最相似的训练样本的类别，通过投票的方式来确定测试样本的类别。
- 距离度量：
 - L1 范数（曼哈顿距离）：
 - $\|x\|_1 = \sum_{i=1}^n |x_i|$
 - L2 范数（欧几里得距离）：
 - $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$
 - 无穷范数（切比雪夫距离）：
 - $\|x\|_\infty = \max_i |x_i|$
 - 曼哈顿距离 ($p=1$) : $d(x_1, x_2) = \sum_{i=1}^d |x_{1i} - x_{2i}|$
 - 欧氏距离 ($p=2$) : $d(x_1, x_2) = \sqrt{\sum_{i=1}^d (x_{1i} - x_{2i})^2}$
 - 切比雪夫距离 ($p=+\infty$) : $d(A, B) = \max(|x_1 - x_2|, |y_1 - y_2|)$
 - 余弦距离: $\cos(x_1, x_2) = \frac{x_1^T x_2}{\|x_1\|_2 \|x_2\|_2}$
 - 杰卡德距离(Jaccard Distance) : 略
- K近邻算法最常用的数据结构为 k-d树，它是二叉搜索树在多维空间上的扩展
- 优点：
 - 简单，易于理解，易于实现，无需估计参数，无需训练；
 - 特别适合于多分类问题(multi-modal,对象具有多个类别标签)
- 缺点：
 - 当样本不平衡时，样本容量较小的类域采用这种算法比较容易产生误分；
 - 该方法的另一个不足之处是计算量较大，因为对每一个待分类的文本都要计算它到全体已知样本的距离；
 - 可理解性差，无法给出像决策树那样的规则。

朴素贝叶斯算法

- 根据训练集，推导出一个概率公式，然后计算概率，根据概率的大小，确定将该归为哪一类
- 贝叶斯公式：
 - $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$
 - 假设各个属性之间相互独立：
 - $P(A, B) = P(A)P(B)$ $P(A, B|X) = P(A|X)P(B|X)$
 - 例子：
 - $P(\text{违约} | \text{男, 中, 本科, 未婚}) = \frac{P(\text{男} \text{ \ , 中, 本科, 未婚} | \text{违约})}{P(\text{男, 中, 本科, 未婚})}$
 - 其中： $P(\text{男, 中, 本科, 未婚}) = P(\text{男})P(\text{中})P(\text{本科})P(\text{未婚})$
 $P(\text{男, 中, 本科, 未婚} | \text{违约}) = P(\text{男} | \text{违约})P(\text{中} | \text{违约})P(\text{本科} | \text{违约})P(\text{未婚} | \text{违约})$
 - 特征之间相互独立：
 - $P(X=X_i | Y=k) = \prod_{j=1}^d P(X_{ij}=x_{ij} | Y=k)$
 - 预测样本 X_i 最有可能的类别标签 y :
 - $y = \arg\max_k (\prod_{j=1}^d P(X_{ij}=x_{ij} | Y=k) P(Y=k))$
- 0概率处理
 - 加1平滑： $P(B_i) = \frac{0+1}{N}$
- 伯努利模型
 - 特征是离散的且所有取值分别仅有两个值
- 高斯模型
 - $P(X_i = x | Y = c) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

支持向量机 (SVM)

- 超平面：将样本数据分为两大块的边界



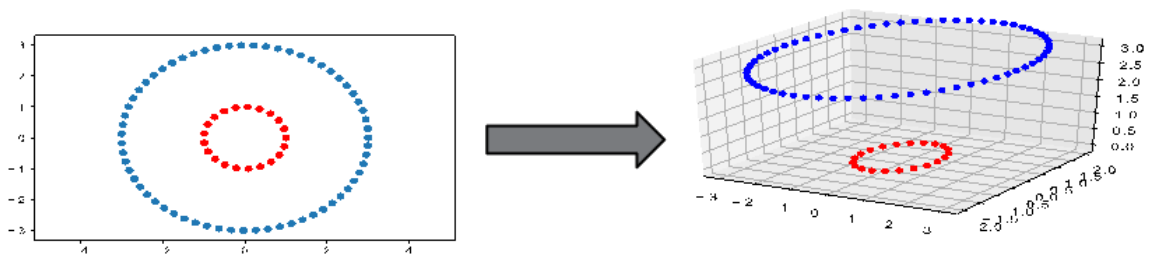
- 示例：

- 假设我们有一个简单的数据集，包含两类点，类别A和类别B。数据集如下：
 - 类别A（正类）：(2, 3), (3, 3), (4, 5)
 - 类别B（负类）：(1, 1), (2, 1), (3, 2)
- 数据可视化
- 我们可以将这些点在二维平面上绘制出来：

A: (2, 3), (3, 3), (4, 5)

B: (1, 1), (2, 1), (3, 2)
- 找到超平面：
 - SVM的目标是找到一个超平面，将两类数据分开。超平面可以表示为： $w \cdot x + b = 0$
 - 其中，(w) 是权重向量，(b) 是偏置项。
- 确定支持向量
 - 支持向量是离决策边界最近的点。在这个例子中，可能的支持向量是类别A和类别B中的某些点。
- 计算权重和偏置
 - 选择支持向量：
 - 假设选择 (3, 3) 和 (2, 1) 作为支持向量。
 - 构建方程：
 - 我们需要解以下约束条件：
 - $(w \cdot (3, 3) + b \geq 1)$ (对于类别A)
 - $(w \cdot (2, 1) + b \leq -1)$ (对于类别B)
 - 优化问题：
 - 最小化目标函数： $\frac{1}{2} ||w||^2$
 - 使得上述约束条件成立。
- 结果
 - 通过求解优化问题，我们可以得到权重 (w) 和偏置 (b)。假设我们最终得到：
 - (w = [1, 1])
 - (b = -4)

- 分类新点
 - 现在我们可以使用这个模型来分类新的点。例如，点 (3, 4):
 - 计算: $w \cdot (3, 4) + b = 1 \cdot 3 + 1 \cdot 4 - 4 = 3$
 - 由于结果大于0，这个点被分类为类别A。
- 罚分值 (penalty) :对每个误分类的数据点，结合其与超平面的距离计算其罚分 ξ ，对罚分进行求和
- 超参数: C越大容错空间越小 反之C越小容错空间越大
- 非线性划分:



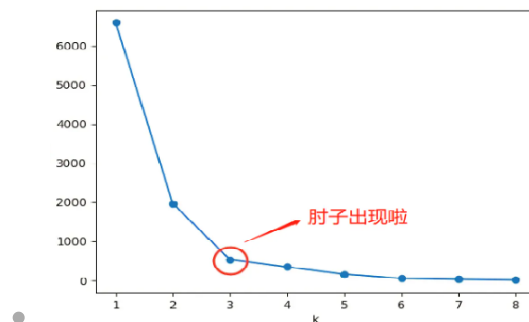
- 即进行高纬度转化，然后进行超平面划分
- 核函数: 核函数可以将原始特征映射到另一个高维特征空间中，解决原始空间的线性不可分问题
 - 高斯核函数: $K(x, y) = e^{-\gamma \|x - y\|^2}$
 - 超参数gamma: γ
- SVM算法实现:
 - 确定分类边界
 - 通过将同一类的数据点两两相连，它们的边界就是该类的边界。因为线性可分，所以不同类别边界不存在交叉
 - 找出最佳超平面
 - 找出最佳超平面，最佳超平面应使间隔最大化的同时使罚分值最小
 - 分类
 - 判断测试集位于超平面的哪一边，实现最终的分类任务

聚类

- 由聚类所生成的簇是一组数据对象的集合，这些对象与同一个簇中的对象彼此相似，与其他簇中的对象相异。

原型聚类

- K-MEANS聚类：
 - 寻找质心：从数据集中随机选取k个分散的样本作为聚类分析的初始质心
 - 划分数据点：计算每个点与各个质心的距离，将数据点划分到距离最近的质心所在的簇内
 - 寻找新的质心：按照新簇计算每一个簇的新质心
 - 反复计算并更新质心
 - 优点：
 - 当簇接近高斯分布时，它的效果较好。
 - 对处理大数据集，该算法保持可伸缩性和高效性；
 - 缺点：
 - 只对球状数据聚类效果好
 - K值的确定：
 - 先验法（根据数据进行一个大致的估计）
 - 手肘法：
 - $WSS_k = \sum_{i=1}^k \| x_i - \mu_k \|^2$
 - 我们通过确定WSS随着K的增加而减少的曲线拐点，作为K的取值。

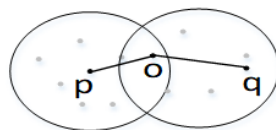


- 对离群点敏感:可以采用K-Medoids来代替K-Means
- 对初始中心选择敏感：计算所有的样本值，对比选出K个聚类中心

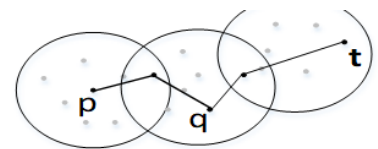
- 大规模数据计算成本高：通过每次仅使用一个小的随机子集来更新聚类中心，从而减少计算时间

密度聚类

- 以数据集在空间分布上的稠密程度为依据进行聚类
 - 即只要一个区域中的样本密度大于某个阈值，就把它划入与之相近的簇中。
- DBSCAN算法：
 - 设定邻域参数(ϵ , $Minpts$)
 - 以各个数据点为中心，以 ϵ 为半径，计算其 ϵ -邻域内的数据点密度。
 - $Minpts$ 代表一个簇中最少的数据点个数
 - $\rho > Minpts$ 高密度区域
 - $\rho < Minpts$ 低密度区域
 - 数据点分类
 - 将数据点进行分类
 - 核心点: $\rho > Minpts$ 的点
 - 边界点:位于核心点邻域之内,但 $\rho < Minpts$, 起到将高密度区域与低密度区域分割开的作用
 - 噪声点: 既不是核心点也不是边界点的其他数据点, 他们组成低密度区域
 - 密度直达: 两个同属于一个邻域的数据点
 - 密度可达: $p \rightarrow o \rightarrow q$
 - 密度相连: q 和 p 是密度可达的, q 和 t 也是密度可达的, 则 p 和 t 是密度相连的。



密度可达



密度相连

- 聚类


- 首先核心点各自成簇，采用密度相连的概念逐步对簇进行合并，打上标记。
- 最终核心点密集的区域会被低密度噪声点包围，噪声点不单独成簇
- 噪声点没有被标记

层次聚类

- 自顶而下：分裂层次聚类
- 自底向上：凝聚层次聚类
- 簇间距离的计算
 - 单连接
 - 两簇之间最近的成员之间的距离
 - 全连接（完整连接）
 - 两簇之间最远的成员之间的距离
 - 平均连接
 - 表示两簇间所有成员对的平均距离
- AGENS（类似哈夫曼树结构）
- 刚开始共有5个簇：C1={A}、C2={B}、C3={C}、C4={D}和C5={E}

样本点	A	B	C	D	E
A	0	0.4	2	2.5	3
B	0.4	0	1.6	2.1	1.9
C	2	1.6	0	0.6	0.8
D	2.5	2.1	0.6	0	1
E	3	1.9	0.8	1	0

- Step1:
 - 簇C1和簇C2合并，得到新的簇结构：C1={A,B}、C2={C}、C3={D}和C4={E}。

- Step2:
 - 簇C2和簇C3合并, 得到新的簇结构: $C1=\{A,B\}$ 、 $C2=\{C,D\}$ 和 $C3=\{E\}$
- Step3:
 - 簇C2和簇C3合并, 得到新的簇结构: $C1=\{A,B\}$ 和 $C2=\{C,D,E\}$
- Step4:簇C1和簇C2合并
 - 假设我们使用平均链接法,那么计算过程如下:
 - 簇AB与簇C的距离 = (簇A与簇C的距离 + 簇B与簇C的距离) / 2 = $(1.6 + 1.6) / 2 = 1.6$
 - 簇AB与簇D的距离 = (簇A与簇D的距离 + 簇B与簇D的距离) / 2 = $(2.0 + 2.0) / 2 = 2.0$
 - 簇AB与簇E的距离 = (簇A与簇E的距离 + 簇B与簇E的距离) / 2 = $(2.4 + 2.4) / 2 = 2.4$
-  ![[Pasted image 20250107220626.png|300]]

模型性能评估 —— 聚类性能度量 (有效性指标)

- 好的聚类结果
 - 聚类结果的簇内相似度高且保证簇间相似度低。
- 外部指标
 - 与参考模型进行比较, 被划分在同一簇样本与参考模型样本也被同样划分到一个簇的概率越高越好。
 - $a+b+c+d=n(n-1)/2$
 - Jaccard系数 (Jaccard Coefficient, JC)
 - $JC = \frac{a}{a+b+c}$
 - FM指数 (Fowlkes and Mallows Index, FMI)
 - $FMI = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}}$
 - Rand指数 (Rand Index, RI)

- $RI = \frac{2(a+d)}{n(n-1)}$
- 以上性能度量指标的取值均在区间[0,1]内，取值越大代表聚类效果越好。
- 内部指标
 - 通过计算簇内样本间的距离以及簇间样本的距离评估模型的性能
 - DB指数 (Davies-Bouldin Index, DBI)
 - $DBI = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left(\frac{\text{avg}(C_i) + \text{avg}(C_j)}{d_{\text{cen}}(C_i, C_j)} \right)$
 - Dunn指数 (Dunn Index, DI)
 - $DI = \min_{1 \leq i \leq k} \left\{ \min_{j \neq i} \left(\frac{d_{\min}(C_i, C_j)}{\max_{1 \leq x \leq k} \text{diam}(C_x)} \right) \right\}$
 - DBI值越小越好，DI值越大越好

关联规则挖掘

- 置信度 $P(A \Rightarrow B) = P(B|A) = \frac{P(A \cup B)}{P(A)}$
- 支持度
- 强关联规则：达到预设的最小支持度阈值和最小置信度阈值，成为强关联规则
- 步骤：
 - 先找出所有满足最小支持度的商品集合
 - 通过这些集合生成关联规则
 - 最后利用置信度筛选出强关联规则
- 以购物的例：
 - 事务：每一条购物信息可以被定义为一个事务；
 - 项：购物信息中的一项物品被定义为项；
 - 项集：包含零个项或多个项的集合被称为项集，含有k个项的项集称为k项集；
 - 关联规则就可以表示为形如“A=>B”的蕴涵式，其中A，B均为非空项集，且 $A \cap B = \emptyset$ 。
 - 支持度大于预定义的最小支持度阈值的项集称为频繁项集。
- Apriori性质
 - 如果一个项集A是频繁项集，那么它的非空子集B也是频繁项集。

- 或者：如果一个项集不是频繁项集，那它的超集也不是频繁项集。
- Apriori算法
 - 基本过程：见PPT 289-301
- FP-Growth 算法
 - 步骤：
 - 第一步根据原始数据集构造频繁模式树（FP树）；
 - 第二步基于频繁模式树挖掘频繁项集。

PageRank

- PageRank是一种在搜索引擎中根据网页之间相互的链接关系计算网页排名的技术
- 其值为0-10，PR值越高说明该网页越受欢迎（越重要）
- 核心思想：
 - 链入链接数 (单纯的意义上的受欢迎度指标)
 - 链入链接是否来自推荐度高的页面 (有根据的受欢迎指标)
 - 链入链接源页面的链接数 (被选中的几率指标)
- 计算方法：

$$PR_i = \sum_{j \in B_i} \frac{PR_j}{L_j}$$

PR_i : 网页*i*的pagerank值

PR_j : 网页*j*的pagerank值

L_j : 网页*j*链出的连接数

B_i : 链接到网页*i*的网页集合

- 转移矩阵为M
- 初始状态Rank值为 V_0
- $V^k = M^k V_0$
- M^k 是收敛的，所以迭代到一定次数以后， V^k 趋于稳定
- Dead End问题：见PPT举例说明

- Rank Sink: 整个网页图中的一组紧密链接成环的网页如果没有外出的链接就产生Rank Sink
- 链接作弊问题: 人为堆砌大量连接
- 交换链接: 人为地互相增加对方网站的链接
- 黄金链: 高权重的网站出售首页的链接给作弊网站

集成学习

- 结合各个学习器的优点, 获得较单一学习器更优越的泛化性能
- 结合策略:
 - 平均法: 对于若干个弱学习器的输出进行平均得到最终的预测输出, 一般用于对于数值类的回归预测问题。
 - 投票法: 用于对于分类问题的预测
 - 相对多数投票法: 选择预测次数多的
 - 绝对多数投票法: 结果中, 出现次数超过一半
 - 加权投票法: 票数乘权重计算
 - 学习法:
 - 过另一个学习器来结合预测结果
 - 将训练集弱学习器的学习结果作为输入, 将训练集的输出作为输出, 重新训练一个学习器来得到最终结果。

序列化方法

- 个体学习器之间具有强依赖关系, 必须串行生成
- Boosting算法族 (AdaBoost, GBDT, Xgboost)
 - AdaBoost实现步骤:
 - 用于特征选择(Feature Selection)
 - 用于二值分类或多分类的应用场景
 - 其中 w 代表样本权重值, α 代表模型权重值, $b_k(x)$ 代表第 k 个基础模型, 初始值为1
 - 给每个训练样本附上权重值: $w_1(i) = \frac{1}{n}$

- 训练集的抽样子集，训练出第k个基础模型 $b_k(x)$
- 计算该基础模型误差率:

$$e_k = \frac{1}{n} \sum_{i=1}^n w_k(i) \cdot I(b_k(x_i) \neq y_i)$$
 - 其中 y_i 为第i个样本的标记值， $b_k(x_i)$ 为样本的预测值。如果预测正确， $I(x)=0$ 否则 $I(x)=1$ 。
- 该模型的权重值计算为: $\alpha_k = \ln(1 - e_k) / e_k$
- 根据新模型的预测值更新训练集的权重:

$$w_{k+1}(i) = w_k(i) \cdot e^{\alpha_k \cdot F(b_k(x_i) \neq y_i)}$$
 - 其中如果预测正确， $F(x)=-1$ ；如果预测错误， $F(x)=1$
- 跳转回第2步，直到k到达指定值退出循环。

并行化方法

- 个体学习器之间不存在强依赖关系，可同时并行生成
- Bagging
- 随机森林 (Random Forest, RF)
 - 假如有N个样本，则有放回的随机选择N个样本。这选择好了的N个样本用来训练一个决策树。
 - 当每个样本有M个属性时，在决策树的每个节点需要分裂时，随机从这M个属性中选取m个属性，满足条件 $m \ll M$ 。然后从这m个属性中采用某种策略（比如说信息增益）来选择1个属性作为该节点的分裂属性。
 - 决策树形成过程中每个节点都要按照步骤2来分裂，一直到不能够再分裂为止。决策树形成过程中没有进行剪枝。
 - 按照步骤1-3构造多个决策树，最终构成随机森林。