

**Data Preprocessing** : **Accuracy** : errors in recorded values, outlier values (unexpected) ex : enormous price for a cheap item once (introduced 0 by mistake) **Completeness** : lacking attribute values, only aggregate data, ... ex : sometimes type of payment (card/cash) not recorded **Consistency** : discrepancies in values used ex : some department codes with leading 0, some without **Timeliness** : data missing or incomplete for some time, but eventually complete ex : data recorded by person too late for monthly analysis, but available afterwards **Believability** : Not trusted by users ex : due to a previous bug (now corrected), people distrust system and use manual alternatives **Interpretability** : ambiguous data ex : adjustment orders (not real but to correct mistake/complaint) can be confused as real orders.

**Tasks** : **Data cleaning** (fill missing value, smoothing, outliers, inconsistencies), **Data integration** (multiple database,...), **Data reduction** (Dimensionality reduction, Numerosity reduction, Data compression), **Data transformation** : Smoothing, Attribute/feature construction, Aggregation

**Données incomplètes** : ignoré tuple, remplir à la main, automatiquement (mean,std,...). **Clustering** : detect and remove outliers. Coefficient de corrélations :  $p'_k = (p_k - \bar{p})/\sigma_p$ ,  $q'_k = (q_k - \bar{q})/\sigma_q$ ,  $corr(p, q) = p' \cdot q'$ . **Mode of data** : données qui apparait le plus, **midrange of data** :  $min + max/2$ ,  $\sigma_p = \sqrt{v(p)}$ ,  $v(p) = \sum (p_i - \bar{p})^2$ .

**Data reduction** : **Dimensionality reduction** (attribut inutiles), techniques : Wavelet transforms, Principal Component Analysis, Supervised and nonlinear techniques. **Numerosity reduction** : régression, histogramme, data cube. **Data compression** : Audio/video compression, string compression.

**Data Transformation** : **min-max** :  $v' = \frac{v - min_A}{max_A - min_A} \cdot (new_{max_A} - new_{min_A}) + new_{min_A}$ . **z-score** :  $v' = \frac{v - \bar{A}}{\sigma_A}$ . **decimal-scaling** :  $v' = \frac{v}{10^j}$   $j$  is the smallest integer such that  $|max(v)| < 1$ . **Binnings method for data smoothing** : D'abord, séparer les données en  $n$  groupe différent. Puis, pour chaque groupe, remplacer par la moyenne (**smoothing by mean**), le min ou max le plus proche (**smoothing by boundaries**), la médiane **smoothing by median**.

**Market-Basket analysis** : permet de trouver les groupes d'articles qui ont tendance à apparaître ensemble. On dispose des transactions. Soit  $I$  un ensemble d'article,  $D$  un ensemble de transactions, chaque transactions  $T_j$  est un ensemble d'article. Règle de dissociation (comme associations mais peut avoir non).  $Support(X \Rightarrow Y) = P(X \text{ et } Y)$ .  $Confiance(X \Rightarrow Y) = P(Y|X) = P(X \text{ et } Y)/P(X)$ .  $Lift(X \Rightarrow Y) = Confiance/Support$ . **Agrégation** : agréger produits très similaires en produit moins spécifique. **Taxonomie** : Généralisation par catégorisation (p.ex : "Vêtement"). À partir de seuil définis par nous  $min_{sup}$ ,  $min_{conf}$ , trouver les règles intéressantes (juste regarde si ok ou non).

**Algorithme d'extraction de règles** : calculer les fréquences de chaque item ( $L_1$ ), supprimer de  $L_1$  les éléments plus petit que  $min_{sup}$ , on calcule toute les combinaisons 2 à 2 d'éléments, à partir de  $K = 3$  combiner seulement les éléments avec le même départ, enlever de  $C$  les éléments dont au moins un des sous-ensemble n'est pas présent dans  $L$  précédent, calculer  $L$  comme étant  $C$  sans les éléments avec le support dans  $T \leq min_{sup}$ , arrêter quand on ne fait plus de combinaisons. Prendre dans le sac tous les éléments non-traçés. **Phase 2** : Pour chaque éléments qui ne sont pas du premier niveaux, calculer  $conf(X \Rightarrow Y)/conf(X)$  (faire avec  $Y$  aussi).

**Partitionnement** : Diviser  $T$  en  $n$  partition et appliquer *apriori*() sur chacune et prendre les unions des résultats. Avantages : s'adapte à la mémoire dispo, facilement //. Défaut : on peut avoir bcq de candidat au deuxième passage.

**Decision Tree** : Accuracy rate :  $\frac{n_{pred}}{n_{true}}$ , évaluation du modèle : accuracy, speed, robustness, scalability. **Basic algorithm** : Tree is constructed in a top-down recursive divide-and-conquer manner, At start, all the training examples are at the root, Attributes are categorical, Examples are partitioned recursively based on selected attributes. **Condition d'arrêt** : no more sample, plus d'attribut à splitter, All samples for a given node belong to the same class. **Sélection d'attribut** : plus gros gain de performance :  $Gain(A) = Info(D) - Info_A(D)$ ,  $Info(D) = -\sum_{i=1}^m p_i \cdot \log_2(p_i)$ ,  $Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times I(D_j)$ .  $SplitInfo_A(D) = -\sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2(\frac{|D_j|}{|D|})$ .  $GainRatio = Gain(A) - SplitInfo(A)$ .  $gini(D) = 1 - \sum_{j=1}^n p_j^2$ ,  $p_j$  est la fréquence relative  $\frac{n_j}{n_{tot}}$ .  $gini_A(D) = \frac{D_1}{D} gini(D_1) + \frac{D_2}{D} gini(D_2)$ .  $\Delta gini(A) = gini(D) - gini_A(D)$ . **Information gain** : biaisé au travers des attributs multi-valués. **Gain ratio** : tendance à choisir des séparations mal balancée quand des partitions sont plus petite que d'autre. **Gini index** : biaisé au travers des attributs multi-valués, difficultés avec beaucoup de classe, à tendance à favoriser les tests qui donnent des partitions et des "purity" équilibrés. **Overfitting** : preprunning (ne split pas un noeud si le gain est en dessous des certains threshold), postprunning (supprime les branches qui minimise une certaine erreur).

**Clustering** : hiérarchique (analyse détaillé) ou non (gd ensembles de données). **HAC** : démarre avec 1 cluster par élément, groupe les clusters les plus similaire 2 par 2 jusqu'à atteindre  $k$  cluster voulus. **Saut minimum** : valeur maximum de la fonction de similarité. **Diamètre** : valeur minimum de la fonction de similarité. **Moyenne** : faire la moyenne de toutes les combinaisons. **K-means** : On démarre avec  $k$  centroid, lors de chaque itérations on fait : attribuer chaque point au centroid le plus proche, recalculer le centre des centroids (vecteurs moyen des distances aux centroid). Faire jusqu'à convergence ou bien un nombre d'itération fixé atteint. Distances : distance euclidienne, distance de manhatan, similarité par le cosinus  $1 - \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \cdot |\vec{y}|}$ . Complexité :  $O(iknm)$  iteration, taille vecteur, nombre cluster, nombre sample. Buckshot : faire premier et utiliser comme base pour k-mean. Qualité du clustering :  $Purity(\omega_i) = \frac{1}{n} max_j(n_{ij}), j \in C$ .

**Data warehouse** Type de schéma : **Star** une fact table relié à des dimensions table, **Snowflake** idem mais avec des dimensions table qui peuvent être normalisé. **Fact Constellation** comme star mais avec plusieurs fact table. Utilisation des schémas : datawarehouse plutôt constellation et datamart plutôt star ou snowflake. Type de warehouse : **enterprise warehouse** sujet qui concerne toute l'entreprise (données détaillées et summarisé), **datamart** un groupe spécifique d'utilisateur (département) les données sont souvent summarisé (par mois p.exemple). **Virtual warehouse** un ensemble de vue sur une base de données opérationnelle. Une **fact table** est composé de clé (vers les autres **tables de dimensions** (information sur différents objets)) et de mesure. **Star net query model** Groupe de ligne qui émanent d'un point central, chaque ligne est une dimension et chaque point de la ligne est un niveau de la dimensions. **Pourquoi DW** competitive advantage, business productivity, customer relationship management, cost reduction. **OLTP vs OLAP** label : users, function, DB design, data, usage, access, unit of work, records accessed, users, metric, DB size. **Architecture** data sources -> data storage -> OLAP engine -> Front-end tools. **Opérations sur un cube** : **roll-up** -> summarize data (ville -> pays), **drill-down** -> inverse of roll-up, **slice** -> where en SQL, **dice** -> select en SQL, **Pivot** -> reorient cube. Si chaque dimensions (4) à 5 niveaux -> donne  $5^4$  cuboïde. Facteur du nombre de niveaux par dimensions. Le cuboïde avec 0 dimensions s'appelle *apex*. **Base cuboïde** : cuboïde de base avec toutes les dimensions. **Data transformation** : convertir les données des sources vers un modèle unifié.

**Vecteur d'incidence** : vecteur binaire qui correspond à si un mot est dans chaque document ou non. OR-query  $O(x + y)$ . Si on a plusieurs OR avec des AND, il faut faire la somme des  $df$  et commencer par le plus petit.

TODO

skip pointer jaquard avec exemple tf idf score calcul cosinus similarity tf idf tableau mettre algo cosine score inverted index stemming lemming,... wild-card query precision recall ect... expliquer + formule MAP MAP score r-score f1-score non-interpolated precision shingling min-hashing locality sensitive jaccard similarity IR description term-document incidence matrix IR architecture search model tokenization normalization stemming stop words query processing AND / OR Posting list AND : tu commences au début des deux listes , tu incrémente la liste ayant le nombre le plus petit si intersection tu récupères le numéro puis tu incrémente les 2 terminé quand au bout d'une liste.