C1: Fouille de données: étude collecte, nettoyage, traitement, analyse et connaissances issus des données. Systèmes automatisés génèrent données à des fins de diagnostic ou d’analyse. Données brutes sont souvent non structurées. Pipeline fouille data :Data collection🡪DataPreprocessing(Feature\_extract|Cleaning/integration)🡪Analyctical Processing(Building block i)🡪 Output for analyst. Nettoyage : logs contiennent bcp d’info pas utilse. Extraction d’attributs : le détaillant créer enregistrement pour chaque client, ak choix attrib. Intégration des données : enregistrements combinés aux données démographiques des clients. Analyse des données : analyste décide cmt use cet ensemble de données nettoyé pr recommander. Data science vs. Computer science : informaticiens feel pas data 🡪 des chose a execute ds progra. Tester performances algo : exécuter ak données aléatoires. Ensembles de données intéressantes sont rares, faut work pr get interesting data. ***Science traditionnelle***:Hypothesis-driven :fait des questions et ensuite génère données spécifiques qui sont nécessaires pour confirmer ou réfuter l’hypothèse. ***Nouvelle science****:* Data-driven :axée génération de données, en croyant que de nouvelles découvertes seront faites aussitôt que l’on sera capable de les explorer.|C2 : Data munging : acquérir données et les préparer pour analyse.Métadonnées : source impt de data Sources de données : 1)**Données privées**(Google,Fb ex :nom, date naissance)🡪 organisations ont bases de données internes. Obtenir accès extérieur à ces données est généralement impossible (Desjardins est une exception car leur data a leak). Ont API donne acces a qln data.**2)Données gouvernemaentales**🡪Villes et pays start ouvrir leurs données. Bd --> publique préservation de l’anonymat.(remove info cle ex nom/origine) **3) Données académiques**🡪revues, recherches scientifiques par (obligatoire data public pr publications). Permettent trouver données économiques, médicales, démographiques et météorologiques. Repérez les documents pertinents et demandez aux propriétaires. Cependant, ces données très explorées. **4)Web search et web scrapping**) Scraping : l’art de récupérer (extraire) des données d’une page Web. Faut d’abord vérifier s’il y a des APIs rendues disponibles par le site et s’il y a qln a déjà codé un scraper. Conditions de service limitent ce qu’on peut légalement faire (copyrights). **5)Internet of Things** (IoT)🡪 Sources de données sont très variées. Le stockage n’est pas cher! Type de données : **1)Données indépendantes** : soit au niveau des individus (observation), soit au niveau des caractéristiques (attrib). ***Données multidimensionnelles***: jeu données *D* ensemble *n* enregistrements X1, … Xn où chaque enregistrement Xi contient *d* d’attributs ( Ex : chien contient (race, couleur), personne(grandeur, poids. Nature des attributs classifie données comme numériques, catégoriques, binaires, textuelles. **2)Données dépendantes** : présentent des relations (ex. réseau sociaux, séries temporelles) Ex: chaque élève on enregistre son poids --> caractéristiques (ex: taille et poids) dépendantes et (ex: taille et lunettes) indépendantes. Connaissance des dépendances préexistantes modifie la fouille de données. **Dépendantes vs. Indépendantes** :Données dépendantes sont + complexes en raison des relations préexistantes entre les éléments. Séries temporelles : contiennent data générées par mesures continues ds temps. Ces données ont des dépendances implicites intégrées aux valeurs observées au fil du temps. Certaines séries peuvent montrer des patrons périodiques d’attribut(qch se repete everytime). Données spatiales : Attributs non spatiaux (ex. température, pression, intensité de couleur de pixel d’image) peuvent être mesurés dans l’espace. ont des attributs spatiaux et temporels. **Portabilité** : capacité à s’adapter facilement en vue de fonctionner dans différents environnements d’exécution. Improtantcar données sont hétérogènes et peuvent contenir plusieurs types d’attributs. C’est toujours + compliqué et long de développer des algos qui puissent marcher pour des jeux de données hétérogènes. **Les outils *off the shelf*** *(license publique)* sont souvent pour un type spécifique de données. Numérique → Catégorique  1)**Discrétisation**🡪divise l’intervalle de valeurs d’un attribut numérique en φ intervalles. attrib aura 1, . . , φ valeur catégorique. Ex. attrib numérique âge : Création de 4 intervalles : [0, 20]; [20, 40]; [40, 60]; [60, 100]. val symbolique d’un enregistrement dans l’intervalle [20, 40] est 2, ds intervalle [60, 100] : 4. Variations dans intervalle perdues après la discrétisation. Data non uniformément distribuées à travers les intervalles. Normaliser la tailles des intervalles en fct distribution des données p/r distrub data.**2) Binarisation**🡪passer de x categorie a x val binaire (numériques). L’un des x attributs prend la valeur de 1, et le reste prend la valeur 0. (1000|0100) Texte → numérique : 1) **Bag-of-words** : attributs 🡪 mots (allo); valeurs 🡪 nb occurance(2). \*Représentation très creuse (≈ 105 mots anglais. Perte des relations entre mots. **2)Word embeddings** : predire mots manquants ak contexte. Use apprentissage profond. Série temporelle → numérique : 1)Transformée en ondelettes discrètes *wavelets* : convertit données de séries temporelles en données multidimensionnelles, en un ensemble de coefficients représentant les différences moyennes entre différentes parties de la série. Un sous-ensemble de plus grands coefficients peut être utilisé pour réduire la taille des données numériques. Les données résultantes sont moins dépendantes que la série temporelle originale. (8,6,2,3,..) 🡪 coeff 1 : 8-6/2 = 1. 2e coe : 2-3/2 = -0.5 N’importe quel type → Graphe : 1) Graphe de voisinage : observation = nœud graph. arête existe entre deux éléments *i* et *j* si leur distance *d(i, j)* est inférieure à un seuil. poids *w(i, j)* (arête *(i, j))* = fonction kernel appliquée sur *d(i, j)*. Ex. *heat kernel* : où t est un hyperparamètre. Nettoyage de données : **Erreurs :** informations perdues lors de leurs acquisitions (probleme ds le capteur qd on get temp). **Artefacts** : problèmes découlant du traitement de données (corrigeable car c ds la transformation mistake). Compatibilité de données : données ont besoin d’être compatibles pour faire des comparaisons (ex: metres avec pieds pr mesure => work pas). Conversion d’unités : décision système métrique incohérences potentielles : cm, m, km ? Attention au moment d’intégrer des données de bases différentes (ex : pieds vs. Mètres). Si distribution 2 valeurs, alors → problème de compatibilité. Représentation numérique : Attributs discrets : entiers always. Attributs mesurés physiquement : jamais quantifiés avec précision=> chiffres réels (double)(ex :22,3 cm). Les quantités fractionnaires doivent être décimales, pas par *(q,r)* ex :(livres, oz) ou (pieds, pouces). Représentation de caractères : UTF-8 est un codage multi octets pr tt caract Unicode, compatible avec ASCII. tj générer texte en UTF-8!! **Unification de noms** : **1) Utiliser transformations simples** pour unifier les noms(ex. minuscules). **2) Considérer les méthodes phonétiques**(défauts de l'alphabet en le remplaçant par un alphabet spécial, each signe a une valeur fixe) de hashing (ex :Soundex et Metaphone ) (ex : élimination des lettres doublées). **3) Compromis entre faux positifs et négatifs**(ex : deux personnes avec même nom🡪 faux positifs: 2 personnes pour la même personne comme Daniel, Daniel Aloise peut être compris pour Daniel Aloase). Unification de dates dans le temps : Aligner événements temporels de différents ensembles de données/systèmes peut être problématique. Use temps universel (UTC) ou le temps UNIX. Unification financière : Nécessaire de **tenir compte de la valeur de l’argent** au cours du temps (inflation) pour faire des comparaisons justes à long terme. Utiliser rendements/changements de % au lieu des changements de prix absolus pr compare variables dans temps. Normalisation : diff attrib ont différentes échelles de référence et ne peuvent pas être comparés entre eux(ex : âge et salaire). tt fct agrégée calculée sur diff enregistrements (ex : les distances euclidiennes) dominée par attrib de + grande magnitude. Considérons *µj* la moyenne de l’attribut *j* et *σj* son écart type : **standardisation ou z-scores : (moyenne = 0 ecart-type = 1)**. Prendre les valeurs (i), soustraire la moyenne et diviser l'écart type. suppose une distribution normale de moyenne 0 et d’écart type 1. Le biais optimal est 0 lorsque données centrées (mu = 0). interpreter coefficients de reg si attributs ont la même échelle. éviter les instabilités num 🡪 échelle attrib petite.Used pr treat data ak distrib aysmetrique et val aberante. **Min-max scaling** : ∈ [0, 1], sensible aux données aberrantes : 🡪 les valeurs entre 0 et 1,max n'est pas garanti(ex: age, c'est pas garanti jusqu'à quand on va vivre). Min-max scaling implique écarts + petits.Traitement d’images/ reseau neurone, données ∈ [0, 1] → min-max scaling. Valeurs manquantes : impt de bien representer data manquant. Mettre ces valeurs à 0 = mauvaise idée ! Try estimer val : 1) **Imputation(affectation) de valeurs manquantes** : souvent préférable d’estimer ou d’affecter des valeurs manquantes au lieu de les laisser vides (ou a 0)(ex. une bonne estimé pour votre année de décès est l’année de naissance + 80). **Deux facon : use moyenne u ou prendre val aleatorie**. 2) **Régression** : use régr linéaire pour prévoir les valeurs manquantes work si pas bcp de champs manquants. **Coefficients régression linéaire** : Ex :lorsque des attributs se multiplient🡪 Si données ont été normalisées,coefficients reflètent importance des attributs qu’ils multiplient par rapport à la sortie. Ils peuvent être comparés entre eux. Si données ne sont pas standardisées, alors impossible de comparer l’importance des attributs avec valeur des coefficients (ex. l’âge avec un coefficient 1 peut avoir la même importance que le salaire avec un coefficient 0.0001 si les données pas standardisées). Détection de données aberrantes : Regarder de manière critique les valeurs max et min pr tt var. Data distribuées pas avoir de grandes valeurs aberrantes. Éviter solution facile de juste la supprimer. Visuellement easy voir valeurs aberrantes (qd dim faible). Considéré comme un problème d’apprentissage non supervisé. C3 : Réduction de données : Lorsque la taille des données est + petite, il est + facile et performant d’appliquer des algos sophistiqués et complexes.visualiser et interpreter data + simple. Réduction des données peut se faire en termes du **nb de lignes (enregistrement = observ)** ou/et **nb de colonnes (attribs**). Réduction data 🡪 trjs une perte d’information. **1) Échantillonnage**: les enregistrements(nb lignes) sont échantillonnés pour créer BD bcp + petite. fraction data est sélectionnée et used pour l’analyse. **A)** ***l’échantillonnage biaisé,*** certains data sont + probablement etre retenus en raison de leur grande importance pour l’analyse(ex : certains types de données temporelles).**B)**Dans ***l’échantillonnage stratifié***, données divisées en ensemble de strates(genre cluster), puis échantillonnage est fait indépendamment à partir de chacune des strates(ex : sondage sur style de vie des individus popula ; 1 échantillon aléatoire 1 million de participants pas capture un milliardaire) **2) Sélection d’attributs**: seul un sous-ensemble des attributs est utilisé dans le processus analytique(sélectionner les colonnes). **Moyenne :** **Écart-type(variance^2)** :**Covariance :** (S, t : attri \ i : observ) **3)Réduction de dimension** : corrélations entre lignes ou colonnes sont exploitées pour représenter données dans une dimension + petite. Matrice de covariance : calculer covariance entre paire d’enregistrements (ou chaque pair d’attributs), on obtient matrice de covariance contenant chaque paire d’attributs. Centralisation par la moyenne : . Une matrice *d × d* de produits scalaires, mesurant "le synchronisme" entre les attributs 🡪 . Une matrice *n × n* de produits scalaires, mesurant "le synchronisme" entre les enregistrements (individus (observ) ou les lignes) 🡪 Sélection des attributs : Coefficient de corrélation(tenir compte de l’écart type de nos attributs) 🡪 Matrice corrélation (diag = 1. Ligne et colone -> attrib) permet évaluer slm des pairs d’attributs. On peut évaluer des triples ou des corrélations pour des sous-ensembles d’attributs + grands. Cependant, le nb de sous-ensembles est exponentiel. La plupart des méthodes de sélection évaluent les attributs indépendamment les uns des autres. Les méthodes sont différentes en fonction des données disponibles (avec ou sans étiquettes). Mat : + : val attrib i aug, autre aug aussi. = 0 : pas de relation. - : val attrib i aug, col j diminue. Filtrage : Classe chaque attribut en fonction d’une mesure univariée. Sélectionne les attributs avec les mesures les plus élevées. La mesure doit refléter le pouvoir discriminant de chaque attribut. Filtrage corrélé : Algorithme itératif 🡪 **1)** sélectionner un attribut ***s*** (selon une métrique quelconque). **2)** vérifier corrélation de ***s*** avec attributs déjà sélectionnés (par pair d’attributs). **3)** Si somme de ces corrélations dépasse un seuil, enlève ***s*** de la sélection et STOP ! Filtrage🡪 **Avantages** : Rapide à calculer. **Désavantages** : Un attribut qui n’est pas "utile" tout seul peut être très utile lorsqu’il est combiné avec d’autres. **Discriminant linéaire de Fisher** : filtrage des **combinaisons linéaires** des attributs. Emballage : Influencé par le modèle choisi (biaisé). Coûteux en temps de calcul. Si pas d’étiquettes,exécuter une méthode de *clustering* pour en avoir. Réduction de dimension : Décomposition par valeur propres 🡪 Toute matrice *A* ***carrée symétrique*** de taille *n × n* peut être décomposée de la façon suivante : où *Ui* est le vecteur propre associé à la valeur propre *λi*. Remarque :les produits les + importants sont ceux associés aux valeurs propres les plus grandes. Analyse de composantes principales(PCA) : **La matrice de covariance** est symétrique *(d × d)* et semidéfinie positive. Elle se décompose par : où tous les *λi ≥ 0, i = 1, . . , d*. Les vecteurs (propre) *Ui , i = 1, . . . , d* sont nommés composantes principales pour la PCA et sont triés en ordre croissant par rapport à *λi*. On dénote la matrice *n × d* obtenue par : Pour la **PCA**, slm *k << d* colonnes de présentent des valeurs qui varient de façon significative. On peut prouver que vecteurs propres *Ui* représentent de solutions orthogonales successives au problème de la maximisation de la variance Σv le long d’une direction unitaire *v*. Le problème réduction de dimension d’une matrice X de données peut aussi être vu comme celui de factoriser une matrice : X = B *\** C où B et C sont de taille *n × k* et *k × d*. Si k < min(n, d), B et C réduisent X. Décomposition par valeurs singulières : La **SVD** d’une matrice X de taille *n × d* la factorise comme : X = WD où D est une matrice diagonale rectangulaire *n × d*, W est de taille *n × n* et V est de taille *d × d*. Les matrices *W* et *V* sont orthogonales. Donc, *WD* pondère chaque colonne de W par D, de même pour D . Le fait de ne conserver que les lignes/colonnes avec des poids dii importants (**singular values**) nous permet de compresser X avec relativement peu de perte. Soit A et B,2 vecteurs de taille *n × 1* et *1 × m*, le **produit externe** des vecteurs donne une matrice est : P = A ⊗ B avec P[i, j] = A[i]B[j]. La matrice X peut être exprimée par la somme des produits externes de la SVD avec les termes : 🡪 En additionnant slm les + grands produits matriciels, on obtient une approximation de X. Multidimensional scaling (MDS) : projette un espace de grande dimension d’origine sur un espace de petite dimension en préservant les distances entre les paires. **input** : matrice de distances D de taille *n × n* obtenue dans l’espace original, et la dimension *k < d* de la projection. **output** : une configuration *Y1, . . . , Yn* de *n* points dans . **metric** : **MDS** minimise : Un algorithme de descente du gradient peut être utilisé pour optimiser MDS. ISOMAP : Les données font partie d’un *manifold (ex : plan)* (tubulure) non linéaire de dimension plus faible dedans un espace de plus grande dimension (on calcule la distance bleue alors qu'on veut la rouge). **ÉTAPE 1)** Pour chaque *Xi , i = 1, . . . , n*, nous trouvons ses voisins situés à l’intérieur d’une petite distance euclidienne de *Xi* . **ÉTAPE 2)** Construire un graphe avec une arête entre tous les points voisins. **ÉTAPE 3)** La distance géodésique entre deux points quelconques est ensuite approchée par le chemin le plus court entre les points dans le graphe. **ÉTAPE 4)** MDS est appliqué aux distances obtenues pour produire une représentation en dimension faible. t-distributed stochastic neighbor embedding (t-SNE) : Chaque donnée multidimensionnelle est modélisée par une autre à plus petite dimension. t-SNE préserve la structure locale en maintenant autant que possible la structure globale intacte. **Modèle probabiliste** : Enregistrements similaires (ptit distance) sont modélisés par pts voisins et des données dissemblables (grande dist) sont modélisées par des pts éloignés avec une **probaba élevée** dans espace original et dans espace réduit. Pige pt hasard pi ds espace original, 2e pt proba choisir proprot a dist entre pt 1 et le new pt. **ÉTAPE 1)** Distribution de probaba sur paires de données est construite de manière que données similaires ont forte proba d’être sélectionnées ensemble. **ÉTAPE 2)** Chercher distribution semblable pour données mappées en dimension réduite. Enfin, la divergence de Kullback-Leibler (une mesure de divergence) est minimisée entre les 2 distributions (problème d’optimisation non convexe chaque run donne result diff). Similaire a mds. Transformation spectrale : Graphe de similarité ⇒ données multidimensionnelles. Conserve structure de similarité d’un point de vue **local**. Très utile étant donné l’importance des graphes comme structure de représentation. Soit G(N, A) un graphe non dirigé 🡪 On assume que |N| = n. Une matrice symétrique W de taille *n × n* contient similarités entre noeuds liés par une arête. Cette matrice est **creuse**. On veut incorporer noeuds de ce graphe dans espace de dimension *d* afin que structure de similarité des données soit préservée. Transformation spectrale : cas simple 🡪 *n* noeuds sur un ensemble de valeurs *y1, . . . , yn* à 1D. Il n’est pas souhaitable que des noeuds connectés avec des arêtes de poids élevé (très similaires) soient mappés sur des points éloignés de cette ligne. On veut minimiser alors : *.O* peut être réécrit en termes de la **matrice Laplacienne L** de W 🡪 où *D* est une matrice diagonale telle que et L = D − W . O est minimisé avec égal au + petit vecteur propre de la relation Le + petit vecteur propre associé n’est pas informatif *(λ = 0)*. On prend le **vecteur propre associé à la deuxième + petite valeur propre**. Les autres vecteurs propres U2,U3, . . . ,Uk+1 associés aux valeurs propres *λ2 ≤ λ3 ≤ . . . ≤ λk+1* forment coordonnées de la matrice multidimensionnelle de taille *n × k*. C5 : Big Data : analyse de données massives. Les données massives se distinguent des données qui rentrent en mémoire ou qui peuvent être analysées par un seul ordinateur. Cela requiert des infrastructures à grande échelle et robustes(l'impact que toutes les séquences de défaillance peuvent avoir sur le système). Les 3 « V » - **Volume** : Infrastructures de stockage sophistiquées. (ex : ensemble des produits et des avis Amazon) Algos linéaires en temps. Les 3 « V » - **Variété** : sources de données sont très hétérogènes.(ex : posts Fb peuvent contenir texte, émojis, images, vidéo, fichier audio).Technique d’intégration *ad hoc*(tous ensemble et connectés). Les algorithmes d’analyse sont très différents selon type des données (textuelle, vidéo, audio). Les 3 « V » **Vitesse** : Systèmes **temps réel** (live) et collection de données en **continu** (always on). (ex :flots de données, requêtes Google) Infrastructures sophistiquées pour collecter, indexer, récupérer, et visualiser les données. Utilisateurs souhaitent accéder aux dernières données en temps réel. Ingénierie et technologie. **Big Data limitations** Données massives limitations : **Participation non-représentative** 🡪Données de n’importe quel réseau social ne reflètent pas idées des personnes qui ne l’utilisent pas(ex : IG les jeunes, New York Times les riches). **Données générées par des machines** 🡪 Une armée de critiques (reviewers) écrit chaque jour de faux commentaires. De nombreux agents numériques (bots) écrivent et consomment massivement des tweets et autres textes ! 90% des mails envoyés sont des pourriels (spams). Les données nous mentent peut-être! Twitter estime que 23 M de ces utilisateurs actifs sont des agents numériques. **Trop de redondance** 🡪 Majorité des données correspondent à objets déjà connus.Plupart des données que nous voyons sont des choses que nous avons déjà vues. La déduplication,c-à-d la suppression des doublons est une étape essentielle de beaucoup d’analyses. (ex : utiliser des photos prises sur internet pour identifier des bâtiments). Exploiter stockage hiérarchique : Les algos d’analyse des données massives sont souvent limités par **stockage** ou le **débit** plutôt que par puissance de calcul(Il faut 30 min pour lire 1 To depuis un HDD, et 5 minutes depuis un SSD). Infrastructure est importante ! car performance dépend + de gestion des données que de qualité des algos. Gestion des données massives : La **latence** suit une remise de volume(le 1er accès aux données est + coûteux que les accès suivants. **Organiser les calculs** pour tenir compte de cette remise en analysant les données sous la forme d’un flot (stream), en pensant « gros fichier » plutôt que « dossier », en compressant les données. Paradigmes modernes de calculs : *Ordinateurs individuels*🡪Téléphone ≈ 0.005 TFLOPS, Ordinateur central (mainframe) 143 000 TFLOPS. *Matériel spécialisé*🡪Concentre sur sous-ensemble d’opérations, les GPUs ex : NVIDIA A100 : 19 TFLOPS (FP32), 156 TFLOPS (TF32 Tensor Core), 312 TFLOPS (TF32 Tensor Core + Sparsity). *Système distribué*🡪de nombreux ordinateurs « peu puissants » qui travaillent ensemble. Peut atteindre 100 000 TFLOPS. Calcul distribué : UTILES pour données massives et pour accélérer calculs. Calculs + rapides en décomposant problèmes en pls sous-problèmes identiques.Diviser problème en sous-problème + simple à résoudre : **1)** Ordinateurs (peu puissant) résolvent simultanément 1 sous-problème chacun. **2)** Combiner les sln des sous-problèmes pour résoudre le prob initial. Ex :compter le nb de parcmètres à Montréal 🡪 **A)** **Approche centralisée (1 ordinateur)** : 1 marathonien parcourt toute la ville et compte parcmètres. 1 système de comptage automatique à partir d’images satellites. **B)Approche distribuée (beaucoup d’ordinateurs)** :1,000 personnes parcourent une petite zone géographique et comptent les parcmètres. Une fois terminé, chacun envoie son rapport au QG. Parallélisme de donnée : Meilleure façon d’exploiter le parallélisme pour traiter données massives. **Problème** :données trop massives. Envoyer données sur réseau prend du temps. **Solution** : rapprocher calculs des données. Traiter données séquentiellement car recherches sont coûteuses. Stocker données plusieurs fois pour augmenter fiabilité. Algo Big Data classique :**1)** Itérer sur grand nombre d’enregistrements. (Work1, Work2, Work3) **2)** Extraire de chaque enregistrement quelque chose d’intérêt(worker).**3)** Mélanger et trier résultats intermédiaires(r1, r2, r3).**4)** Agréger les résultats intermédiaires.**5)** Générer sortie finale(result). Difficultés de la parallélisation : Cmt assigner les sous-problèmes aux workers. Que faire si + de sous-problèmes que workers. Cmt savoir si tous workers ont fini. SLN 🡪 **MapReduce**stocker + traiter big data) : framework permettant distribuer un problème en divisant les calculs en sous-problèmes. Modèle de programmation parallèle simple, mise à l’échelle à des milliers d’ordinateurs simples, tolérance aux panes grâce à de la redondance. Composants de Hadoop : **1)MapReduce**🡪traitement données massives de manière parallèle/distribuée (fault-tolerant, scheduler, execution). **2) HDFS**🡪système de fichiers distribué (fault-tolerant, high-bandwidth, high-availability). Map et Reduce : **Map** : fonction exécutée par chaque ordi pour résoudre sous-problème. **Reduce** : fonction combinant solutions de chaque sous-problème en la solution finale. Il y a + de sous-problèmes que de machines dispo. Temps linéairement proportionnel au nb de machines. Si une machine crash, faut recalculer le sous-problème associé. Données sont lues depuis disque au début et écrites à la fin. Services de cloud computing : Plateformes comme Amazon AWS, Google Cloud, Microsoft Azure rendent facile location d’un grand nb de machines pendant une période courte. Coût difficile à évaluer, car pls facteurs: processeurs, cartes graphiques, mémoire vive, stockage à long terme et bande passante. Possible de réduire les coûts pr certains usages : *spot instances* permettent payer uniquement le temps où instances sont utilisées. ***Reserved instances*** permettent réserver des instances pour longue période. **Éthique et société Big Data** : **Transparence et propriété des données**(si organisation suit bonnes pratiques de stockage et d’utilisation des données, si les erreurs peuvent se propager et mécanismes de correction). **Biais des modèles**(algos de machine learning héritent des biais des données d’apprentissage, ex : fil actual qui renforce la polarisation politique, search engine montre meilleures opportunités de job aux hommes qu’aux femmes). **Préserver sécurité données massives**(encrypter et supprimer données pour éviter failles de sécurité🡪adresses, id, et mdp divulgués persistent des années). Préserver anonymat dans données agrégées(utilisateurs ne doivent pas être identifiables mm si les noms, adresses, et id sont supprimés).C5 : Big Data – SQL : Un **attribut** est un id (un nom) décrivant une info stockée dans une base(ex :âge d’une personne). Le **domaine** d’un attribut est l’ensemble, fini ou infini, de ses valeurs possibles(ex : l’attribut numéro de sécurité sociale a pour domaine l’ensemble des combinaisons de 15 chiffres). Une **relation** est un sous-ensemble du produit cartésien de *n* domaines d’attributs et est représentée sous la forme d’un tableau à 2D dans lequel les *n* attributs correspondent aux titres des *n* colonnes. Une **occurrence**, ou n-uplets (**tuples**), est un élément de l’ensemble figuré par une relation. Une occurrence est une ligne du tableau qui représente la relation. **Schéma de relation** précise nom de relation ainsi que liste des attributs avec leurs domaines. Relation stockée avec service de fichiers répartis : Une relation de schéma Lien(url1 : Chaîne, url2 : Chaîne) décrivant la structure du web contient ≈ 109 occurrences. Cette relation peut être stockée comme un fichier dans un service de fichiers répartis. La relation est partitionnée en plein de petits fichiers qui sont stockés sur plusieurs nœuds. **Algèbre relationnelle-Sélection** ex🡪 selection (age > 20). Génère relation regroupant exclusivement toutes les occurrences de la relation qui satisfont la condition. **Map** : pour chaque ligne *r* dans la relation, *retourne (r,r)* si et seulement si la condition est satisfaite. Autrement, la fonction ne retourne rien. **Reduce** : retourne la clé reçue en entrée. De manière informelle, la fonction « ne fait rien ». **Algèbre relationnelle – Projection** ex🡪 projection(Name, isActive). Supprime attributs d’une relation qui ne sont pas sélectionnés et élimine occurrences en double apparaissant dans nouvelle relation. **Map** : pour chaque ligne *r* dans la relation, *retourne (r ′ ,r ′ )* tel que *r ′* ne contienne que les attributs sélectionnés. Après suppression des attributs, il est possible que sortie contienne des occurrences en double. **Reduce** : reçoit en entrée *(r ′ , [r ′ ,r ′ ,r ′ , . . .])* et retourne une seule paire *(r ′ ,r ′ )*, ce qui supprime les doublons. **Algèbre relationnelle – Union** : opération portant sur 2 relations ayant le même schéma et construisant une 3e relation constituée des occurrences appartenant à chacune des deux relations sans doublon. **Map** : pour chaque ligne *r*, retourne *(r,r)*. **Reduce** : reçoit en entrée *(r ′ , [r ′ ,r ′ ,r ′ , . . .])* et retourne une seule paire *(r ′ ,r ′ ),* ce qui supprime les doublons. **Algèbre relationnelle – Intersection** : opération portant sur 2 relations ayant le même schéma et construisant une 3ième relation dont les occurrences sont constituées de ceux appartenant aux deux relations. **Map** : pour chaque ligne *r*, *retourne (r,r)*. **Reduce** : chaque clé peut être associée à 1 ou 2 valeurs (car nous supposons qu’il n’y a pas de doublons dans relations). Dans le cas où il y a 2 valeurs, la fonction *retourne (r,r)*, autrement rien n’est retourné. **Algèbre relationnelle - Jointure naturelle** : EXEMPLE : **P = M \* N**  La jointure retourne (i, j, k, ,). **Grouper** selon les attributs *i* et *k*, puis **écraser** en faisant somme des produits . **Map** retourne l’ensemble des données pour calculer chaque élément de *P* . Chaque élément de M et N contribue à plusieurs éléments de *P* et doit être retourné plusieurs fois. Chaque clé *(i, k)* est associée à une liste contenant les tuples *(“M”, j, mij )* et *(“N”, j, njk)* pour toutes les valeurs de *j*. **Reduce** doit connecter les 2 valeurs et qui ont le même j, pour chaque j. Ces valeurs sont multipliées, et l’ensemble des produits est sommé pour obtenir . **Jointure naturelle** : opération portant sur 2 relations contruisant une 3e relation **regroupant tt les possibilités de combinaison** des occurrences des relations qui satisfont un test d’égalité entre les attributs qui portent le même nom dans les relations. Ces attributs ne sont pas dupliqués, mais fusionnés en une seule colonne par couple d’attributs. **MapReduce** : Si 2 valeurs avec la même clé proviennent de relations différentes, alors faut créer une occurrence contenant ces 2 valeurs. La jointure peut faire exploser le nb d’occurrences puisqu’il est possible de créer chaque combinaison possible entre les occurrences des deux relations. **Map** : pour les 2 relations *Table1(A, B)* et *Table2(B, C)*, la fonction Map retourne *(b,[T1, a])* ou *(b,[T2, c])* selon la relation d’origine. **Reduce** : Pour chaque clé *b*, la fonction construit toutes les paires possibles contenant une valeur de chaque relation. La fonction retourne toutes les combinaisons sous la forme *(b, [a, c])* représentant une occurrence *(a, b, c)* dans la nouvelle relation. Si une clé ne contient que des valeurs de T1 ou T2, alors la fonction Reduce ne retourne rien. **Algèbre relationnelle - Regrouper et agréger** : regroupe occurrences selon un ensemble d’attributs et effectue une opération d’agrégation (**sum, count, max, min**) pour chaque groupe sur un autre attribut. \*Supposons que nous regroupons les occurrences selon l’attribut A et que nous agrégeons selon l’attribut B. **Map** : Pour chaque ligne (a,b,c) dans la relation, retourne (a,b). **Reduce** : Chaque clé ***a*** représente un groupe. La fonction applique l’opération d’agrégation (**sum, count, max, min**) sur les valeurs et retourne le résultat. C5-suite : **MapReduce - Produit Matriciel** : **Cas simple** : Le vecteur *v* est stocké entièrement dans la mémoire de chaque worker. Supposons que les données de M et v soient stockées sous la forme *(i, j, mij)* et *(j, vj)*. **Map** retourne (i, mij × vj). **Reduce** retourne la somme de toutes les valeurs associées à la clé i. **Map(Tuple (i, j, mij))** // Suppose que v soit stocké en mémoire emit(i, mij × vj). **Reduce(Int i, List products) -- xi ← sum(products) -- emit(i, xi)**. Cas réaliste : Le vecteur v ne peut pas être stocké entièrement dans mémoire de chaque worker. Il suffit lire le vecteur v depuis disque chaque fois que nécessaire 🡪 cela entraîne très grand nb de communications et une **baisse des performances**! SLN : découper M en stripes verticales de taille égale, et v en un nombre identique de stripes horizontales. La i-ème stripe de M ne multiplie que la i-ème stripe v. Chaque Map reçoit un stripe de M et de v. **Map et Reduce** fonctionnent de la même manière que dans cas simple.1) **Jointure naturelle** : Faire la jointure naturelle de M et N, c-à-d selon l’attribut j. La jointure retourne (i, j, k, ,). Grouper selon les attributs i et k, puis agréger en faisant la somme des produits mij × njk. **Pour la matrice M** : **Map(Tuple (i, j, mij)) --emit(j, (“M”, i, mij))**. **Pour matrice N** : **Map(Tuple (j, k, njk )) -- emit(j, (“N”, k, njk ))**. **Reduce(Int j, List values) -- for each (“M”, i, mij) and (“N”, k, njk ) in values do -- emit((i, k), mij × njk ) -- end. // Pour chaque combinaison de (i, k)**.**2) Groupe et agrège** : **Map(Tuple (i, k), Int pik )** -- // Fonction identité -- emit((i, k), pik ). **Reduce(Tuple (i, k), List produits) -- pik ← sum(produits) -- emit((i, k), pik )**. **Solution en une étape** : **Map** retourne l’ensemble des données nécessaires pour calculer chaque élément de **P**. Chaque élément de **M** et **N** contribue à plusieurs éléments de **P** et doit donc être retourné plusieurs fois. Chaque clé *(i, k)* est associée à une liste contenant les tuples *(“M”, j, mij)* et *(“N”, j, njk )* pour toutes les valeurs de *j* possible. **Reduce** doit connecter les 2 valeurs *mij* et *njk* qui ont le même *j*, pour chaque *j*. Ces valeurs sont ensuite multipliées, et l’ensemble des produits est sommé pour obtenir *pik* . **matrice M** : **Map(Tuple (i, j, mij)) -- // Pour chaque colonne de N -- for k = 0, 1, ...,r do -- emit((i, k), (“M”, j, mij))-- end**. **matrice N** : **Map(Tuple (j, k, njk )) -- // Pour chaque ligne de M -- for i = 0, 1, ..., p do -- emit((i, k), (“N”, j, njk )) -- end**. **Reduce(Tuple (i, k), List values) -- m ← dict() -- n ← dict() -- for matrix, j, value do -- if matrix is equal to “M” then -- m[j] ← value -- end -- if matrix is equal to “N” then -- n[j] ← value -- end -- end -- pik ← 0 -- for all j do -- pik ← pik + m[j] × n[j] -- end -- emit((i, k), pik )**.  **5 a 7 étapes prétraitement des données**: 1) **uniformiser dates** afin qu’elle soit au même format que autres valeurs. 2) **Convertir date (chaîne de caractères) en temps UTC**/UNIX interprétable par la régression linéaire. 3) Ajuster montant prêt et salaires en fonction de la date du prêt afin **tenir compte de l’inflation**. 4) **Affecter valeur manquante**, ex :valeur « vide » par la valeur majoritaire « BAC », car régression linéaire ne supporte pas valeurs manquantes. 5) **Normaliser attributs numériques** (âge, salaire, date du prêt, montant du prêt) afin d’éviter instabilités numériques et de pouvoir comparer les poids de la régression. 6) **Mettre en majuscule toutes les valeurs** de diplômes afin d’éviter que « PhD » et « PHD » soient considérés comme valeurs différentes. 7) **Binariser attribut diplôme**, **car régression linéaire n’accepte pas chaînes caractères en entrée, mais slm les valeurs numériques.** 8)**Identifier attributs hautement liés et les éliminer**, car il ne contribuent pas à la prédiction.

Concept Reduction dim ds PCA ?: reduct dim c processus de reduire nb dim ds un ensemble data tt en conservant majeure partie des info impt. PCA fait ca en transformant attrib origine en new ensemble attrib non correler (called composante princp). Composante princp (vect propre) ordonnees en fct de quantite de variance quelle explique ds le data. Objectif standarisation es attrib avant apply pca ? standarisation attribs avant apply pca important pr make sure tt les attribs sont sur echelle similaire. PCA sensible echelle attrib, ca aide a eviter que attrib ak echelle + grande dominent analyse. Soustraire moyenne et mettre a echelle pr avoir variance unitaire, chaque attrib contribue manière egale. Apres pca on a valeur propre, cmt interprer ? val propre obtenue reprensente quantite de variance expliquee par chacune composantes princp. Val propre + grande = composant princp corresp capture + info ou variation ds ensemble data. Val propre tirer par ordre decroissant. 1e val aka plus grand val propre -> composante + variance. Taux de variance? Mesure proportion variance explique pr chaque compos princp p/r variance totale de tt data. Aide eval import contributuion chaque composante a var globale. 3 1st composante princp explique 80% var totale ds ensemble data. aka 3 composantes capture partie significative info presente ds donnee origine. Les autres contribue moins. Alors. On peut representer ds un espace reduit avec c 3 compo princp sans trop perdre dinfo.| Articles: Factors Involved in Estimating ML Carbon Emissions Training neural networks involves complex architectures, numerous connections, and parallel calculations, requiring parameter tuning and experimentation. While a single training procedure may not emit significant carbon, considering all the experiments conducted, the cumulative emissions can be substantial, making it essential to consider factors affecting emissions. Type of Energy UsedThe type of energy source connected to a power outlet determines the carbon emissions associated with device usage. Renewable energy sources (e.g., wind, solar, hydro) are environmentally friendly, while nonrenewable sources (e.g., coal) contribute to emissions. When training machine learning models on the cloud, selecting the server's location allows for choosing the energy source, impacting the estimated carbon emissions based on the local energy grid.Cloud Providers Choosing a cloud provider is crucial as their server locations determine the emissions factors associated with different energy grids. Carbon emissions can vary significantly depending on the grid's energy mix, with variations of up to a factor of forty. Some cloud providers are carbon neutral through carbon offsetting and renewable energy credits, while efforts toward transparency and accountability aid in selecting environmentally responsible providers Hardware and Training Time The increasing complexity of machine learning tasks, coupled with the need for more data and powerful hardware, leads to longer training times and rising carbon emissions, making it challenging for smaller entities to afford computational resources. Additionally, factors such as architecture search and hyperparameter tuning contribute significantly to emissions, highlighting the potential for reusing pre-trained models and training them for specific tasks to mitigate carbon impact.Action Items1/Choose environmentally conscious cloud providers by reviewing their sustainability efforts and comparing their environmental footprints.2/Select data center locations with lower carbon intensity when requesting cloud servers, considering emissions factor documentation for energy grids worldwide. 3/Minimize wasted resources by conducting a literature review, utilizing pretrained models, and employing random search instead of grid search for more efficient experimentation.4/Opt for efficient hardware, such as GPUs and TPUs designed for parallel computations, to improve the efficiency and reduce energy usage during ML model training.5/Utilize an ML emissions calculator to estimate the carbon emissions associated with specific training details, such as server region, GPU type, and training time.6/Promote transparency and awareness by disclosing the infrastructure used and the emissions generated in ML research publications to address the environmental footprint of machine learning. Carbon Emissions and Large Neural Network Training the rapid growth in computation demand for machine learning (ML) and the challenges of estimating energy costs without detailed information. The authors calculate energy use and carbon footprint for several large ML models and suggest opportunities for improving energy efficiency. They emphasize the importance of location in ML workload scheduling and the significant impact of choosing specific datacenter infrastructure. The authors advocate for transparency in reporting energy consumption and CO2e in ML research and propose including energy usage as a key metric in evaluating models. They are collaborating with MLPerf developers to incorporate energy usage in industry benchmarks. How to stop data centres from gobbling up the world’s electricity Data centers, the vast treasuries of information traders, are transforming small towns like Prineville, Oregon, where gigantic data centers are being built. These facilities, larger than aircraft carriers, house thousands of circuit boards in endless rows, creating an environmental impact due to their massive energy consumption. Currently, data centers consume an estimated 200 TWh annually, surpassing the energy consumption of some countries and contributing to global carbon emissions. The entire ICT ecosystem, encompassing personal digital devices, mobile-phone networks, and televisions, accounts for over 2% of global emissions, equivalent to the aviation industry's emissions. With the demand for internet and mobile traffic skyrocketing, experts predict that by the time a child born today reaches her teens, ICT's electricity use could exceed 20% of the global total, with data centers responsible for a significant portion. While efficiency gains and innovations in cooling systems have helped mitigate the energy impact for now, future projections remain uncertain. Scientists and engineers are exploring solutions like streamlining computing processes, adopting renewable energy sources, improving cooling methods, and recycling waste heat to manage ICT's energy consumption. However, the challenge lies in staying ahead of the increasing energy demands as ICT continues to expand and evolve. C4 utiliser régression linéaire en convertissant classes en chiffres(ex : homme 0, femme 1). 0/1 fonctionne pour classifieurs binaires. Positif 🡪1, négatif 🡪0. Trjs donner valeur 1 à la variable d’intérêt. Droite régression coupe ces classes, même s’il existe séparateur dû à la minimisation de l’erreur carrée. **Régression logistique**  1ere méthode pr trouver fonction de séparation de 2 classes. Vise à convertir valeur continue en valeur de probabilité ∈ [0, 1]. Utilise fonction d’erreur différente de celle de régression linéaire🡪l’entropie croisée. 1)Fonction sigmoïde : donne la proba de *Xi* d’appartenir à une classe particulière.Proba d’appartenir à la classe 1. F(0) = ½, F()=1, F()=0. quand juste 1 attribut. Si ++ attrib, remplacer x par (une proba appartenir classes). Fct erreur : entropie croisée. Une fct convexe (1 min local= min global). Permet find meilleur separateur entre 2 classes ak methode gradient. Erreur = 0 ssi classes lineairement separable (Si donnees peut etre split pr plan, then find plan ak reg logistique). Faut ajouter colonnes non linéaires dans X pour séparer classes non linéairement séparables(). *y* est binaire. 1 seul des termes de la somme est actif pr chaque enregistrement. Classification-Problèmes : 1) Ensemble d’entraînement déséquilibré : (ex :identification d’un terroriste) Considérons droite séparation optimale pour classes très déséquilibrées, 1 positif(etre terroriste) vs 1,000,000 négatifs(pas etre terroriste). Meilleure droite trouvée par régress logistique essaiera d’être très loin du grand groupe des gens non terroristes au lieu de se placer entre les classes. Présence des faux négatifs! Mm classer tt le monde comme non terroriste aura peu d’impact pour l’entropie croisée. Solution🡪utiliser le même nb d’exemples + et - . Travailler + fort pour trouver des membres de la classe minoritaire. Supprimer des éléments de la classe majoritaire(problème, on perd des données qui peuvent etre utiles pr notre prédiction). Peser + lourdement(add poids) données de la classe minoritaire, mais surapprentissage. Répliquer membres de la classe minoritaire, idéalement avec une perturbation aléatoire[clones ak des attrib pris aleatoire](ex : on a 10 terroristes pr 10 millions de non terroristes).2) Classification multi-classe : Tâches de classification ne sont pas trjs binaires (+ 2 classes). Un film est-il une comédie, drame,sci-fi, documentaire? Sélectionner classe de proba la + élevée comme étiquette prédite (predire si appartient 1 classes ou autre et on fait ca pr tt les classes). Classific multi-classe devient + difficile à mesure que le nb de classes augmente(prédire si chat, chien, plante et prédire leurs sous-classes race, couleur). 3)Fonctions de partition: normaliser valeurs de sorties pr obtenir des probas. Classifieurs binaires indépendants ne produisent pas vraies probas (somme ne vaut pas 1). Régression multinomiale combine les classifieurs au niveau de l’entraînement (add tt les proba(prediction)). Apprentissage machine : regression linéaire et logistique + les algos qui apprenent à partir d’exemples. TT les algos qui résoudent des problemes complexes. Limitations de la régression : Besoin attributs non linéaires (ex. x1 x2) pour que régression puisse bien séparer données non linéairement séparables. Implique bcp de combinaisons d’attributs à tester. Apprentissage profond(Deep learning) : Couches cachées créent compositions de fonctions non linéaires ⇒ + grand pouvoir de représentation. Faire apprendre le réseau signifie définir les valeurs des coefficients *w*. + il y a de connexions entre neurones, + il y a des paramètres à apprendre. Analyser l’ensemble de données d’entraînement étiquetées et ajuster coefficients pour que nœuds de sortie génèrent qqc proche de *yi* lorsque réseau est alimenté avec *X*i . Pour de nombreuses tâches, la complexité des réseaux n’est pas nécessaire. Réseaux de neurones fonctionnet par surapprentissage, trouvant moyen d’utiliser millions d’exemples pour s’adapter à des millions de paramètres. **Surapprentissage** : Faible erreur de prédiction. Variance élevée (add new data 🡪 bcp changer la sln (separateur)). Uniquement lié à la compléxité, flexibilité du modèle. Modèle trop complexe, si nouvelles données arrivent, modèle va faire des erreurs(faible biais). Fonction apprise doit avoir une bonne performance de généralisation. Cela est vérifié sur **un ensemble de test**. Trop flexible, il va avoir une fonction compliqué qui va séparer les donnees et qui va etre specifique à nos donnees (apprendre par coeur) faible. **Sous-apprentissage** :Erreur de prédiction élevé (biais elever), variance faible(add qln pt ds donnes va apeine changer la prediction). Le modèle est pas assez complexe pr resoudre probleme, pas capable d’apprendre. Pas assez complexe, hypothese linearite. Apprentissage profond : évite le pire comportement du surapprentissage en utilisant des moyens - précis **d’encoder** la connaissance(arreter apprentissage un peu d’avance pr eviter le surapprentissage quand modele devient complexe). Adapté aux domaines avec d’énormes quantités de données étiquetées. Facilitent construction de modèles d’apprentissage profond. Chaque neurone *j* du réseau calcule une fonction non linéaire φ(vj) où v est donné par : où : tij est l’input au neurone j avenant du neurone i et wij est le poids donné à cet input. La neurone *j* est activé en fonction de la valeur φ(vj) et de la fonction d’activation du neurone. Backpropagation : la sortie depend de la couche d’avant qui, dont la couche d’avant depend de la couche d’avant elle-mm. Réseaux de neurones entraînés par un algo de descente gradient. Changements pr chaque exemple d’entraînement sont ramenés à des niveaux inférieurs. Fonctions d’activation non linéaires aboutissent à une fonction d’erreur non convexe, mais son optimisation produit de bons résultats. *k* plus proches voisins : 1)Trouver les k enregistrements de X qui sont les + proches de la nouvelle donnée Z que l’on veut classifier.2) Faire voter chacune de ces enregistrements selon leurs classes associées y. 3) Retourner la classe majoritaire. Succès de l’algo dépend de 2 facteurs : quantité de données d’entraînement et qualité de la mesure de distance (2 enregistrements similaires doivent faire partie de la mm classe). Grand pouvoir de représentation🡪 peut classifier des données non linéairement séparables. performance liée à notre capacité de stockage. Bagging : choisit des sous-ensembles de données choisies au hasard pour entraîner chaque arbre. Diminue le surapprentissage. Forêt d’arbres de décision : Extension de la technique de bagging. Effectue aussi la sélection aléatoire des features plutôt que d’utiliser toutes les features pour construire les arbres. Diminue la corrélation entre les arbres. Un feature très dominant oblige chaque arbre de décision à le choisir pour les premiers splits, ce qui fait que tous les arbres se comportent de façon similaire. La forêt d’arbres est composée par des arbres de décision non corrélés. Évaluation d’un modèle de classification : Étapes d’une évaluation : Paramètres🡪 ce que la fonction va apprendre. Hyperparametres🡪valeurs choisies au départ et non apprises. 1)Donnés séparés en 3 sous-ensembles : entrainement(70%), validation(15%) et test(15%). 2)On fait une liste de valeurs des hyper-paramètres à essayer.3) Pr chaque élément de cette liste, l’algo est exécuté sur l’ensemble d’entraînement et sa performance est mesurée sur l’ensemble de validation. Avec les meilleurs valeurs des hyper-paramètres identifiés avec l’ensemble de validation, on calcule la performance de l’algo sur l’ensemble de test. Procédure d’évaluation d’un algo de classification : quand pas assez de données pr séparer en 3 sous-ensembles. La validation croisée partitionne les données en *k* blocs de taille égale. On entraîne *k* modèles distincts. Le modèle *i* est entraîné sur l’union de touts les blocs sauf 1 (le bloc d’index i), et validé sur le bloc *i*. Bagging combine tout les *k* modèles entraînés dans un seul modèle de classification. Accuracy : le rapport entre les prédictions correctes et les prédictions totales. Un classificateur randomisé aurait une espérance d’accuracy de 50%. Si on prend juste la classe + nombreuse comme prédition (classificateur gourmand) accuracy ≥ 50%. Précision : Quand |P| << |N|, mesurer l’accuracy est inutile(si on a des classes qui sont sur-représenté). L’accuracy d’un classificateur gourmand serait de 95% lorsque Donc, ex :dataset de 10 TP et de 10 FP donne 10 / (10+0) = 1, mais pas trjs très précis! Atteindre valeur de precision élévée est impossible pr le classificateur randomisé ou le gourmand. Recall : mesure la capacité d’avoir raison uniquement sur les instances positives. Être + tolérants aux faux positifs (ex : erreurs où nous effrayons une personne en bonne santé avec un mauvais diagnostic) que les faux négatifs. Attention : dire que tout le monde a un cancer donne un recall parfait! Ceci donne un bon recall, mais une mauvaise précision! F-score : Mesure équilibré de score unique Moyenne harmonique est trjs < ou = à la moyenne arithmétique, ce qui rend difficile l’obtention d’un F-score élevé. If Precision > recall. Ca fait quoi ? Alors, classificateur classifie peu les donnees ds la classes positives. Cmt ameliorer ? augmenter le poids de erreur sur entrainement des data classe positives. Recall > precision ca afit quoi ? classificateur trop agressif pr classes data ds classes positive. Cmt ameliorer? Augmenter seul classification de classes pos. Receiver-Operator Curves (ROC) : Faire varier le seuil d’un modèle modifie le *recall* et la *precision*. La surface sous une ROC est une bonne mesure d’évaluation d’un modèle. Si seuil = 1, aucune instance +. Si seuil = 0, ca veut dire je prédis que tout est + et aucune instance est -.{check ak proba pr savoir si c FP ou TP. ak seuil} Évaluation de systèmes multiclasses : classification devient + difficile avec + de classes(ex : reconnaissance de chiffres). On note C[i, j] le nombre d’objets de la classe i classés comme de la classe j. Precision*i* est la fraction de tous les objets déclarés de la classe i qui étaient en fait de la classe i 🡪 🡪 instance de la classe i correctement prédite de la classe i / instances de la classe j. Recall*i* est la fraction de tous les membres de la classe i qui ont été correctement identifiés comme tel 🡪 🡪correctement prédit classe i / instances de la classe i. C[i,j] 🡪 i est l’objet de la classe i, j est le prédit commun appartenant à la classe j. Un taux de classification trop bas est décourageant et trompeur avec pls classes. Le taux de réussite du top-K donne du crédit si la bonne étiquette aurait été l’une des premières suppositions. Il est important de choisir K afin que de réelles améliorations puissent être reconnues. Si K = au nb de classes, le taux de réussite est 100%. Si K = 1 on a simplement l’accuracy(la précision et le rappel). On choisit K t.q la performance du classificateur est supérieure à celle d’un classificateur aléatoire. Choisir K pas trop élevé et pas trop petit! Ensemble d’arbres de décision : Combine ++ arbres de décision pour produire une meilleure performance prédictive. Un groupe de modèles faibles se réunit pour former un modèle fort. Vote majoritaire entre plusieurs classificateurs augmente la robustesse et permet de quantifier notre niveau de confiance. Arbres de décision : Méthode très populaire pour l’apprentissage supervisé. Facile à utiliser et rapide. Peut aussi souffrir d’un surapprentissage. Mieux vaut s’arrêter lorsque gain d’information est faible, au lieu de 0. Entropie: Meilleur attribut : Le gain (G) est la différence entre l’entropie initiale et l’espérance de l’entropie après que le feature A est choisi pour la séparation. C**ritère glouton** dans algo de construction d’une arbre de décision va choisir le feature qui maximise le gain. Question Avantage apprenti profond p/r autre methode apprent. Automatique? 1)Grande flexibilite terme params. 2)Grand nb libraries et aide tech dispo. 3) mise echelle plus naturelle. C quoi Surapprentissage ? modele suffisament flexible ou complexe, capable predire parfaitement les data ensemble apprentissage. Mais peut pas generaliser. Donc de bien predire la sortie de new data. Surapprentissage caractirise par biais faible et variance haute. C lies a la complexite du modele et nn au temps entrainement Ou qualite data. Cmt detecter? Sous ensemble eval ou la validation croise permet de detect surapprenti. Performance ensemble entrainement much better que ensemb Eval. Cmt reduire ? lower complexite modele. Why compromis entre precision et rappel ? par def. Si modele fait F = FN + FP prediction faux. Precision et rappel sont egal. FP = FN. Si FP augmente (precision deminue car denomi augmente) alors FN diminue (rappel augmente car dennomi diminue). Si rappel = 1 modele predit juste positif et presision est basse. Cmt augmenter precision ? - augmenter seuil de detection pr que individus ak grande valeur de sortie soit classifier ds la classes positive (so diminuer lke nb de FP). – faux positif sont exemple negatif classe comme appartenant classe positive, faut mettre poids supperieur a 1 sur classe nega pr reduire fp. – add individu classe neagtive ou enlever indiv classe pos. Modele reg logistique classifie courriel 2 classe. Est un spam (classe negative) si proba (val returned pr fct sigmoide) > t = 0.5. si on use autre val de seuil. Il se passe quoi au rappel? Augmenter le seuil va causer une diminution ou maintient du nb de TP et causer augmentation ou maintient du nb de FN. Rappel stay cste ou diminue. 2 biais ou limitation du big data? 1) underepresentative participation : data from any particular social media do not reflect the view of people who don’t use it. 2) machine generated content : bots write tweets and long txt in volume and even consime it. 3) redundancy : much of the data we see is something we allready seen. C quoi map reduce ? framework qui permet de distribuer un prob en divisant les calculs en sous problemes. Compioser de 2 partie. 1) map : fct executer par chaque ordi pr resoudre souis prob (usually easy). 2) reduce : fct qui combine les sln de chaque sous- prob en la sln finale.