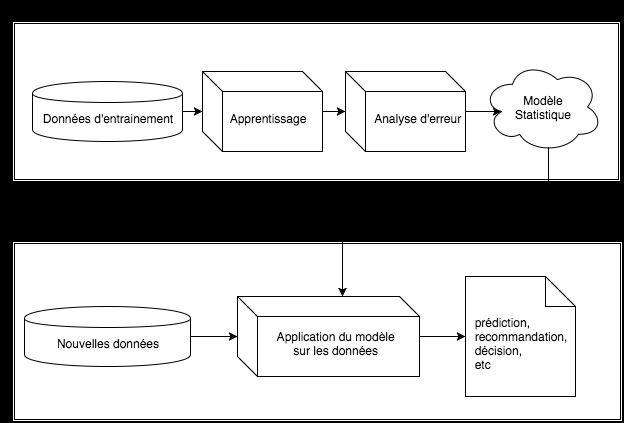
Le quartet d'Anscombe pour démontrer l'importance de l'exploration graphique avant d'analyser un ensemble de données.

C’est donc un jeu d’allers-retours entre modélisation et évaluation qui s’effectue pour obtenir les performances les plus satisfaisantes possibles

Nous allons donc être responsable dans une première phase du choix et de l'entraînement de l'algorithme d'apprentissage du modèle.



Pour accomplir la tâche spécifique poursuivie (prédiction, recommandation, décision...).

Le deep learning regroupe les algorithmes et modèles assez gros et complexes pour pouvoir traiter les données brutes directement, sans pré-traitement.

On voit donc qu'il faut bien réfléchir à une métrique plus pertinente pour être sûr de mesurer correctement la qualité de mon algorithme

Et forcément, parce qu’on fait une approximation, on a une perte d’information qui est un bruit non modélisé, qu’on estime indépendant.

Le modèle sous-jacent (statistique) représente une contrainte de “forme” non dépendante des données.

S’il y en a beaucoup, on travaille d'abord uniquement avec un échantillon représentatif de la population pour pouvoir aller plus vite.

On sépare dès le départ en deux parties notre jeu de données : un training set pour créer le modèle et un testing set pour tester la qualité du modèle.

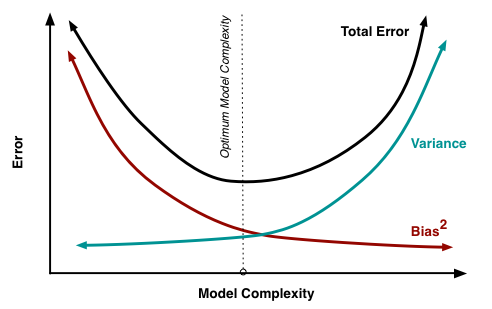
Ce qu'il faut également retenir de cette règle, c'est que vous allez construire un modèle efficace basé sur vos hypothèses de départ. Il faut donc que ces hypothèses servent votre problématique.

En machine learning, on retrouve en majorité des problèmes de classe EXPTIME, i.e. la classe des problèmes décidés en temps exponentiel par une machine déterministe.

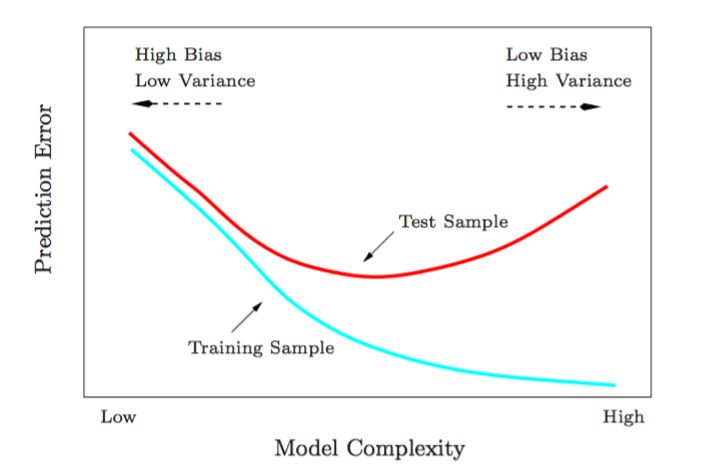
Le principe de compromis entre biais et variance est une des problématiques à laquelle vous serez tout le temps confronté lors de votre travail quotidien !

En utilisant un modèle comportant une trop grande complexité, dit "à haute variance", on peut mal capturer le phénomène sous-jacent et devenir trop dépendant aux données d'entraînement et aux petites fluctuations aléatoires, non représentatives du phénomène.

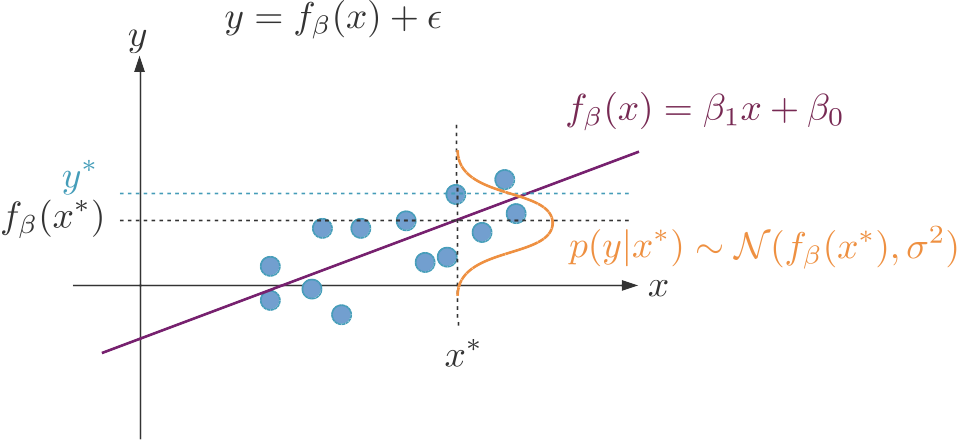
A contrario, il ne faut pas choisir un modèle trop "simple" qui biaise le résultat et ne parvient pas à capturer toute la complexité du phénomène.



En réalité, ce graphique n'est pas exact !



En fait si on augmente encore le nombre de features, il devient de plus en plus difficile d'avoir assez de données d'entraînement aux alentours pour pouvoir effectuer une prédiction correcte.

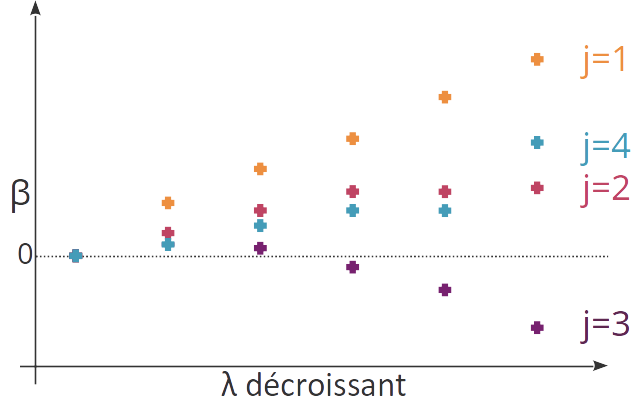


(Moindre de carrées)

Par ailleurs, si les variables sont corrélées, la matrice X ne sera pas de rang colonne plein et donc X⊤X ne sera pas inversible, ce qui veut dire qu'il n'y aura pas de solution unique. En particulier, les coefficients des variables corrélées seront dépendants les uns des autres, et une modification de la valeur de l'un peut être compensée par une variation dans la valeur des autres. Cela signifie que, dans le cas où les variables sont corrélées, la régression linéaire est instable : plusieurs solutions sont possibles.

* On apprend les coefficients d'une régression linéaire en **maximisant le log de la vraisemblance**, où, de manière équivalente si l'on suppose l'erreur normalement distribuée et centrée en zéro, en **minimisant la somme des carrés des erreurs**.
* Cette méthode s'appelle la **méthode des moindres carrés**.
* Si la matrice X^\top X est inversible, la régression linéaire admet une solution unique et explicite.
* Sinon, on peut calculer une solution grâce à un algorithme de calcul de pseudo-inverse, mais cette solution n'est pas unique.

Dans scikit-learn, la régression linéaire est implémentée comme [LinearRegression](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html) dans le module linear\_model.

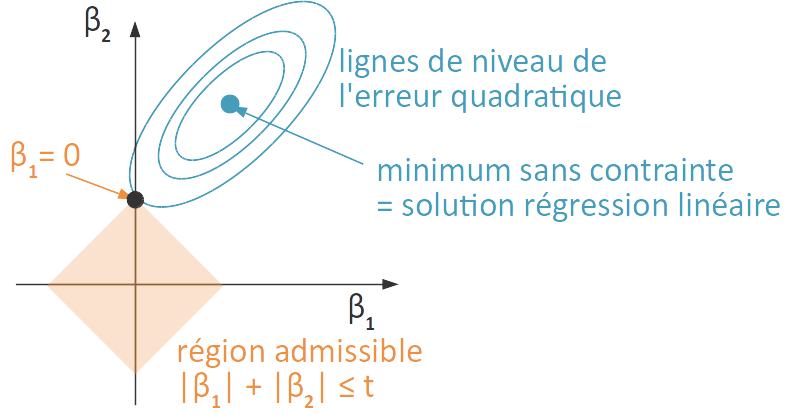


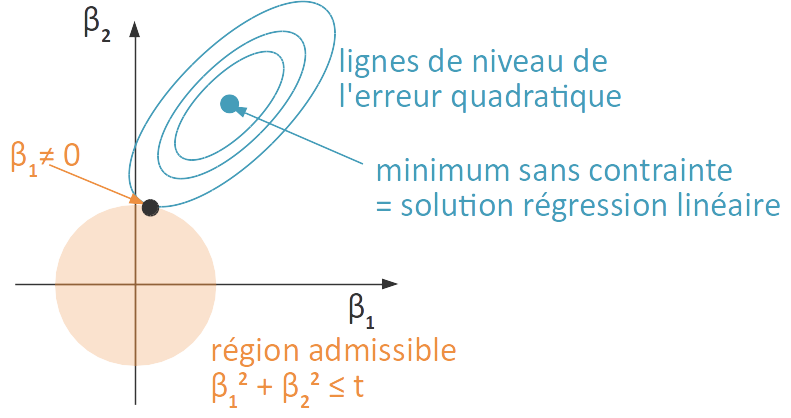
* La norme ℓ2 du vecteur de poids peut être utilisée comme terme de régularisation de la régression linéaire.
* Cela s'appelle la **régularisation de Tykhonov**, ou **régression ridge.**
* La régression ridge admet *toujours*une solution *analytique unique*.
* La régression ridge permet **d'éviter le surapprentissage**en **restraignant l'amplitude des poids.**
* La régression ridge a un effet de **sélection groupée** : les variables corrélées ont le même coefficient.

 La régression ridge est implémentée dans scikit-learn : [linear\_model.Ridge.](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html" \l "sklearn.linear_model.Ridge)

[linear\_model.RidgeCV](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeCV.html#sklearn.linear_model.RidgeCV)permet de déterminer la valeur optimale du coefficient de régularisation par validation croisée.

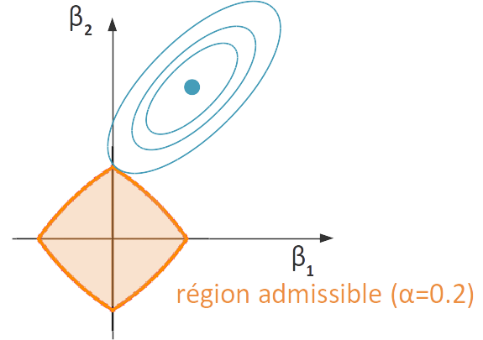
La régression ridge nous permet de réduire l'amplitude des coefficients d'une régression linéaire et d'éviter le sur-apprentissage.





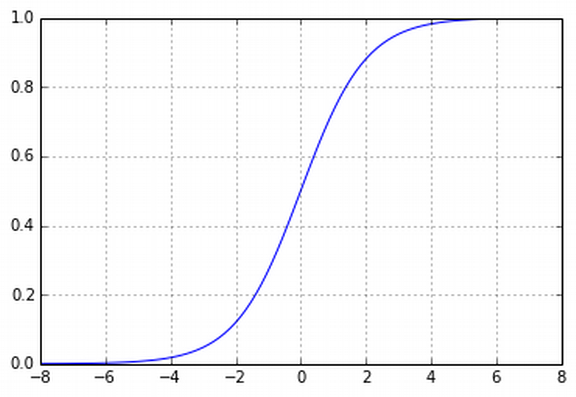
Si plusieurs variables corrélées contribuent à la prédiction de l'étiquette, le lasso va avoir tendance à choisir une seule d'entre elles (affectant un poids de 0 aux autres), plutôt que de répartir les poids équitablement comme la régression ridge. C'est ainsi qu'on arrive à avoir des modèles très parcimonieux. Cependant, laquelle de ces variables est choisie est aléatoire, et peut changer si l'on répète la procédure d'optimisation. Le lasso a donc tendance à être instable.

Elsatic Net :



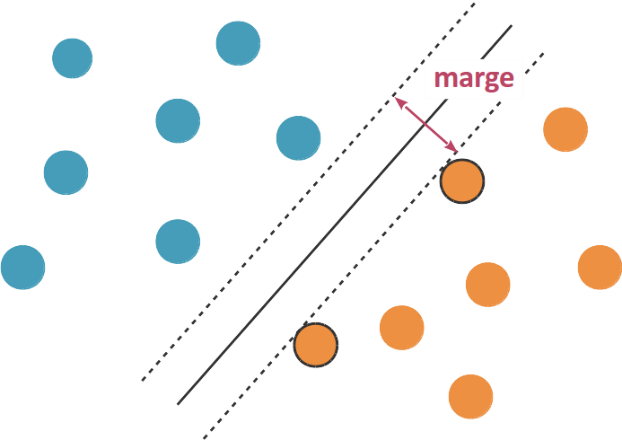
Le prix à payer en est le fait d'avoir maintenant non pas un seul mais deux hyperparamètres à sélectionner, λ et α , ce qui demande plus de ressources computationnelles...

* Le **lasso** utilise la norme ℓ1 du vecteur β comme régularisateur pour obtenir un modèle **parcimonieux**.
* Le lasso peut donc être utilisé comme un algorithme de **réduction de dimension supervisée.**
* Le lasso n'a pas de solution explicite, ni nécessairement unique.
* L**'elastic net** combine les normes ℓ1 et ℓ2 pour obtenir une solution *moins parcimonieuse* que le lasso, mais plus stable et dans laquelle toutes les variables corrélées pertinentes pour la prédiction de l'étiquette sont sélectionnées et reçoivent un poids identique.

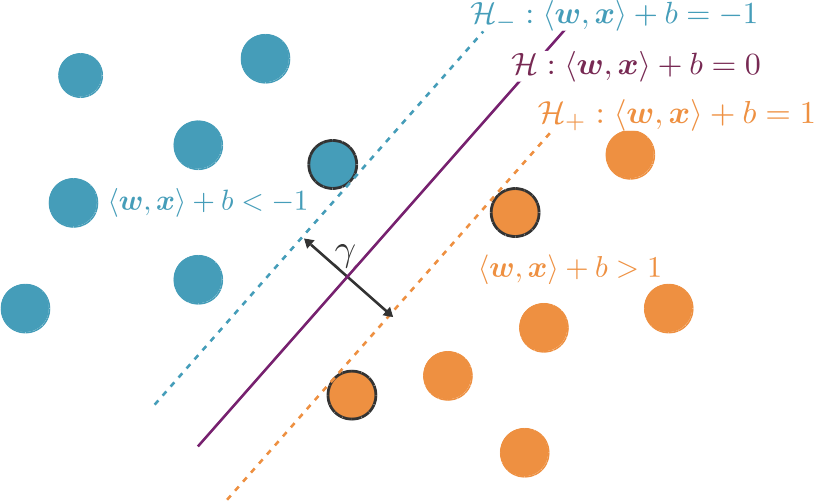


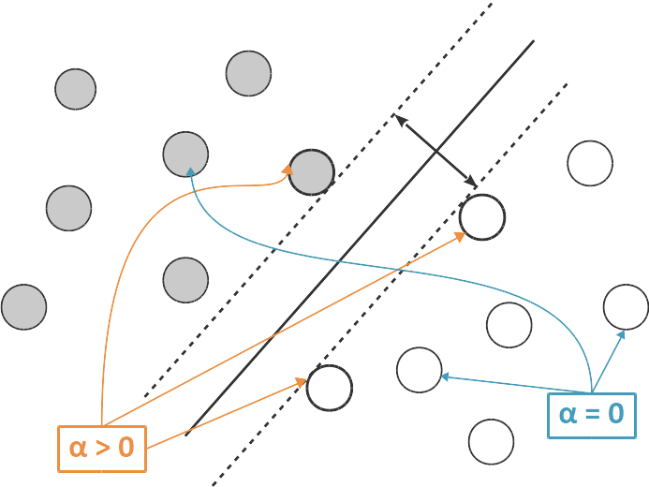
On pourra donc utiliser :

* La régression logistique avec régularisation ℓ2 pour éviter le sur-apprentissage (dans scikit-learn, c'est même l'implémentation par défaut de la régression logistique) ;
* La régression logistique avec régularisation ℓ1 pour obtenir un modèle parcimonieux (dans scikit-learn, il suffit d'utiliser l'option'penalty'=l1).



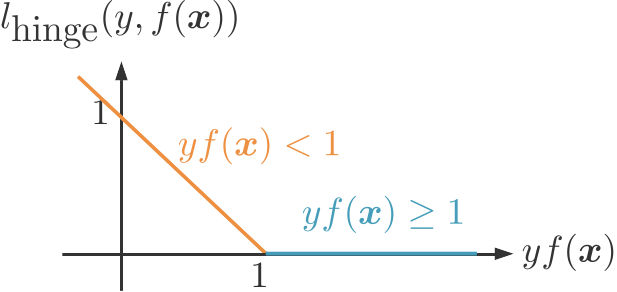
L'équation d'un hyperplan en dimension p est paramétrisée par les coordonnées du vecteur normal à cet hyperplan, w∈R^p ainsi que par un scalaire b∈R . Nous pouvons ainsi poser que l'équation de notre hyperplan séparateur de marge maximale H est ⟨w,x⟩+b=0 .



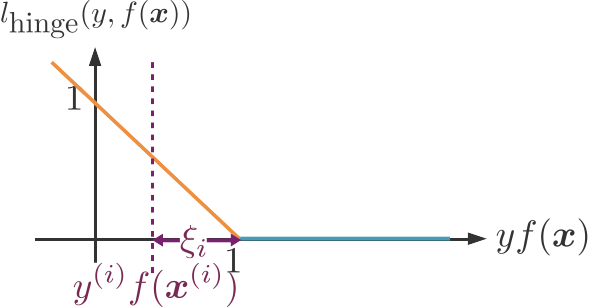


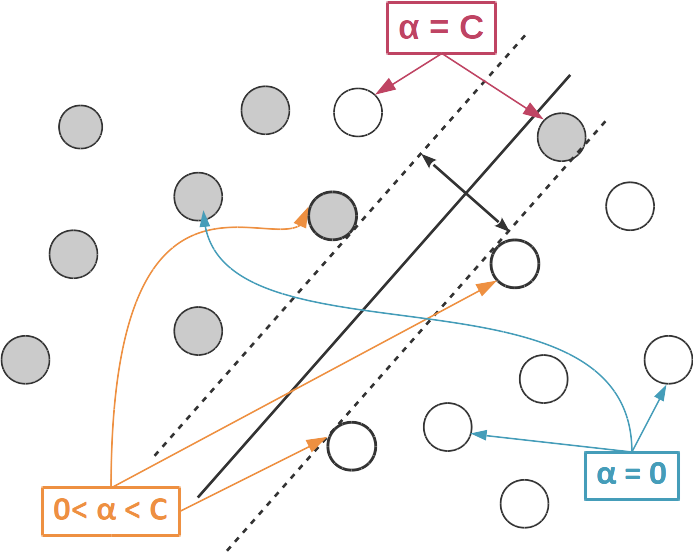
Par ailleurs, souvenez-vous de notre interprétation géométrique plus haut : seuls certains points auront un coefficient α∗i≠0 et ces points sont les vecteurs de support. La solution ne dépend donc pas des points qui ne sont pas vecteurs de supports, pour lesquels α∗i=0 Et c'est logique, car si l'on déplace un peu ces points, ils restent loin de la zone d'indécision. et l'hyperplan séparateur ne changera pas. Par contre, si l'on déplace un vecteur de support, il pourra entraîner avec lui H+ (ou H− s'il s'agit d'un point négatif), ce qui déplacera H.

Remarquons que le primal est un problème d'optimisation en p dimensions, alors que le dual est un problème d'optimisation en n dimensions. Si p≪n , alors résoudre le primal sera plus efficace. À l'inverse, si l'on a peu d'échantillons et beaucoup de variables (n≪p), résoudre le dual sera plus efficace.

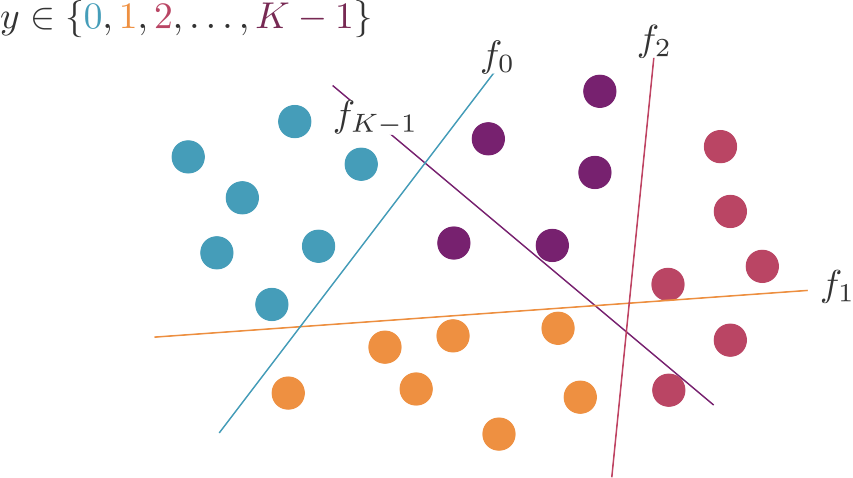


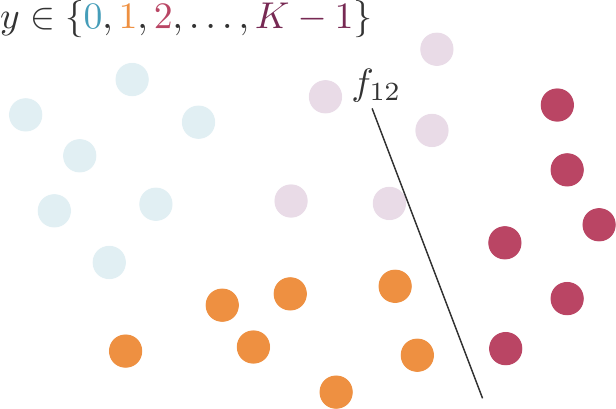
**SVM à marge souple :**





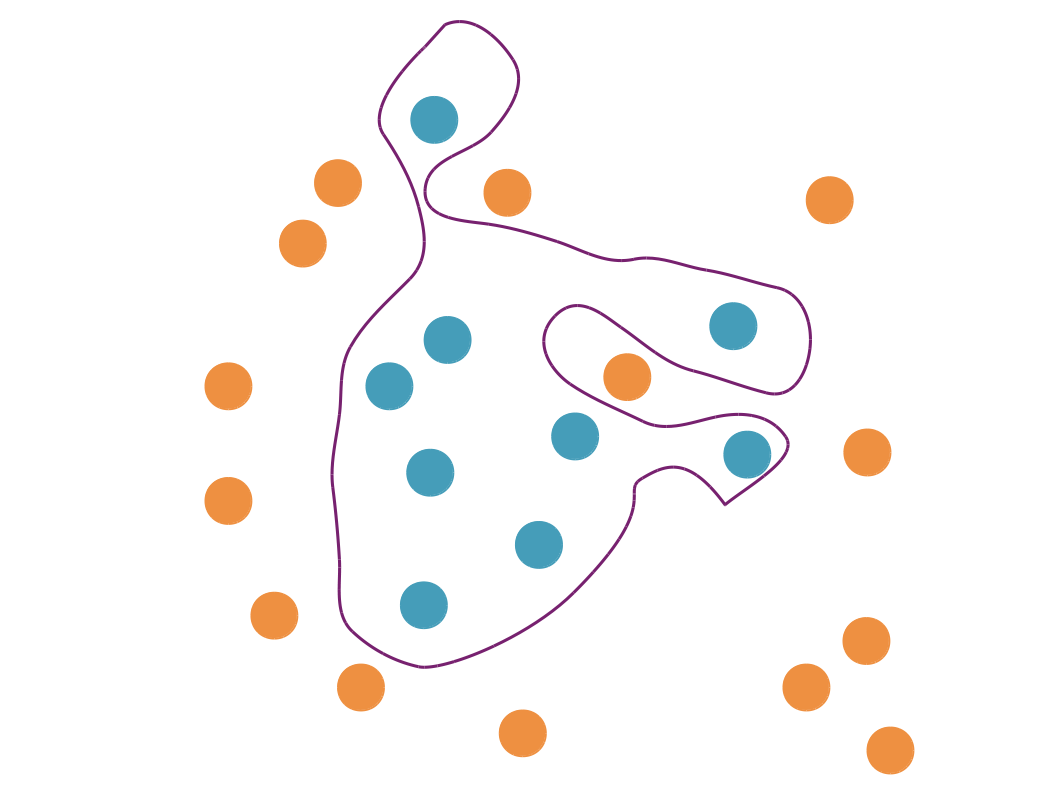
* Les **SVM (Support Vector Machines)**, aussi appelées en français **Machines à Vecteurs de Support**et parfois **Séparatrices à Vaste Marge**, cherchent à séparer *linéairement* les données.
* La version **primale**résout un problème d'optimisation à p variables et est donc préférable si on a moins de variables que d'échantillons.
* À l'inverse, la version **duale**résout un problème d'optimisation à n variables et est donc préférable si on a moins d'échantillons que de variables.
* Les**vecteurs de support** sont les points du jeu de données qui sont les plus proches de l'hyperplan séparateur.
* La **fonction de décision** peut s'exprimer uniquement en fonction du produit scalaire du point à étiqueter avec les vecteurs de support.

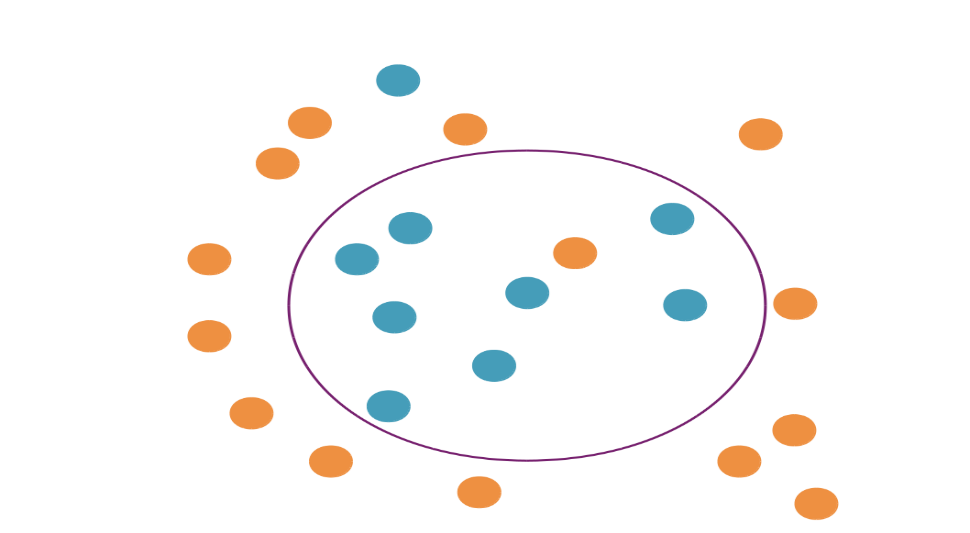


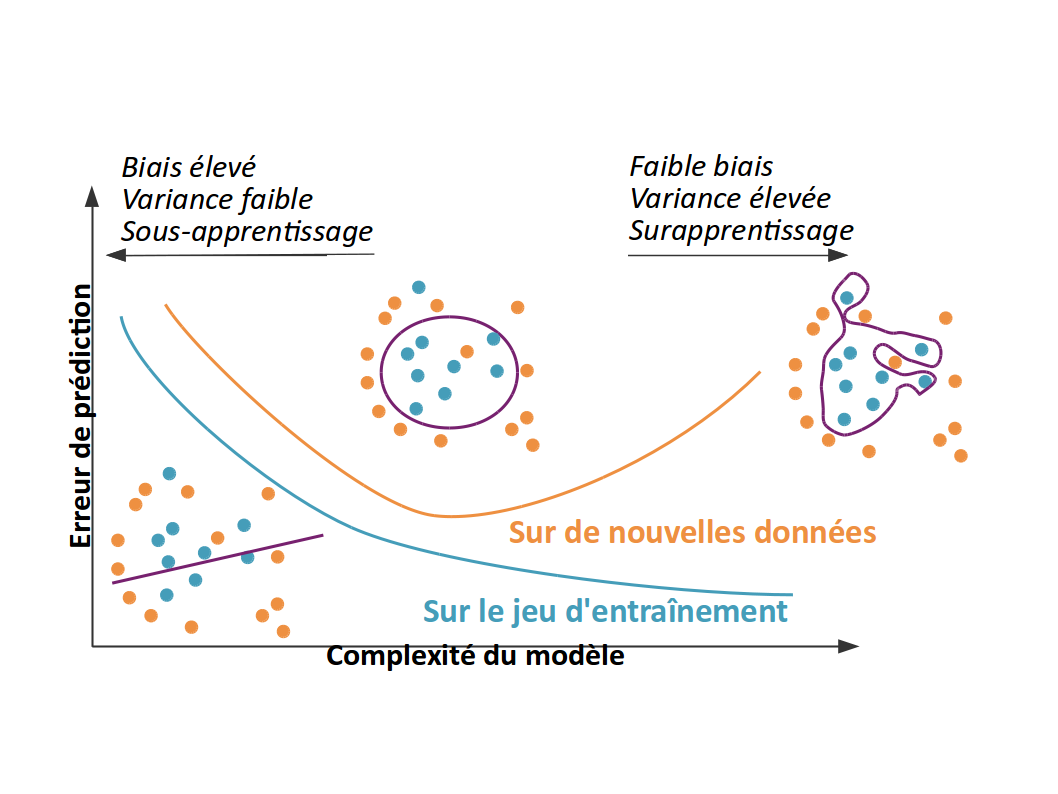


* L'approche **one-versus-rest** de la classification multi-classes consiste à créer K classifieurs binaires qui séparent chaque classe k de l'union des autres classes. On prédit alors la classe pour laquelle la fonction de décision est **maximale.**
* L'approche **one-versus-one** consiste à créer K(K-1) classifieurs binaires qui séparent chacun une classe d'une autre, en ignorant tous les autres points. On utilise alors un **vote de la majorité** pour prédire. Cette approche crée plus de classifieurs, mais chacun est entraîné sur moins d'observations.
* On peut créer des**SVM multiclasse** en optimisant simultanément K classifieurs binaires (méthode de **Crammer et Singer**).

EVALUATION

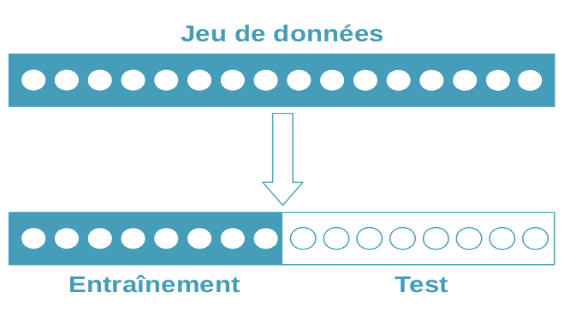


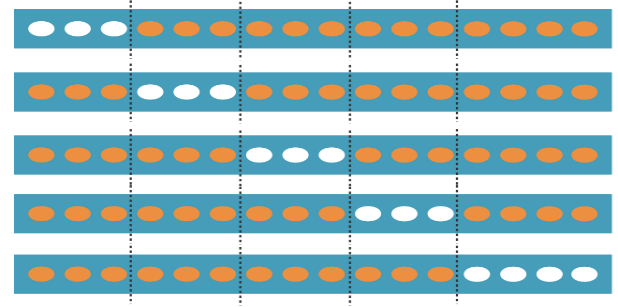


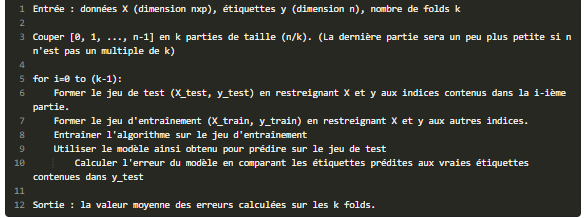


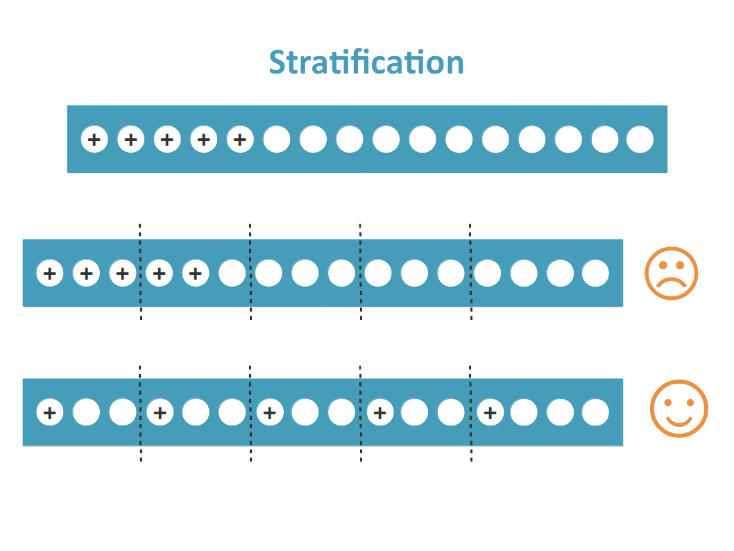
**Compromis biais-variance :** Un modèle simple (variance faible) risque le sous-apprentissage (biais élevé y compris sur les données d’entraînement). Un modèle complexe (variance élevée) risque le sur-apprentissage (biais faible sur les données d’entraînement mais élevé sur de nouvelles données). On souhaite trouver un modèle intermédiaire, vers le creux de la courbe orange, là où le biais de prédiction est le plus faible et la généralisation la meilleure.

* En apprentissage supervisé, le but est de produire des modèles qui **généralisent**, c’est-à-dire qui sont capables de faire de bonnes prédictions sur de nouvelles données
* De bonnes performances sur le jeu d’entraînement **ne garantissent pas** que le modèle sera capable de généraliser !
* On cherche à développer un modèle qui soit suffisamment complexe pour bien capturer la nature des données (et éviter ainsi le sous-apprentissage), mais suffisamment simple pour éviter le **sur-apprentissage**.
* Attention aux **contraintes de temps de calcul** et aux **ressources en mémoire** !



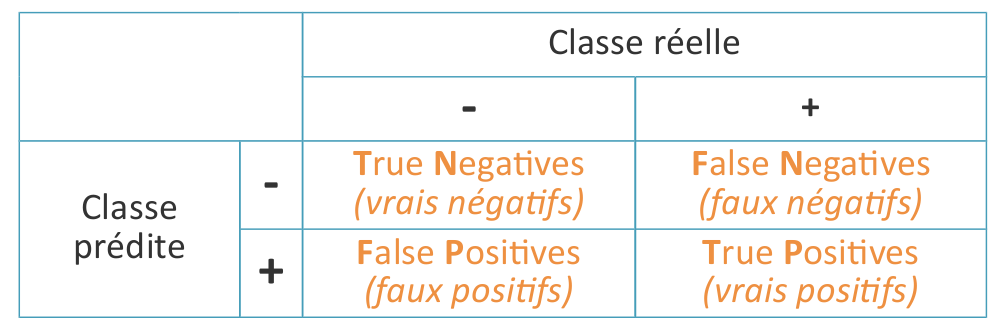




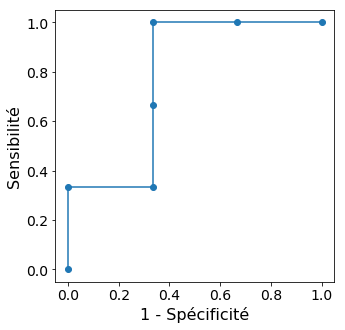


* Il ne faut **jamais** évaluer un modèle sur des points qui ont été utilisés pour l’entraîner.
* On sépare donc les données entre **un jeu d’entraînement**, sur lequel on apprend le modèle, et **un jeu de test**, sur lequel on l’évalue.
* Pour utiliser l’intégralité de nos données pour entraîner et pour tester, et pour éviter un biais potentiel lié au fait de faire une évaluation unique, **on préfère faire une validation croisée**.
* Dans le cas d’un problème de classification, on fait attention à **stratifier la validation croisée** pour éviter d’introduire des biais.

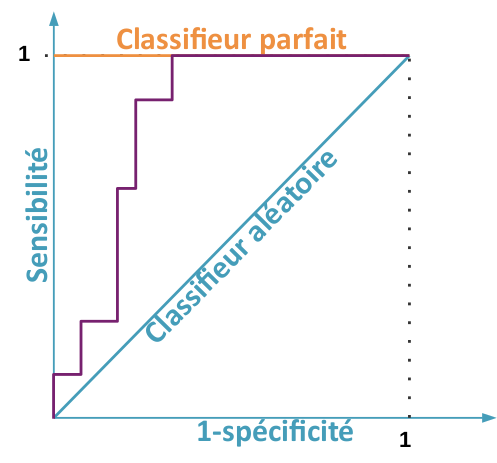
Matrice de confusion



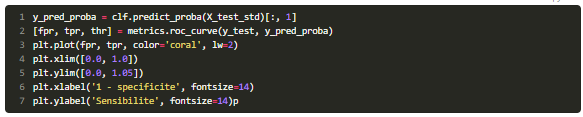
* **Toutes les erreurs ne se valent pas**.
* On peut représenter les performances d’un modèle de classification dans une**matrice de confusion**.
* On peut extraire de cette matrice différentes mesures de performance, comme le **rappel**, la **précision**, la **spécificité** et le **F-score**, qui reflètent différents aspects du modèle.

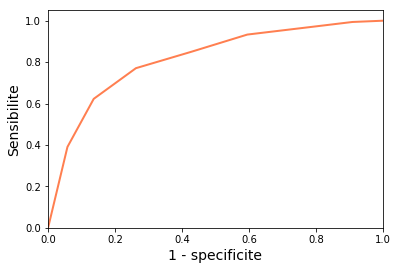


Chaque fois que l'on considère une nouvelle fourchette de valeur de seuil (ou, de manière équivalente, que l'on prédit positive une observation de plus), seulement l'une des deux valeurs (sensibilité ou anti spécificité) change (la première si l'observation est effectivement positive, la seconde sinon). La courbe ROC est donc une courbe en escaliers

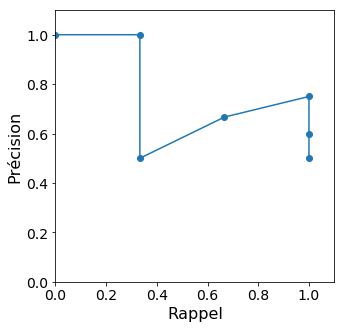


Traçons la courbe ROC du kNN

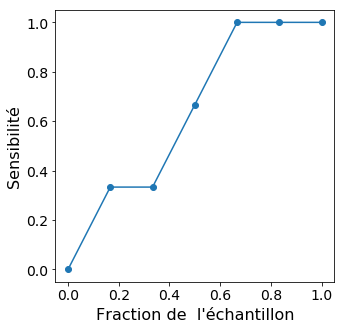


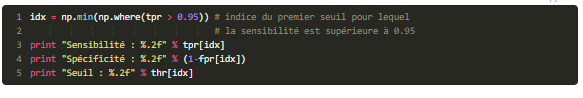


Courbr ROC de KNN



Petit problème : pour le seuil le plus élevé, la précision n'est pas définie car aucune observation n'est prédite positive... Par convention, on choisira souvent une précision de 1 si la première observation à considérer est positive, et une précision de 0 sinon.



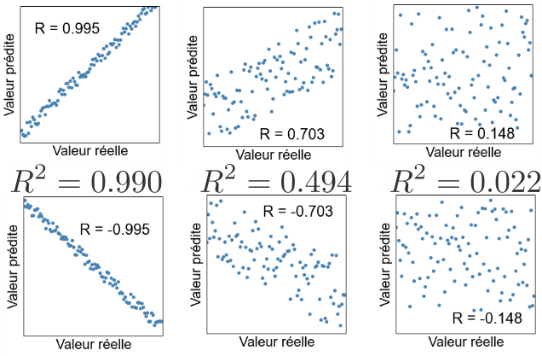


Utiliser un seuil de 0.29 nous garantit une sensibilité de 0.97 et une spécificité de 0.21, soit un taux de faux positifs de… 79%.

* De nombreux modèles de classification retournent des **valeurs réelles**, qui peuvent souvent être interprétées comme la probabilité que le point appartiennent à la classe positive.
* Dans ce cas, il faut se fixer un **seuil** sur cette valeur réelle pour séparer les négatifs des positifs.
* La **courbe ROC** permet de visualiser comment la spécificité et la sensibilité d’un modèle évolue en fonction de ce seuil.
* L’**AUROC** permet de résumer la courbe ROC en un seul nombre : l’aire sous cette courbe.
* Les performances d'un modèle **dépendent du jeu de données**.
* Les**approches naïves** sont des approches simples qui n'apprennent pas vraiment mais servent de point de comparaison pour évaluer nos modèles.
* Pour un problème de classification, on peut utiliser **une des approches naïves suivantes** :
  + Retourner toujours la même classe ;
  + Retourner une classe aléatoire ;
  + Retourner un score aléatoire, puis utiliser un seuil.

Le coefficient de détermination nous indique donc à quel point les valeurs prédites sont corrélées aux vraies valeurs.

Attention, si les prédictions sont fortement anti-corrélées aux vraies valeurs, le coefficient de détermination sera élevé aussi.

* 
* la *somme des carrés des résidus* (**RSS**) ;
* la *moyenne* de cette somme (**MSE**) ;
* la *racine carrée*de cette moyenne (**RMSE**).

 On peut préférer calculer la **corrélation** entre valeurs prédites et vraies valeurs :

* l'**erreur carrée relative** (**RSE**) est la RSS normalisée par la somme des carrés des distances entre les étiquettes et leur moyenne ;
* elle est le complément à 1 du **coefficient de détermination** (R2), qui est le carré de la corrélation de Pearson entre valeurs prédites et vraies valeurs.

On en conclut que la réduction de dimension est nécessaire à la qualité de l'apprentissage.

Réduire la dimensionalité des données, c'est-à-dire le nombre de variables utilisées pour les représenter, permet :

* de **faciliter la visualisation** des données ;
* de **réduire les coûts**de calcul, de stockage et d'acquisition des données ;
* d'**améliorer l'apprentissage** en construisant des **modèles moins complexes**, en éliminant les variables non pertinentes qui pourraient **fausser les prédictions** et enfin en réduisant le problème du **fléau de la dimensionalité.**

Dans la suite de cette partie, nous allons voir des techniques *linéaires* de réduction de dimension non-supervisée.

Nous cherchons donc w1w1 tel que

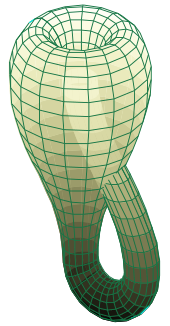
* w1⊤Σw1 est maximale
* ||w1||=1

On peut l'utiliser pour choisir :

* le nombre de composantes principales qui explique un pourcentage de la variance que l'on s'est initialement fixé (par exemple, 90%)
* le nombre de composantes principales correspondant au « coude » du graphe, à partir duquel ajouter une nouvelle composante principale ne fait pas grande différence.
* RESUME :
* Les composantes principales forment une base orthonormée et sont construites pour que les données aient une variance maximale selon ces nouveaux axes.
* Les composantes principales sont en fait les vecteurs propres de la matrice de covariance des données, classées par ordre décroissant de valeur propre correspondante.
* Pour choisir le nombre de composantes à utiliser, on regarde la proportion de la variance totale expliquée par k composantes.

Dans l'analyse factorielle, nous ne faisons plus l'hypothèse que les nouvelles dimensions sont orthogonales. On peut ainsi en particulier se retrouver avec des dimensions dégénérées (des vecteurs colonnes de W dont toutes les entrées soient 0) et moins de dimensions qu'avec une ACP.

* L'ACP peut être vue comme une factorisation de la matrice de données X=WH, où W continent les nouvelles variables et H la représentation latente des données selon ces variables.
* Toute une famille de méthodes réduisent la dimension des données par une décomposition approximative, X≈WH.
* L'analyse factorielle et l'ACP probabiliste sont issues de modèles génératifs utilisant la factorisation de la variable aléatoire x en Wh. Dans le cas de l'ACP probabiliste on modélise la covariance du bruit comme étant un multiple de l'identité. Dans le cas de l'analyse factorielle on estime la covariance du bruit indépendante dans chaque direction.
* L'**ACP** cherche à maximiser la variance de X selon des directions orthogonales.
* L'**analyse factorielle** cherche à modéliser la structure de la covariance des variables observées et ne définit pas nécessairement des axes orthogonaux.
* La **NMF (factorisation non-négative de matrice)** s'applique à des matrices X dont les entrées sont toutes positives et force les entrées de W et de H à être positives aussi. Elle permet aussi de prédire les valeurs manquantes de X.



Bouteille de Klein

L'action de dérouler (aplatir) la bouteille est une transformation. Plus tard, on appellera cette transformation fonction Φ. Très souvent, l'action de dérouler sera très complexe.

Comme je le disais plus haut, l’idée est de généraliser des algorithmes qui travaillent dans des espaces euclidiens pour passer sur des variétés, donc des espaces non-linéaires.

* Les données non-linéaires sont abordables de manière explicites en effectuant directement un feature engineering sur les observations de départ, ou bien de manière implicite en utilisant des méthodes spécifiques à la non-linéarité. On retrouvera parmi ces méthodes les méthodes à noyaux et les méthodes à plongement dans des variété.
* Les données à grandes dimensions reposent souvent sur une variété non-linéaire de dimension bien inférieure (ou bien sont proches de celle-ci).
* Les méthodes de réduction dimensionnelles non linéaires que nous allons étudier permettent ainsi de retrouver une approximation de ces variété et de bien mieux représenter ces données dessus.
* On s’intéresse aux informations liées à la structure du phénomène représenté, mais c’est parfois suffisant de n'observer que les paires de distances entre les points dans l’espace original. C’est le cas d’une partie des méthodes que nous allons étudier.

 L’idée générale, c’est qu’un kernel doit permettre de représenter la similarité entre les deux points sur lesquels on effectue le produit. On peut même ne pas utiliser de données numérique (par exemple des données textes) tant que cette notion de similarité entre les observations est respectée.

En effet, un des problèmes principaux des méthodes à noyaux sont qu’elles nécessitent le stockage en mémoire de la matrice K(x,y) sur les données afin de pouvoir être utilisées.

* Le plongement d’un algorithme linéaire dans une variété non-linéaire peut s’effectuer à l’aide du kernel trick
* Les méthodes à noyaux permettent de ne pas avoir à définir la variété dans laquelle on travaille pour appliquer nos algorithmes
* La réduction dimensionnelle non-linéaire par kPCA n’est pas la plus utilisée, mais elle illustre la transition entre les méthodes linéaires et non-linéaires

Une première famille de méthodes de réduction dimensionnelles non-linéaire est celle des méthodes dites globales. Comme je l’ai expliqué dans le chapitre précédent, cette famille permet de traiter le jeu de données avec un focus sur les distances entre les paires de points, qu’ils soient distants ou proches.

### MDS (Multidimensional Scaling) :

Méthode de réduction dimensionnelle qui utilise les distances entre les points, qui vous aidera à expliquer les similarités et dissimilarités observées dans votre jeu de données, en les représentant dans un espace géométrique de dimension inférieure à celle de l’espace initial.

MDS permet d’utiliser n’importe quel type de mesure de similarité comme “distance” (dont la distance euclidienne bien sûr 😉).

Il faut minimiser une fonction objective (ou fonction de coût). Dans notre cas présent, l’objectif est de minimiser l’erreur entre les distances approximées dans l’espace de dimension inférieure, et les distances réelles dans l’espace de départ.

En effectuant cette MDS de base avec les distances euclidiennes (appelée MDS métrique).

La première c’est que l’ACP conserve les propriétés de distances ce qui est une bonne chose. La seconde, c’est que MDS est un algorithme linéaire.

la famille de méthodes MDS non-métriques.

Une première famille de méthodes de réduction dimensionnelles non-linéaire est celle des méthodes dites globales.

L’algorithme Isomap est une méthode pratique lorsqu’on veut conserver la structure géométrique globale du phénomène, à petite et à grande échelle. C'est un des algorithmes à tester lorsqu'on est face à un problème non-linéaire à explorer.

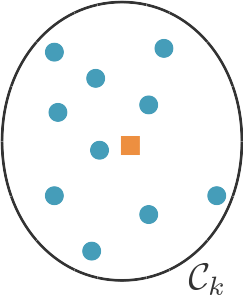
Resumé :

On a vu dans ce chapitre l’algorithme t-SNE qui est une des méthodes les plus utilisées pour visualiser des données à grandes dimensions non-linéaires afin de **repérer des structures locales**intéressantes pour le travail de modélisation.

Attention encore une fois à bien développer une intuition de ce que peut apporter une visualisation t-SNE et à bien en faire plusieurs exécutions avec différents hyperparamètres. Il faut tester un bon nombre de fois ces méthodes avec des datasets différents pour bien les prendre en main.

Les algorithmes de clustering dépendent donc fortement de la façon dont on définit cette notion de similarité, qui est souvent spécifique au domaine d'application.

* Les algorithmes de clustering permettent de partitionner un jeu de données en sous-groupes d'observations similaires ;
* Ils peuvent être utilisés pour
* - mieux comprendre les données ;
* - faciliter la visualisation des données ;
* inférer des propriétés des données.

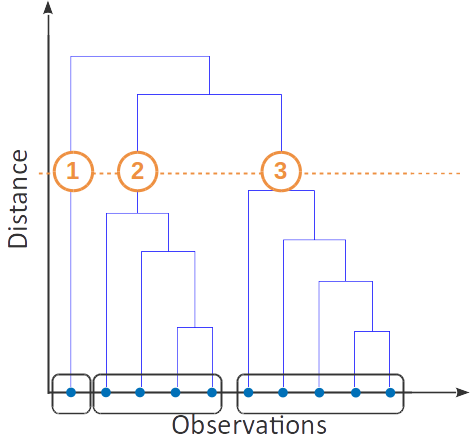


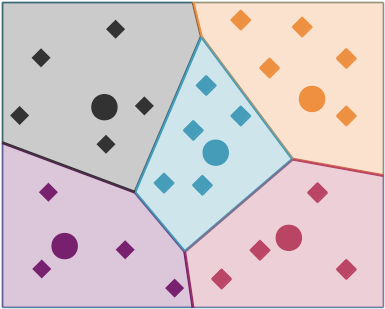
Pour évaluer un algorithme de clustering, on peut s'intéresser à :

* La forme des clusters qu'il produit (sont-ils denses, bien séparés). On utilise ici souvent le coefficient de silhouette ;
* La stabilité de l'algorithme ;
* La compatibilité des résultats avec des connaissances spécifiques au domaine, que l'on peut évaluer à l'aide de mesures d'enrichissement.

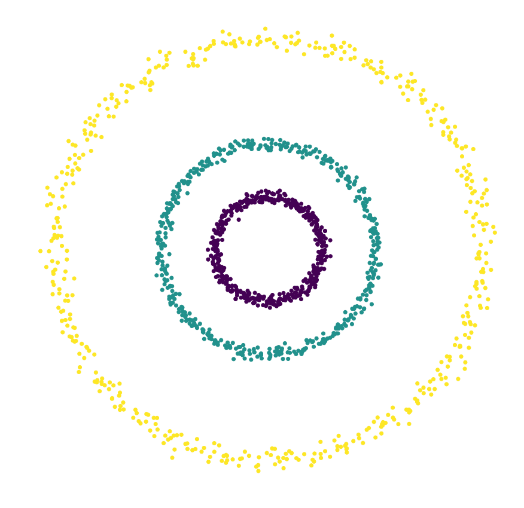
La longueur des U est proportionnelle à la distance entre les deux clusters qu'elle connecte.

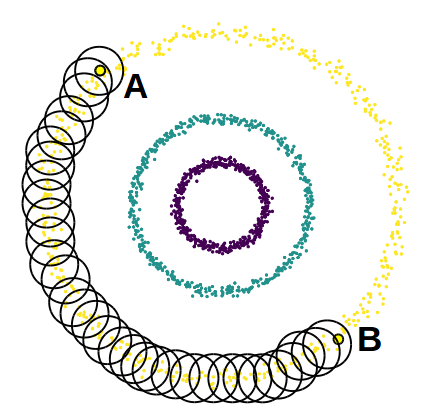
* Le clustering hiérarchique permet de partitionner un jeu de données de manière hiérarchique.
* À chaque étape, on agrège les deux clusters les plus proches
* La distance entre clusters peut être calculée de la façon suivante :
  + Lien simple : la distance entre deux clusters est celle entre les deux points les plus proches.
  + Lien complet : la distance entre deux clusters est celle entre les deux points les plus éloignés.
  + Lien centroïdal : la distance entre deux clusters est celle entre les deux centroïdes.
  + Lien moyen : la distance entre deux clusters est la distance moyenne entre les points des deux clusters.
  + Clustering de Ward : la distance entre deux clusters est calculée de façon à minimiser la variance inter-cluster.
* On peut visualiser une partition hiérarchique des données avec un **dendrogramme**

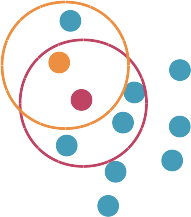




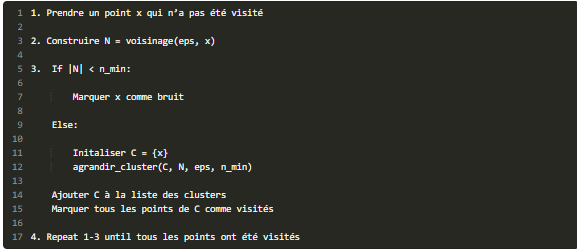
* L'algorithme du k-means permet de rechercher efficacement une partition des données dont la variance intra-cluster est minimale
* Cependant il s'agit d'une approche heuristique qui peut retourner un minimum local plutôt que global
* L'initialisation kmeans++ ainsi que des répétitions multiples permettent de mitiger ce problème



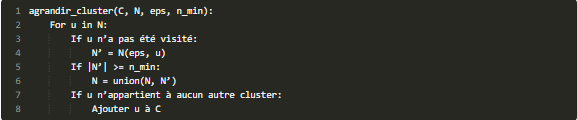
On peut connecter A à B par des petits voisinages (les cercles) contenant uniquement des points du cluster extérieur.



L'algorithme de DBSCAN est le suivant :



La procédure agrandir\_cluster est donnée par



DBSCAN a le grand avantage d'être efficace en temps de calcul sans requérir de prédéfinir le nombre de clusters. Enfin, comme je l'ai dit au début de ce chapitre, il permet de trouver des clusters de forme arbitraire.