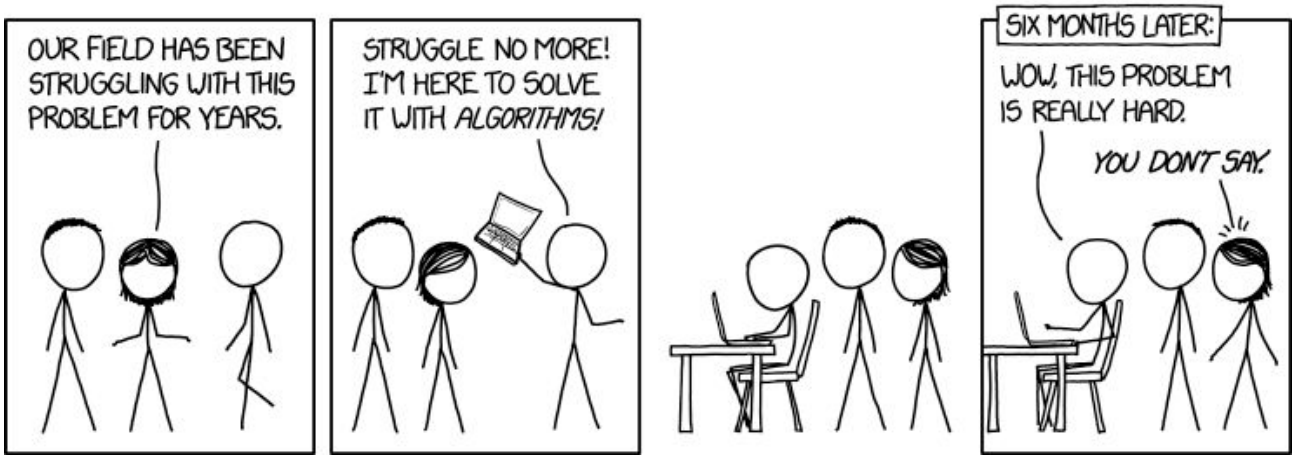

# Algorithmes de Machine Learning

## Clustering et des autres




Algorithms are hard

*Source: xkcd*

## Objectifs du module

**Go deeper to understand the workings of some machine learning algorithms**

- En savoir plus sur les hyperparamètres d'arbres de décision, les forêts aléatoires et les machines à vecteurs de support
- Utilisez des courbes d'apprentissage pour aider à évaluer les performances du modèle et suggérer des améliorations
- Comprendre le fonctionnement de l'algorithme de clustering k-means en l'implémentant vous-même

## Modalités

- Durée du projet : 3 jours
- Travailler en équipes de deux
- Produire vos propres scripts et memos individuels pour terminer le projet

## Contexte

Vous avez maintenant implémenté de nombreux algorithmes à l'aide de scikit-learn. Dans ce module, nous prenons le temps de mieux comprendre le fonctionnement de ces algorithmes d'apprentissage automatique. Nous prenons également un certain temps pour comprendre plus clairement comment les choix de modèle et leurs hyperparamètres affectent le biais et la variance que nous voyons dans nos modèles. Vous mettrez en œuvre des «courbes d'apprentissage» qui vous aideront à interpréter vos modèles par rapport à ces concepts de biais et de variance.

Nous allons également passer un peu de temps à comprendre l'algorithme de clustering k-means en l'implémentant nous-mêmes à partir de zéro en python! C'est une bonne occasion de mettre en pratique vos compétences en écriture algorithmique et en codage. Il est également important d'écrire du code reproductible, nous y réfléchirons également brièvement.

# Etape 1 (1 jour)

## Algorithmes ML et courbes d'apprentissage

### Objectifs de l'activité

- Être capable d'expliquer comment: l'arbre de décision, la forêt aléatoire et les algorithmes SVM fonctionnent
- Comprendre les hyperparamètres qui existent pour régler ces modèles

- Mettre en œuvre des courbes d'apprentissage pour les algorithmes ML et les interpréter pour expliquer comment le choix du type de modèle et des hyperparamètres affecte le sur-ajustement et le sous-ajustement

### Compétences

- Expliquez ce qu'est un SVM
- Expliquez ce que sont les arbres de décision / forêts aléatoires
- Mettre en œuvre des courbes d'apprentissage de modèle

### Consignes

- Prenez le temps de lire les sections du livre pour l'arbre de décision, la forêt aléatoire et les algorithmes SVM.
- Importez ces algorithmes à partir de scikit-learn et recherchez les options d'hyperparamètre disponibles lorsqu'elles sont instanciées.
- Lisez les sections du chapitre 4 de «Régression polynomiale et courbes d'apprentissage» dans le livre «Pratique: Apprentissage automatique…»
- Mettez en œuvre le code du livre sur les courbes d'apprentissage sur les données qu'il a utilisées. Assurez-vous de bien comprendre ce que fait le code. Parlez-en à vos voisins!
- Implémentez le code du livre sur un problème de ML que vous avez déjà rencontré dans le cours (copiez et collez le code du référentiel du livre! Vous pouvez essayer le jeu de données des arbres de Grenoble depuis 'Intro au ML'). Produisez un tracé d'un modèle qui montre des signes de sous-ajustement et un qui montre des signes de sur-ajustement. Essayez de modifier les hyperparamètres du modèle et voyez comment cela affecte la courbe d'apprentissage. Expliquez comment le choix du type de modèle et des hyperparamètres affecte le sur-ajustement et le sous-ajustement. Parlez-en à vos voisins!

### Livrables

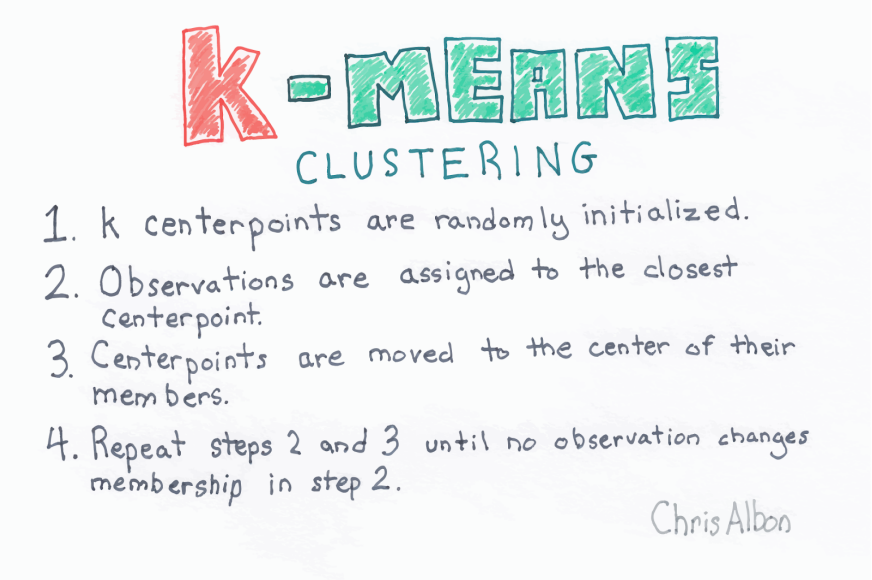
- Un mémo qui explique (dans vos propres mots) les termes suivants:
    - Decision tree
    - Random forest
    - SVM
- Script / cahier python (ou Memo) qui contient:
    - Un exemple de tracé de courbe d'apprentissage d'un modèle qui est «underfit»'
    - Un exemple de tracé de courbe d'apprentissage d'un modèle qui est «overfit»'

### Pour aller plus loin

- Enquêter sur l'utilisation de l'implémentation des courbes d'apprentissage de scikit-learn (ou yellowbrick). Quelle est la différence entre cela et le code du livre?
- Essayez d'implémenter des courbes d'apprentissage sur un problème de classification que vous avez déjà rencontré dans les modules précédents.

# Etape 2 (2 jours)

## Implémentez votre propre algorithme de clustering k-mean




There are no funny pictures of clustering
*Source :* [Chris Albon]

### Objectifs de l'activité

- Comprendre la mise en œuvre du clustering k-means
- Pratiquez l'écriture de votre python et algorithme en créant votre propre module k-means!

### Compétences

- Mettre en œuvre un algorithme k-means pour un apprentissage non supervisé

### Consignes

- Lisez le livre «Hands on ML», chapitre 9, introduction: p.235 - 248. Jetez un œil aux autres ressources ci-dessous.
- Assurez-vous de bien comprendre les étapes que l'algorithme k-means utilise pour trouver les étiquettes de cluster
- Vous allez créer un module qui utilise k-means pour affecter des étiquettes à un ensemble de données en utilisant uniquement des bases python et numpy de base. Les données sont bidimensionnelles et au format d'un tableau numpy.
    - Faites un premier aperçu de votre algorithme (sur papier si nécessaire!). Assurez-vous d'écrire un pseudo-code pour votre algorithme dans un script python - avant de faire du codage! (Ça va vraiment aider!)

    - Pensez aux différents éléments de python dont vous aurez besoin et à leur emplacement dans votre pseudo-code (par exemple, quels types de données, boucles, opérations, vous devrez peut-être utiliser). Si vous n'êtes pas sûr de leur fonctionnement, pensez à passer du temps dans les prochains jours à apprendre comment ils fonctionnent lorsque vous devez les utiliser. Cela aidera vos compétences de codage à long terme. Si vous avez besoin d'aide pour des idées, demandez à l'instructeur.
    - Pensez à utiliser un IDE (par exemple «spyder») pour écrire votre code. Avoir des outils de débogage sera très pratique.
    - Pensez à utiliser git pour contrôler la version de votre code.
    - Assurez-vous de commenter votre code au fur et à mesure, afin que les autres (et l'avenir vous!) Puissent le comprendre. Si votre code final n'est pas bien commenté, vous ne passerez pas la compétence.
    - Testez votre algorithme sur les données fournies. Votre algorithme doit fournir des étiquettes pour chacun des points de données.
    - Créez une fonction pour tracer les: données fournies et les centroïdes de cluster. Colorez les points de données de chaque cluster avec des couleurs distinctes.
- Utilisez votre module pour trouver le nombre optimal de clusters pour les données.

### Ressources

- Clustering dans le chapitre du livre scikit-learn (voir chapitre 9)
  https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/
- Brève explication des k-means
  https://www.youtube.com/watch?v=_aWzGGNrcic
- K-means interactif
  https://www.naftaliharris.com/blog/visualizing-k-means-clustering/
- Introduction à Spyder IDE
  https://www.youtube.com/watch?v=zYNRqVimU3Q

### Livrables

- Script / cahier python qui contient:
    - Une fonction appelée ‘k-means’ qui prend des arguments d’entrée: un tableau d’entrée bidimensionnel et un paramètre ‘nombre de grappes’. Il doit renvoyer un tableau unidimensionnel qui contient les étiquettes de chaque cluster.
    - Un tracé du résultat sur les données fournies (décrit ci-dessus)

### Pour aller plus loin

- Assurez-vous que votre code est conforme aux directives de bonnes pratiques (comme pep8)
  https://www.python.org/dev/peps/pep-0008/
  https://www.python.org/dev/peps/pep-0257/
  https://stackoverflow.com/questions/356161/python-coding-standards-best-practices
- Augmentez le nombre de points dans l'ensemble de données. Temps nécessaire à l'exécution de votre module. Comparez-le à l'implémentation de k-means de scikit-learn.

- Pensez à refactoriser votre code (réécriture) afin qu'il soit plus simple à lire ou qu'il s'exécute plus rapidement (les deux peuvent être importants!).
- Pensez à améliorer la méthode d'initialisation aléatoire de votre algorithme (voir page 243 «ML pratique») pour éviter des résultats sous-optimaux.
- Faites de votre module une classe qui peut être importée et implémentée de la même manière que scikit-learn.
  https://www.youtube.com/watch?v=ZDa-Z5JzLYM
- Combinez votre code avec quelqu'un d'autre dans un dépôt github. Testez le code. Intégrez pour supprimer tous les bugs.