REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABOU-BEKR BELKAID - TLEMCEN

LAT
Laboratoire d'Automatique de Tlemcen

MEMOIRE

Présenté à :

FACULTE DES SCIENCES – DEPARTEMENT DE PHYSIQUE

Pour l'obtention du diplôme de :

MASTER EN PHYSIQUE

Spécialité : Physique Computationnelle

Par :

M^lle GUENNINECHE Amel

Sur le thème

Prédiction des propriétés des matériaux par apprentissage automatique

Soutenue publiquement le 29/06/2019 devant le jury composé de :

Mr Abdelkrim MERAD	Professeur à l'Université de Tlemcen	Président
Mme Latifa BETTADJ	MCA à l'Université de Tlemcen	Encadreur
Mr Mohammed Réda BOUFATAH	MCB à l'Université de Tlemcen	Co-encadreur
Mr Kamel KARA ZAITRI	MAA à l'Université de Tlemcen	Examinateur

Laboratoire Automatique Tlemcen(LAT)
Tlemcen - Algérie

Dédicaces

Je dédie ce travail

A mes chers parents, pour leurs sacrifices, leur amour, leur soutien et leurs prières tout au long de mes études.

Qu'ils trouvent ici le témoignage de ma profonde reconnaissance,

A mes chères sœurs *Nawel* et ***Lynda***, et **mon chère frère *Mohammed Youcef*** pour leurs encouragements permanents, et leurs soutient moral,

A tous mes amis *Mohammed Nadir B, Fatima S* et ***Achouak B***, pour leur aide et support dans les moments difficiles.

Merci d'être toujours là pour moi.

Puisse dieu vous donne santé, bonheur, courage et surtout réussite.

Amel

Remerciements

Je tiens à remercier dans un premier temps, toute l'équipe pédagogique du laboratoire d'automatique à l'université de Tlemcen et les intervenants professionnels responsables de la filière physique computationnelle, pour avoir assuré la partie théorique.

Le travail présenté dans ce mémoire a été réalisé sous la direction du **Madame Latifa BETTADJ**, ma plus grande gratitude va à mon encadreur pour m'avoir donné la chance de réaliser ce modeste travail.

Je remercie également mon Co-encadreur **Monsieur Mohammed Réda BOUFATAH**, pour sa disponibilité et la confiance qu'il m'a accordée. J'ai profité pendant longtemps du savoir et du savoir-faire dont j'ai pu bénéficier au cours de nombreuses discussions. J'aimerais aussi le remercier pour l'autonomie qu'il m'a accordée, et ses précieux conseils qui m'ont permis de mener à bien ce travail.

J'exprime toute ma reconnaissance à **Monsieur Abdelkrim MERAD** pour avoir bien voulu accepter de présider le jury de ce mémoire. Que **Monsieur Kamel KARA ZAITR** de l'université de Tlemcen, trouve ici l'expression de mes vifs remerciements pour avoir bien voulu juger ce travail.

Table des matières

Table des figures

INTRODUCTION GENERALE

Introduction générale :

Depuis les années 1980, la capacité de stockage de données ne cesse d'augmenter, et les outils analytiques traditionnels ne sont pas suffisamment performants pour exploiter pleinement la valeur du Big Data. Un large volume de données ne mène pas toujours à des informations utiles ce qui rend l'obtention des analyses compréhensives de plus en plus difficiles. Par conséquent, les analystes sont tournés vers ce qu'on appelle « Apprentissage Automatique » ou « Apprentissage Statistique » ou « Machine Learning ».

L'une des premières définitions du « Machine Learning » a été proposée en 1959 par l'ingénieur électricien et informaticien Américain Arthur Samuel, qui l'a défini comme étant un domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmés [1].

L'apprentissage automatique a permis à l'homme d'améliorer non seulement de nombreux processus industriels et professionnels, mais aussi de faire progresser la vie quotidienne. Les assistants personnels virtuels utilisent un algorithme d'apprentissage automatique, comme Siri, Alexa et Google. Ils aident à trouver des informations lorsqu'on leur demande par voix, rappeler nos requêtes correspondantes ou envoient une commande à d'autres ressources telles que des applications téléphoniques, pour satisfaire nos demandes. L'apprentissage automatique est également utilisé dans les services de navigation pour prévoir la circulation. De plus, les réseaux sociaux l'utilisent par exemple pour personnaliser le fil d'actualité et cibler les annonces. Prenons l'exemple du Facebook, il prend en compte les amis avec lesquels les utilisateurs se connectent, les profils consultés le plus souvent, leurs centres d'intérêt…etc.

Basé sur un apprentissage continu, une liste d'utilisations sera suggérée. Facebook utilise également la reconnaissance de visage basée sur l'apprentissage automatique, qui est aussi adaptée dans les caméras de surveillance. Cette technologie est également utilisée par les moteurs de recherche comme Google pour affiner les résultats ; à chaque recherche effectuée, les algorithmes situés au niveau du serveur surveillent la manière dont vous répondez aux résultats. Si vous restez longtemps sur une page web, le moteur de recherche suppose que les résultats affichées correspondent à la requête. En outre, Paypal, entreprise qui offre des services de paiement en ligne, utilise l'apprentissage automatique dans la détection des fraudes. Cette société se protège contre le blanchiment d'argent en utilisant un ensemble d'outils capable de

comparer des millions de transactions en cours et d'identifier les transactions légitimes ou illégitimes entre acheteurs et vendeurs [2].

Le principe du Machine Learning (ML) consiste à donner aux ordinateurs des valeurs d'entrées à analyser et le résultat qu'on veut atteindre pour que la machine puisse créer un modèle capable de déduire de nouvelles informations à partir des anciennes données. Les analystes visent à obtenir un programme précis pouvant être exécuté par un ordinateur afin de résoudre un problème ou une tâche bien définie. Les utilisateurs du Machine Learning appelés « Data Scientists » n'essayent pas d'écrire un programme par eux même, au lieu de cela, ils collectent les données d'entrées et les valeurs ciblées souhaitées. Ensuite, ils demandent à l'ordinateur de trouver un programme (ou un algorithme) qui calcule une sortie pour chaque valeur d'entrée ajoutée.

Le ML est appliqué sur des problèmes compliqués que l'être humain est incapable de résoudre. Mais le Data Scientist doit être toujours prêt et capable de tester quelques approches avant de choisir une approche satisfaisante pour obtenir un meilleur modèle.

L'apprentissage automatique peut être appliqué par deux manières. La première, par apprentissage supervisé qui utilise des algorithmes basés sur des données d'entrées étiquetées avec les sorties souhaitées ; si on dispose d'un ensemble d'objets et pour chaque objet une valeur cible associée, il faut apprendre un modèle capable de prédire la bonne valeur cible d'un nouveau objet non étiqueté. La deuxième, l'apprentissage non supervisé où les données sont non étiquetées, c'est-à-dire quelles ne possèdent aucune valeur cible. L'algorithme dans ce cas trouve tout seul des points communs entre ses données d'entrées, ensuite il les classe en groupes de données aussi distincts que possible en maximisant leur homogénéité ; chaque groupe possède des données ayant des caractéristiques similaires. Plus le nombre de données d'entrées est grand plus on s'approche de trouver une meilleure classification ou groupement en choisissant les bonnes caractéristiques [3].

La science des matériaux fait appel à des méthodes numériques pour gérer la complexité des problèmes posés dans tous les domaines de la technologie. Les chercheurs du monde universitaire et de l'industrie recherchent de nouveaux matériaux de haute qualité pour répondre aux besoins d'applications spécifiques. Dans ce cadre, l'apprentissage automatique est devenu une technologie majeure pour l'indentification des modèles décrivant le comportement

des molécules et les phénomènes physiques. ML a été appliqué pour la découverte de nouveaux matériaux et la prédiction de nouvelles propriétés en passant par des calculs quantiques et statistiques, ce qui a donné de nombreux résultats remarquables. En outre, il est également utilisé pour résoudre des problèmes liés à la science des matériaux qui nécessitent un grand temps de calculs et des expériences qui peuvent être couteuse et difficiles à réaliser.

Ce mémoire est divisé en trois chapitres principaux :

Dans le premier chapitre, nous introduisons les algorithmes les plus utilisés dans un processus du Machine Learning (apprentissage statistique) quel que soit le problème posé.

Dans le deuxième chapitre, nous citons quelques bases de données qui existent dans le domaine de la science des matériaux, chacune d'entre elles contient des propriétés spécifiques. De plus, nous mentionnons la bibliothèque Matminer du langage python avec laquelle on pourra extraire les données pertinentes de nos bases de données choisies, ainsi que la bibliothèque Scikit-Learn que nous utiliserons dans notre processus d'apprentissage automatique.

Dans le troisième chapitre, trois applications seront dédiées à l'apprentissage automatique. La première et la deuxième partie consistent à créer un modèle de prédiction à partir d'une base de donnée bien traitée. Enfin, La dernière partie sera une étude statistique sur un ensemble de données contenant les oxydes, les nitrites et les matériaux à base de cuivre.

CHAPITRE I

Algorithmes ML

1 Introduction :

Il existe plusieurs méthodes en apprentissage automatiques que ce soit pour la régression ou la classification. Pour bien choisir une méthode il faut comprendre les fondements de ces méthodes existantes et de ce qui permet de les distinguer afin de déterminer les modèles qui traiteraient au mieux un problème particulier.

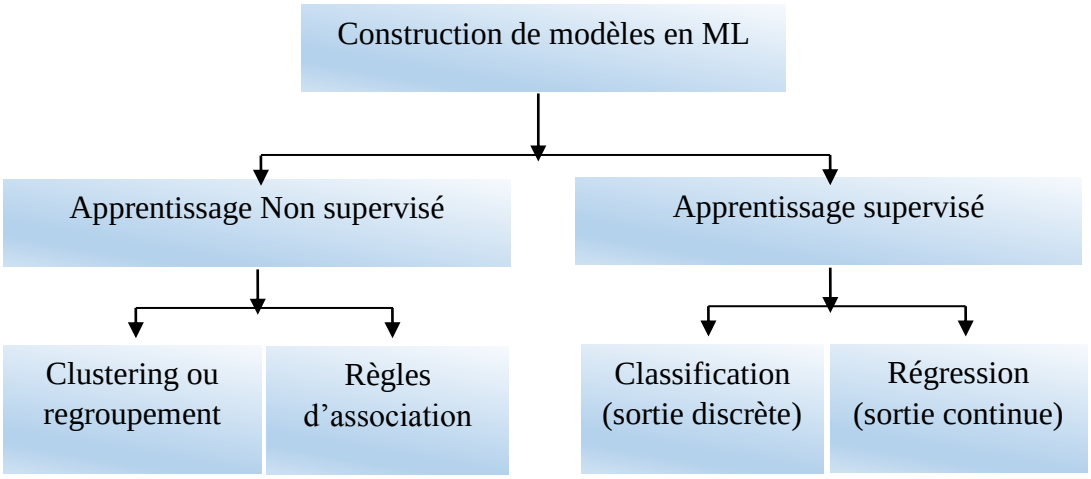


Figure I-1 : Schéma des différentes techniques issues de l'apprentissage automatique pour la construction de modèles de données [4]

2 Algorithmes d'apprentissage automatique :

2.1 Apprentissage supervisé :

L'apprentissage supervisé consiste à utiliser un ensemble de données pour prédire des événements futurs statistiquement probables, c'est-à-dire qu'il frome un modèle de prédiction à partir des évènements déjà prédits auparavant. Les modèles de ML peuvent être utilisées dans des applications de « prédiction » ou de « classification ». On distingue deux types de problèmes d'apprentissage supervisé :

2.1.1 Classification :

Les méthodes de classification s'appliquent lorsque l'ensemble des valeurs résultats est discret. Ceci revient à attribuer une classe (aussi appelée étiquette ou label) pour chaque

valeur d'entrée. Les techniques de classification peuvent être basées sur des hypothèses probabilistes (exemple, naïf bayésien), des notions de proximité (exemple, *k* plus proches voisins) ou des recherches dans des espaces d'hypothèses (exemple, arbres de décisions) [4]. Le choix de la technique convenable est important ; il faut pouvoir choisir la méthode la plus adapté qui sera capable de séparer au mieux les données d'apprentissage.

2.1.1.1 Naïf Bayésien :

La classification naïve bayésienne repose sur l'hypothèse que toutes les caractéristiques sont conditionnellement indépendantes les unes des autres. Cette méthode est basée sur le théorème de Bayes qui calcule la probabilité d'un événement à l'aide de la connaissance au préalable des conditions connexes. Ce théorème a été découvert par un statisticien anglais, Thomas Bayes, au 18ème siècle mai il n'a jamais publié son travail. Après son décès, ses notes ont été éditées et publiées par le mathématicien Richard Price [6]. Le théorème est donné par la formule suivante :

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad \text{(I-1)}$$

- A et B sont des événements.
- $P(A)$ est la probabilité d'observer l'événement A
- $P\ (B)$ est la probabilité d'observer l'événement B.
- $P\ (A \mid B)$ est la probabilité conditionnelle d'observer A, sachant qu'un autre événement B de probabilité non nulle s'est réalisé.

Dans un problème de classification, notre tâche est de trouver l'étiquette la plus probable A, étant donné les caractéristiques B, le théoreme de Bayes devient :

$$P(y|x_1, \dots, x_n) = \frac{P(x_1,\dots,x_n|y)P(y)}{P(x_1,\dots,x_n)} \quad \text{(I-2)}$$

Où n représente le nombre de caractéristiques, y est l'évènement qu'on cherche à classer.

Par conséquent, en tenant compte de l'hypothèse d'indépendance, Bayes prédit la classe qui constitue la probabilité la plus élevée [6] :

$$\hat{y} = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y) \quad \text{(I-3)}$$

2.1.1.2 *k*-plus proches voisins :

Parmi les algorithmes d'apprentissage automatique les plus basiques, le *k*-plus proche voisin, souvent abrégé en *k*-NN où *k* est un entier positif. En Data Science, cet algorithme est largement utilisé pour les problèmes de classification des données. Mais avant de se lancer dans cette méthode, il faut savoir que les calculs peuvent s'avérer très couteuses en temps de calcul, ainsi, les données doivent être prétraitées. Cette méthode peut être également utilisée dans les problèmes de régression.

Prenons par exemple le problème de classification suivant : Dans le diagramme ci-dessous, il y a des objets ronds verts et des objets carrés bleus. Ceux-ci appartiennent à deux classes différentes : la classe des ronds et la classe des carrés. Lorsqu'un nouvel objet est inséré dans l'espace - dans ce cas, un cercle rouge - nous voulons que l'algorithme d'apprentissage automatique classe le cercle dans une certaine classe [3].

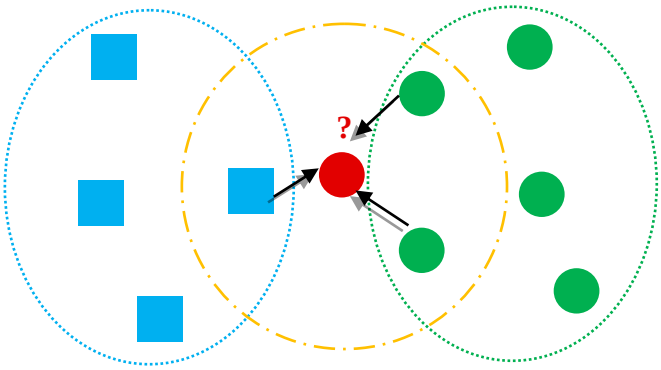


Figure I.2 : exemple de classification par la méthode des k-NN

Si on choisit *k* = 3, l'algorithme cherche les trois plus proches voisins du cercle rouge pour pouvoir le classé soit dans la classe des cercles, soit dans la classe des carrés. Dans ce cas, les trois plus proches voisins du cercle rouge sont un carré et deux cercles. Par conséquent, l'algorithme classera la sphère dans la classe des cercles.

Dans la méthode de *k*-NN, le résultat est l'appartenance à une classe. L'algorithme stocke tous les cas disponibles et classe tout nouvel objet en vérifiant ses *k*-plus proches voisins. Ensuite, l'objet est attribué à la classe avec laquelle il a le plus en commun [3].

2.1.1.3 Arbres de décision :

Les arbres de décision (AD) sont un outil de classification très utilisé. Son principe repose sur la construction d'un arbre de taille limitée [4]. Considérons l'exemple simple représenté ci-dessous :

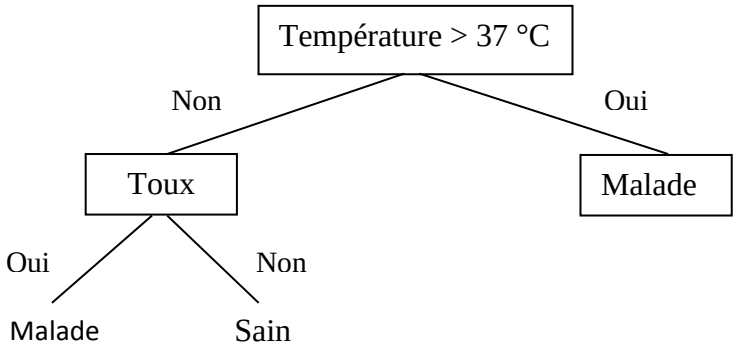


Figure I.3 : schéma d'un arbre de décision [4]

Dans cet exemple, le sommet « Température > 37 °C » représente la racine. Les feuilles correspondent aux classes ou décisions (appelées aussi nœuds terminales), ici « malade » ou « sain ». De plus, on appelle « Température > 37 °C » et « Toux » les attributs ou les variables (appelés aussi nœuds intermédiaires) [4].

Les AD jouent un rôle très important dans ML. Ils sont capables de gérer les variables continues et discrètes et fournissent par la suite une indication claire pour la prédiction ou la classification sans effectuer beaucoup de calcul. Les AD sont si simples à comprendre et à interpréter, qu'ils permettent une meilleure approximation quel que soit la complexité des données. L'arbre le plus simple est souvent le meilleur, à condition que tous les autres arbres possibles produisent les mêmes résultats. Leur construction se réalise en divisant l'arbre du sommet vers les feuilles (du haut vers le bas) en choisissant à chaque étape une variable de

séparation sur un nœud d'où les critères de segmentation. Pour cela, les algorithmes les plus utilisés sont « la classification et arbre de régression » et « le Dichotomiseur itératif 3 » [6].

2.1.1.3.1 La classification et arbre de régression (CART) :

Il a été implémenté par Leo Breiman en 1984 [5]. L'AD générée par CART est binaire c'est-à-dire que la variable peut prendre deux valeurs seulement (oui ou non). L'algorithme de CART utilise l'indice de diversité de Gini pour évaluer les divisions dans l'ensemble de données. Cet indice mesure la fréquence avec laquelle un élément aléatoire de l'ensemble serait mal classé si son étiquette était choisie aléatoirement. L'indice de Gini est donné par l'équation suivante, où c est le nombre de classes, p est le sous-ensemble d'occurrences du nœud et $P(k|p)$ est la probabilité de sélectionner un élément de la classe c dans le sous-ensemble du nœud [6] :

$$Gini(p) = 1 - \sum_{K=1}^{c} P^2(K|p) \quad \text{(I-4)}$$

$$P(K|p) = \frac{N(K|p)}{N(p)}$$

$-\ N(p)$: nombre totale d'éléments.

$-\ N(K|p)$: nombre d'éléments appartenant à la classe K.

2.1.1.3.2 Le Dichotomiseur itératif 3 (ID3) :

L'algorithme a été développé par Ross Quinlan et publié en 1986. Il fait intervenir l'entropie de Shannon qui correspond à la quantité d'information délivrée. L'objectif est de construire un AD à partir d'un ensemble de données constituant les attributs. Nous devons d'abord avoir un nœud racine, alors il faut déterminer l'attribut qui classe le mieux les données d'apprentissage, qui correspond au gain d'informations le plus élevé. Nous pouvons calculer le degré d'incertitude en utilisant l'entropie. Ce dernier est donné par l'équation suivante, où n est le nombre de résultats possible et $P(xi)$ la probabilité d'avoir la classe de position i. Les valeurs communes pour b sont *2*, *e* et *10*, car le résultat souhaité doit être positif [6].

$$H(X) = -\sum_{i=1}^{n} P(x_i) log_b P(x_i) \quad \text{(I-5)}$$

$-$ X : ensemble de données pour lequel l'entropie est calculée.

$-$ x_i : ensemble de classes en X (par exemple, $x_i = \{oui, non\}$)

Quand $H\ (X) = 0$, l'ensemble X est parfaitement classifié (tous les éléments de X sont de la même classe).

L'entropie est calculée pour chaque attribut. Celui ayant l'entropie le plus petit est utilisé pour diviser l'ensemble. Ensuite, on passe au calcul du gain, donné par la formule suivante où T représente tous les sous-ensembles, $H(S)$ est l'entropie du nœud racine et A représente l'attribut pour lequel le gain est calculé.

$$IG(A, S) = H(S) - \sum_{t \in T} P(t)\ H(t) \qquad \text{(I-6)}$$

L'attribut de gain le plus élevé sera choisi comme racine de notre arbre de décision. Ce processus se répète jusqu'à ce que tous les nœuds soient remplis par les attributs les plus convenables pour, enfin, former l'AD voulu.

2.1.1.4 Réseaux de neurones artificiels :

Le terme réseau de neurones est une référence à la neurobiologie. Originalement, ce concept est inspiré du fonctionnement des neurones du cerveau humain, apprend essentiellement de l'expérience [7].

Le fonctionnement exact du cerveau humain est encore un mystère, mais certains aspects sont connus. En particulier l'élément fondamental du cerveau est un type spécifique de cellule incapable de se régénérer. Ces cellules nous fournissent la capacité de nous rappeler, de penser et de réagir en basant sur des expériences antérieures. Le corps humain comporte 100 milliards de ces cellules appelées neurones. Chacun de ces neurones se connecte à 200 milles autres neurones. Les réseaux de neurones artificiels tentent de reproduire que les éléments les plus fondamentaux de cet organisme complexe et puissant [8].

Un réseau de neurones est une organisation hiérarchique de neurones connectés entre eux. Ces derniers transmettent un message ou un signal à d'autres neurones en fonction des paramètres d'entrés reçus et forment un réseau complexe [9]. Ci-dessous une représentation simpliste d'un réseau de neurones de base :

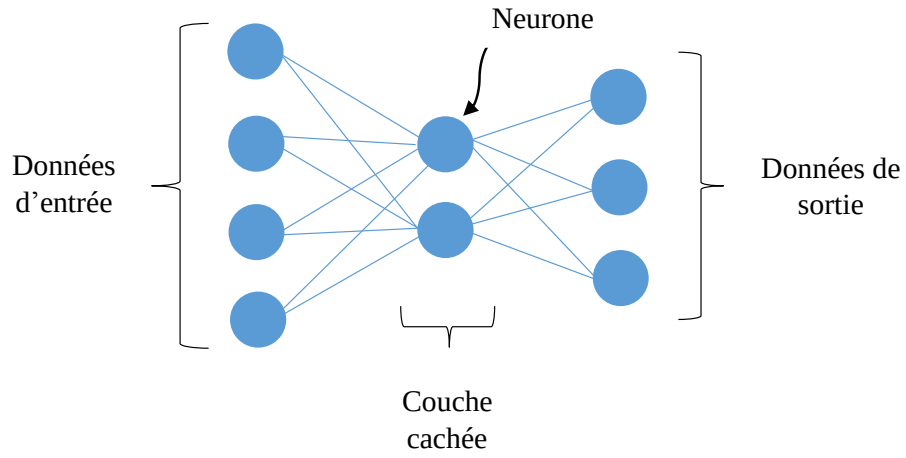


Figure I.4 : Exemple simple d'un réseau de neurones artificiel.

Un réseau de neurones est en général composé d'une succession de couches (Layer en anglais). Ce modèle comporte trois ensembles de règles : multiplication, sommation et activation. Les données d'entrées (Input data en anglais) sont consommées par les neurones de la première couche cachée (Hidden layers en anglais). Chaque couche peut avoir un ou plusieurs neurones. La connexion entre deux neurones de couches successives aurait un poids associé qui définit l'influence de l'entrée sur la sortie du neurone suivant et éventuellement sur la sortie finale globale. Sur chaque neurone artificiel, chaque valeur d'entrée est multipliée par le poids correspondant [7] [9]. Ensuite, le résultat obtenu sera additionné à ce qu'on appelle le biais pour lui appliquer à la fin une fonction d'activation et la valeur de sortie sera donc de la forme :

$$y = F\left(\sum_{i=0}^{m} \omega_i . x_i + b\right)$$

- ω_i représentent les poids.
- x_i représentent les données d'entrées.
- b représente le biais.
- F représente la fonction d'activation.
- y représente la valeur de sortie.

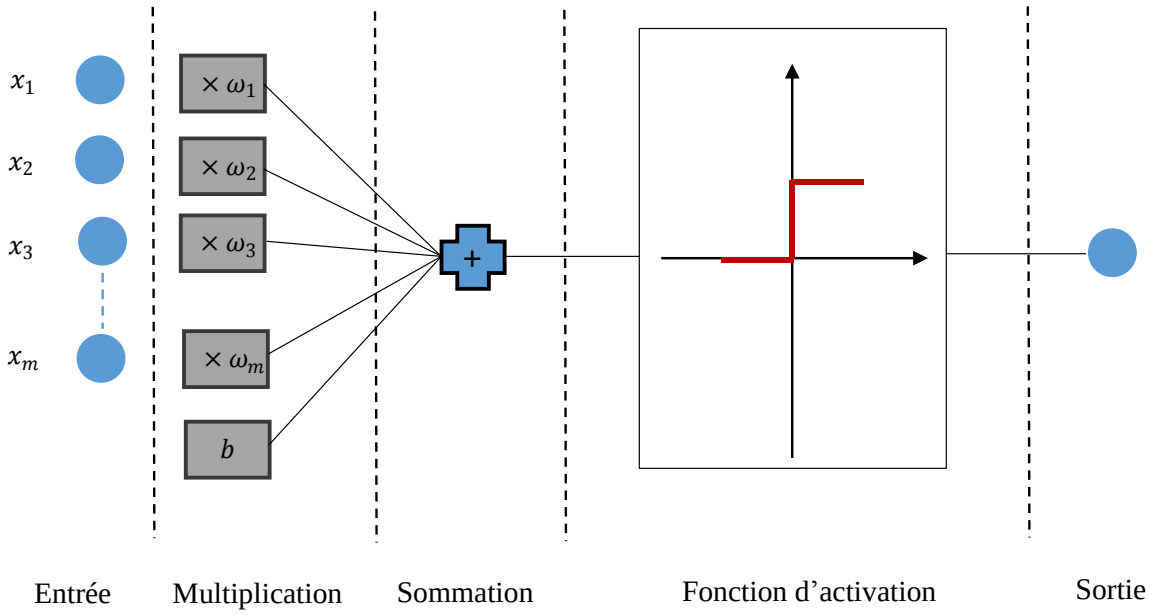


Figure I.5 : Principe de fonctionnement d'un neurone artificiel

Dans un réseau de neurones, les poids initiaux seraient tous aléatoires lors de la formation du modèle, l'inconnu principal est sa fonction de transfert, elle peut être toute fonction mathématique. Le choix de cette fonction dépend du problème posé. Dans la majorité des cas, elle est choisie non linéaire. Cela permet aux réseaux de neurones d'apprendre des transformations non linéaires et complexes des données. Les fonctions d'activation les plus utilisées sont :

- Fonction Heaviside : $f(x) = \begin{cases} 0 & si \quad x < 0 \\ 1 & si \quad x \geq 0 \end{cases}$
- Fonction de la tangente hyperbolique $f(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$
- Fonction sigmoïde $f(x) = \frac{1}{1+e^{-x}}$
- Fonction rampe $f(x) = \frac{x+|x|}{2}$

2.1.2 Régression :

Les méthodes de régression s'appliquent lorsque le résultat que l'on cherche à estimer est une valeur continue. En ML, la régression est un outil important de l'apprentissage supervisé pour

la modélisation et l'analyse des données. Elle est notamment utilisée en statistique et en économie [10].

2.1.2.1 Régression linéaire :

On appelle modèle de régression tout modèle capable à établir une relation linéaire entre une variable, dite expliqué ou dépendante, et une ou plusieurs variables, dite explicatives ou variables indépendantes. Le but principal c'est d'ajuster une meilleure droite représentée par une équation linéaire $Y = f(X) + \varepsilon$ afin de prédire $\hat{Y} = \hat{f}(X)$ pour une valeur de X quelconque [11].

$-$ Y représente les variables dépendantes.

$-$ X représente les variables indépendantes.

$-$ ε représente le terme d'erreur ou perturbation.

La régression linéaire est principalement divisée en deux catégories : la régression linéaire simple et la régression linéaire multiple. La régression linéaire simple est caractérisée par une variable indépendante. Par contre, la régression linéaire multiple est caractérisée par au moins de deux variables indépendantes.

En statistique, la méthode classique est donnée par les Moindres Carrés Ordinaires pour la résolution des problèmes de régression simple ; le meilleur estimateur linéaire d'après le théorème de Gauss-Markov, consiste à calculer la somme des carrés des résidus que l'on notera par $scr\ (f)$:

$$scr(f) = \sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2 = \sum_{i=1}^{n} \varepsilon_i \qquad \text{(I-7)}$$

Dans le cas de la régression linéaire simple, la fonction *f* est un polynôme de degré 1 de X, ce qui donne :

$$scr(f) = scr(a, b) = \sum_{i=1}^{n} \big(y_i - (a + bX)\big)^2 \qquad \text{(I-8)}$$

$\mathbb{P} = \{a, b\}$ est l'ensemble des paramètres du modèle et on cherche les estimations $\hat{a}$ et $\hat{b}$ qui minimisent $scr(f)$. Il faut déterminer les points critiques, solutions des équations normales qui annulent leurs dérivées premières. On obtient une solution analytique :

$$\hat{a} = \bar{y} - \hat{b}\bar{x} \qquad et \qquad \hat{b} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \hat{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{I-9}$$

Avec $$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Le modèle de prédiction est alors donné par :

$$\hat{f}(X) = \hat{a} + \hat{b}X$$

Il existe ainsi la régression non linéaire, c'est une autre forme d'analyse dans laquelle les données sont modélisées par une fonction d'une ou de plusieurs variables indépendantes, qui est une combinaison non linéaire des paramètres du modèle.

2.1.2.2 Régression logistique :

La régression logistique a été développée par le statisticien David Cox en 1958 [12]. Le modèle logistique est un modèle statistique qui utilise une fonction logistique, il est également utilisé dans les problèmes de classification. La régression logistique ne nécessite pas de relation linéaire entre les variables dépendantes et indépendantes, mais elle nécessite des échantillons de grande taille afin d'avoir plus de précision lors de l'estimation de la vraisemblance.

La régression logistique peut être binaire ou multinomiale. La régression logistique binaire traite des situations dans lesquelles le résultat observé pour une variable dépendante ne peut avoir que deux types possibles, par exemple « malade » ou « sain », ces deux possibilités sont étiquetées par « 0 » et « 1 ». La régression logistique multinomiale concerne les situations dans lesquelles le résultat peut avoir trois types possibles ou plus qui ne sont pas ordonnés, par exemple « maladie A » par rapport à « maladie B » par rapport à « maladie C ».

La régression logistique cherche à :

- modéliser la probabilité qu'un événement se produise en fonction des valeurs des variables indépendantes, qui peuvent être catégoriques ou numériques.
- estimer la probabilité qu'un événement se produise pour une observation choisie au hasard par rapport à la probabilité que l'événement ne se produise pas.
- prédire l'effet d'une série de variables sur une variable à réponse binaire.

- classer les observations en estimant la probabilité qu'une observation soit dans une catégorie particulière.

La probabilité à estimer au cours d'une régression logistique est représenté par l'inverse de la fonction « *logit*. ». Cette dernière est équivalente à une fonction linéaire des variables indépendantes, donnée par la formule suivante :

$$Logit(p) = ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 \quad \text{(I-10)}$$

$$p \in]0,1[$$

Avec $$\frac{p}{1-p} = odds(\omega) = \frac{probabilité\ d'existance}{probabilité\ de\ non\ existance}$$

$-$ ω étant comme variable ou évènement.

$-$ β_i sont des coefficients à calculer.

La fonction $logit\ (p) \epsilon]-\infty, +\infty[$ alors que $p \in]0,1[$ c'est ce qui nous ramène à utilise l'inverse du $logit$ qui correspond à la probabilité à estimer, représenté par :

$$logit^{-1}(\alpha) = \frac{1}{1+e^{-\alpha}} \quad \text{(I-11)}$$

α est une combinaison linéaire de variables et leurs coefficients.

Cette méthode consiste à développer une équation de régression qui modélise le $logit$ inverse et utilise les coefficients calculés à l'aide de l'estimateur maximum de vraisemblance.

Ce qu'on essaie de réaliser par la suite et de faire sortir la probabilité p dans l'équation (I.10), ce qui donne après quelques simplifications algébriques :

$$\hat{p} = \frac{e^{\beta_0+\beta_1 x_1}}{1+e^{\beta_0+\beta_1 x_1}} \quad \text{(I.12)}$$

$-$ $\hat{p}$ correspond à l'équation de régression estimée par régression logistique.

2.2 Apprentissage non supervisé :

Dans l'apprentissage non supervisé il n'y a pas de valeurs de sortie, il s'agit de trouver des structures cachées à partir d'un ensemble de données qui doivent être regroupé d'où le terme «clustering ». Le but de ce type d'apprentissage est de séparer les données en groupes ou en catégories.

2.2.1 Clustering :

Le clustering est une technique d'apprentissage automatique non supervisé, utilisé pour le regroupement des données non étiquetées dans de nombreux domaines. Si on dispose d'un nombre fini de points de données et on cherche à les classer dans des groupe de sorte que chaque groupe contient des points de données ayant des propriétés et/ou caractéristiques similaires. Le problème principal qui se pose dans ces algorithmes c'est le choix des propriétés à prendre en compte au cours du regroupement [13]. L'un des algorithmes de clustering les plus utilisés est le « *K*-Means ».

2.2.1.1 *K*-Means :

C'est l'algorithme de classification le plus connu. Son principe est simple, facile à comprendre et à implémenter dans un code. Tout d'abord, on sélectionne un certain nombre de groupes puis, aléatoirement, on initialise le centre associée à chaque groupe. Il est préférable de commencer par analyser globalement les données présentes et essayer d'identifier des groupes distincts afin de mieux déterminer le nombre de classes à utiliser.

Chaque point est classé en calculant la distance entre ce point et le centre de chaque groupe. Par conséquent, le point sera déplacé vers le groupe dont le centre est le plus proche. Pour chaque classe, un nouveau centre est calculé comme étant la moyenne de tous les points de ce groupe. Après chaque itération, les centres se déplacent lentement et la distance totale entre chaque point et le centre qui lui est attribué devient de plus en plus petite. Les étapes ci-dessus seront répétées pour un nombre défini d'itérations ou jusqu'à ce que les centres de groupe ne changent plus. *K*-Means rassure la convergence vers un optimum local. Cependant, cela ne doit pas nécessairement être la meilleure solution globale (optimum global), pour cette raison, on peut initialiser plusieurs fois les centres de groupe de manière aléatoire, puis sélectionner le cycle qui donne les meilleurs résultats [14].

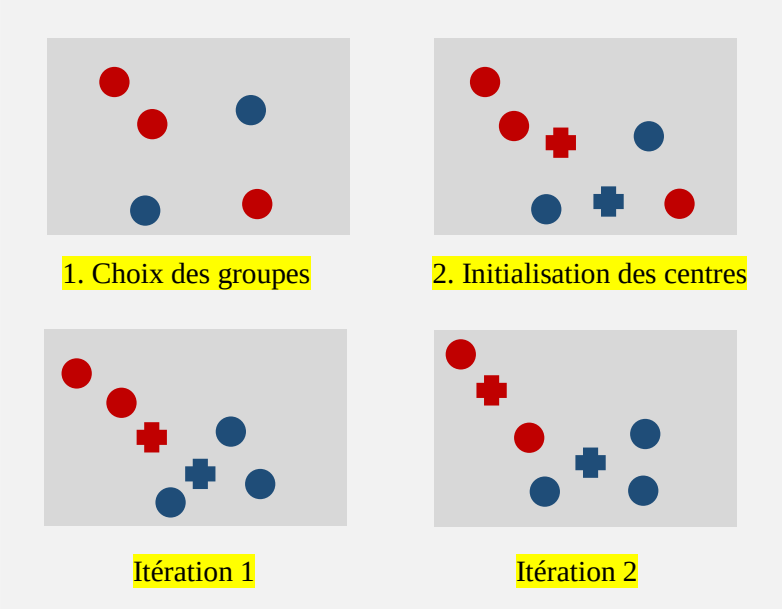


Figure I.6 : Exemple de classification par la technique des *K*-Means avec K=2 [14]

2.2.1.2 T-distributed stochastic neighbor embedding (T-SNE):

C'est une technique linéaire non supervisée, développée par Laurens Van der Maatens et Geoffrey Hinton en 2008. T-SNE est une méthode de réduction de dimension, elle transforme la représentation de données multidimensionnelles en deux ou trois dimensions et donne, par conséquent, une idée sur la façon dont les données sont disposées dans un espace de grande dimension. T-SNE trouve des modèles à partir des données en identifiant des groupes (clusters) contenant les données qui partagent des caractéristiques similaires. Mais ce n'est pas entièrement un algorithme de clustering, il s'agit essentiellement d'une technique d'exploration et la visualisation de données de grande dimension [15].

T-SNE convertit les distances euclidiennes de grande dimension entre les points de données en probabilités conditionnelles représentant des caractéristiques similaires. La similarité du point de donnée x_j avec le point de donnée x_i est la probabilité conditionnelle $p_{j|i}$ que x_i choisirait x_j comme voisin si les voisins étaient choisis par rapport à leur densité de probabilité. Pour les points de données proches, $p_{j|i}$ est relativement élevé et vice versa [16]. Mathématiquement, la probabilité conditionnelle $p_{j|i}$ est donnée par :

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 \,/\, 2\sigma_i^2)}{\sum_{K \neq i} \exp(-\|x_i - x_K\|^2 \,/\, 2\sigma_i^2)} \quad \text{(I-13)}$$

Où σ_i est la variance de la gaussienne centrée sur le point de donnée x_i. Pour les points obtenus dans l'espace de dimension réduite y_i et y_j, il est possible de calculer une probabilité conditionnelle similaire, notée $q_{j|i}$. Dans T-SNE, une distribution de « t-Student » (avec un degré de liberté, identique à une distribution de Cauchy) est utilisée pour mesurer la similarité entre des points dans l'espace de dimension réduite afin de permettre à des objets dissemblables d'être modélisés [16]. Plus précisément $q_{j|i}$ est définie par :

$$q_{j|i} = \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_{K \neq l} (1+\|y_i - y_l\|^2)^{-1}} \quad \text{(I-14)}$$

Avec $$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

$p_{i|i}$ et $q_{i|i}$ sont égales à zéro.

Logiquement, les probabilités conditionnelles $p_{i|i}$ et $q_{i|i}$doivent être égales pour une représentation parfaite de la similarité des points de données dans les différents espaces dimensionnels, c'est-à-dire que la différence entre $p_{i|i}$ et $q_{i|i}$ doit être égale à zéro pour une modélisation parfaite dans l'espace à basse dimension. T-SNE tente de minimiser cette différence de probabilité conditionnelle [16].

Les emplacements des points y_i dans l'espace de dimension réduite sont déterminés en minimisant la divergence non symétrique de Kullback-Leibler de la distribution Q, représenté par la formule suivante :

$$KL(P|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \quad \text{(I-15)}$$

Où P représente la distribution de probabilité conditionnelle sur tout l'ensemble des abscisses pour un point x_i donné [16].

$-\ Q$ représente la distribution de probabilité conditionnelle sur tout l'ensemble des ordonnées données pour un point y_i donné [16].

3 Notion de distance :

Afin de mesurer la similarité entre les éléments d'un ensemble de données, le choix de distance joue un rôle très important. Il est nécessaire d'identifier, de quelle manière les données sont liées entre elles, et à quel point les données sont différentes ou similaires les unes des autres [15]. Pour le calcul de distance dans la méthode des *K*-Means, plusieurs notions existent. Cependant, La distance euclidienne donne le meilleur résultat.

3.1 Distance euclidienne :

La distance euclidienne calcule la racine carrée de la différence entre les cordonnées d'une paire d'objets (points ou classes). Si on considère deux points A et B, de cordonnées respectives (X_A, Y_A) et (X_B, Y_B) [15] [17], la distance euclidienne est données par :

$$Dist_{AB} = \sqrt{(X_A - X_B)^2 + (Y_A - Y_B)^2} \quad \text{(I-18)}$$

3.2 Distance de Manhattan :

Si on considère encore deux points A et B, de cordonnées respectives (X_A, Y_A) et (X_B, Y_B) [15] [17], la distance de Manhattan est définie par :

$$Dist_{AB} = |X_B - X_A| + |Y_B - Y_A| \quad \text{(I-17)}$$

3.3 Distance de Tchebychev :

C'est la distance entre deux point données par la différence maximale entre leur cordonnées. Maintenant, on considère deux points A et B, de cordonnées respectives $(X_1, X_2, \dots, X_n)$ et $(Y_1, Y_2, \dots, Y_n)$ [17], la distance de Tchebychev est définie par :

$$Dist_{AB} = \max_{i \in [0,n]} (|X_i - Y_i|) \quad \text{(I-18)}$$

Bibliographies

[1] John McCarthy & Ed Feigenbaum. « Arthur Samuel: Pioneer in Machine Learning». AI Magazine Volume 11-No.3, Association for the Advancement of Artificial Intelligence, 1990.

[2] https://medium.com/app-affairs/9-applications-of-machine-learning-from-day-to-day-life-112a47a429d0

[3] Metomo JOSEPH BERTRAND RAPHAËL. « Machine Learning : Introduction à l'apprentissage automatique ». SUPINFO International University, 10 Octobre 2017.

[4] Mokhtar Taffar. « Initialisation à l'apprentissage automatique ». Université de Jijel, Département Informatique, Algérie. 2014

[5] https://fr.wikipedia.org/wiki/Classification_na%C3%AFve_bay%C3%A9sienne

[6] Gavin Hackeling. « Mastering Machine Learning with scikit-learn Second Edition ». Packt Publishing Ltd, Livery Place, UK, Juin 2017.

[7] Andrej Krenker, Janez Bešter and Andrej Kos. « Introduction to the Artificial Neural Networks, Artificial Neural Networks - Methodological Advances and Biomedical Applications », Prof. Kenji Suzuki (Ed.), ISBN: 978- 953-307-243-2, InTech, 2011.

[8] Dave Anderson and George McNeill. « ARTIFICIAL NEURAL NETWORKS TECHNOLOGY ». Kaman Sciences Corporation 258 Genesse Street Utica, New York 13502-4627, 20 Août 1992.

[9] Jojo Moolayil. « Learn Keras for Deep Neural Networks-A Fast-Track Approach to Modern Deep Learning with Python ». Vancouver, BC, Canada, 2019. DOI : https://doi.org/10.1007/978-1-4842-4240-7

[10] https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/

[11] Julien Ah-Pine. « Apprentissage automatique ». Université Lyon 2. 2018.

[12] https://en.wikipedia.org/wiki/Logistic_regression

[13] https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

[14] https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/

[15] Archana Singh, Avantika Yadav, Ajay Rana. « K-means with Three different Distance Metrics ». Internatioanl Journal of Computer Applications, Volume 67-No.10, Avril 2013.

[16] Laurens van der Maaten, Geoffrey Hinton. « Visualizing Data using t-SNE ». Journal of Machine Learning Research 9, Novembre 2008.

[17] John V. Guttag. « Introduction to Computation and programming Using Python Second Edition ». The MIT Press, Cambridge, Massachusetts, London, England.

CHAPITRE II

Bases de Données et Bibliothèques

1 Base de données en science des matériaux :

La prédiction de nouveaux matériaux nécessite une base de données contenant un nombre important de propriétés physiques des différents éléments chimiques. Pour pouvoir utiliser les nombreuses données des matériaux collectés au cours des dernières années, il est important de développer un système de base de données intelligent, basé sur la technologie d'exploration de données. Il existe des dizaines de bases de données dans le domaine de la science des matériaux.

1.1 AtomWork-Adv (Inorganic Materials Database) :

L'institut national de la science des matériaux NIMS a mis AtomWork-Adv (prononcé AtomWork Advanced), une base de données des matériaux inorganiques de haute qualité, accessible au grand public en tant que service payant depuis le *28* mai *2018*. Par conséquent, les services qu'elle propose telles que la possibilité de copier, télécharger et rechercher des données, sont limitées. Les données rassemblées dans AtomWork-Adv sont collecté à partir de la littérature publiées jusqu'en *2016* [1].

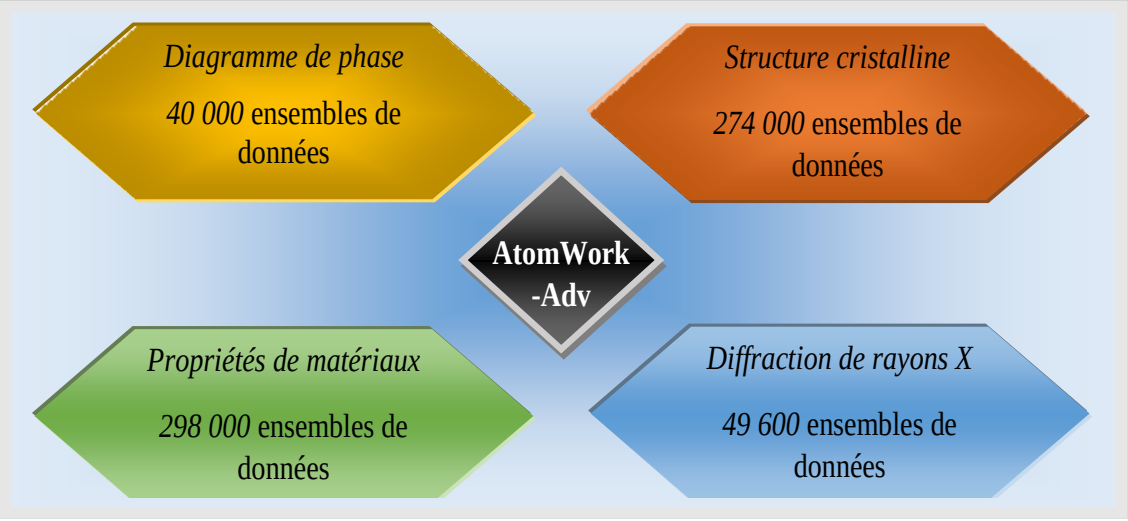


Figure II.1 : Base de données AtomWork-Adv (Mai 2018)

1.2 ICSD (Inorganic Crystal Structure Database) :

Fiz-Karlsruhe offre à la communauté scientifique la plus grande base de données au monde pour les structures de cristaux inorganiques complètement déterminées, y compris leurs

coordonnées atomiques, publié depuis *1913*. La mise à jour la plus récente de cette base de données était le *6* décembre *2018*, contient *203 830* structures cristallines. En particulier, ICSD fournit des informations sur [2] :

- Des données structurelles d'éléments purs, de minéraux, de métaux et de composés intermétalliques.
- Des descripteurs structurels (symbole de Pearson, formule ANX, séquences de Wyckoff).
- Des données bibliographiques et mots-clés sur les méthodes.

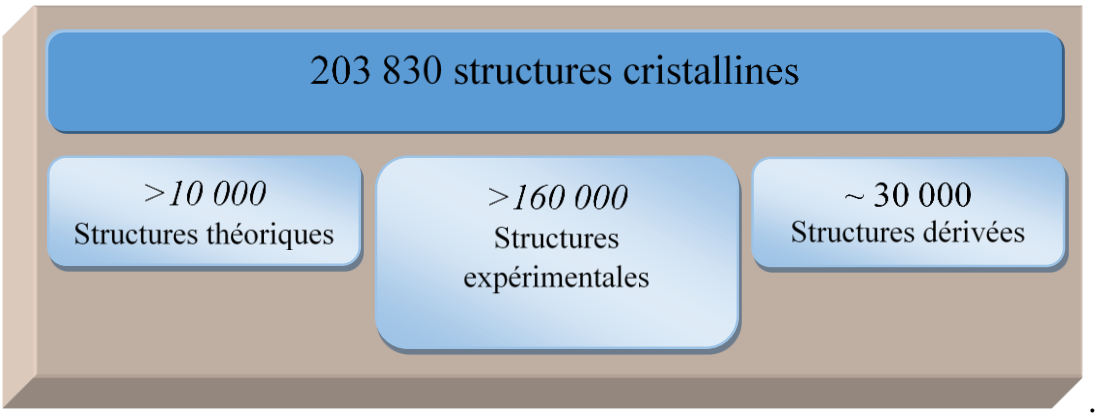


Figure II.2 : Base de données ICSD (Décembre 2018)

1.3 Materials Project :

Créé à l'institut de technologie du Massachusetts MIT, par Kristin Aslaug Persson (Chercheur scientifique au laboratoire national Lawrence Berkeley) et Gerbrand Ceder (professeur en science des matériaux à l'Université de Californie à Berkeley).

Materials Project vise à éliminer les incertitudes liées à la conception des matériaux dans diverses applications. Elle permet aux chercheurs de faire des analyses statistiques sur les propriétés des matériaux représentant l'ensemble des données. Materials Project vise à accélérer l'innovation dans la recherche des nouveaux matériaux [3].

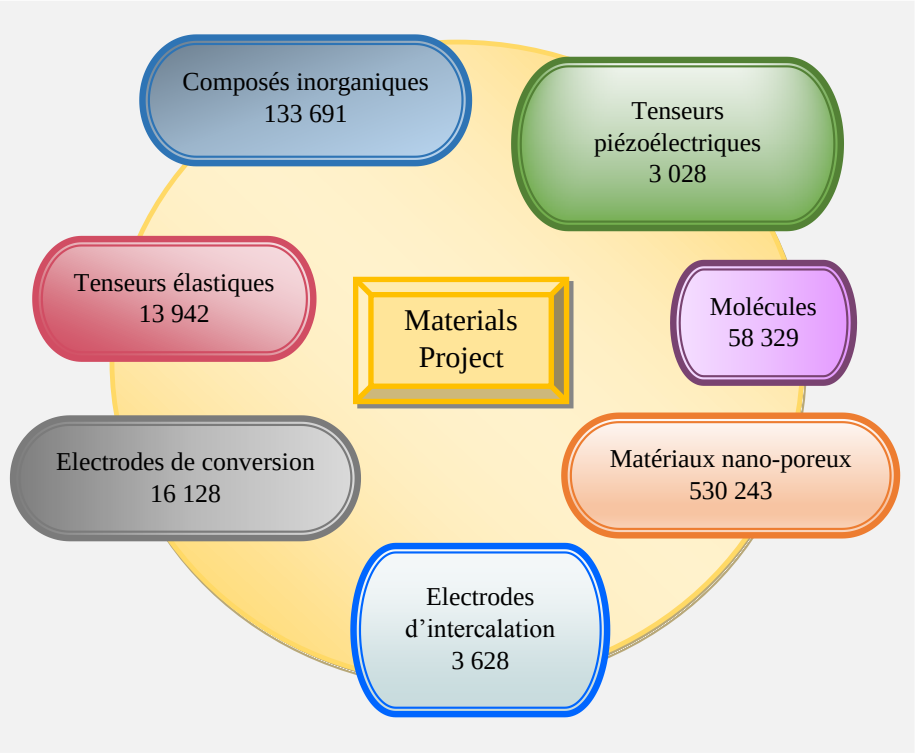


Figure II.3 : Base de données Materials Project (2018)

1.4 OMDB (Organic Materials Database) :

OMDB est une base de données gratuite contenant les structures électroniques des cristaux organiques tridimensionnels et des matériaux organométalliques, développée à l'Institut nordique de physique théorique et contient *22 895* matériaux. OMDB fournit des outils pour les requêtes de recherche basées sur des techniques d'exploration de donnée (data mining en anglais) et d'apprentissage automatique (statistique). Les structures de bandes électroniques sont calculées à l'aide de théorie fonctionnelle de la densité, outil standard de la science des matériaux .

L'interface Web OMDB permet aux utilisateurs de rechercher des matériaux avec des propriétés spécifiques à l'aide de requêtes non triviales sur leur structure électronique, y compris des outils avancés de reconnaissance des formes, de recherche des propriétés chimiques et physiques [4].

1.5 PCD (Pearson's Crystal Data) :

Pearson's Crystal Data est une base de données cristallographique publiée par ASM International (American Society for Metals), éditée par Pierre Villars et Karin Cenzual. PCD découle du projet bien connu PAULING FILE et contient des structures cristallines d'un grand ensemble de matériaux et de composés inorganiques. Les scientifiques ont analysé et traité plus de *103 200* publications originales pour avoir le contenu de PCD présenté dans la figure II.4 [5].

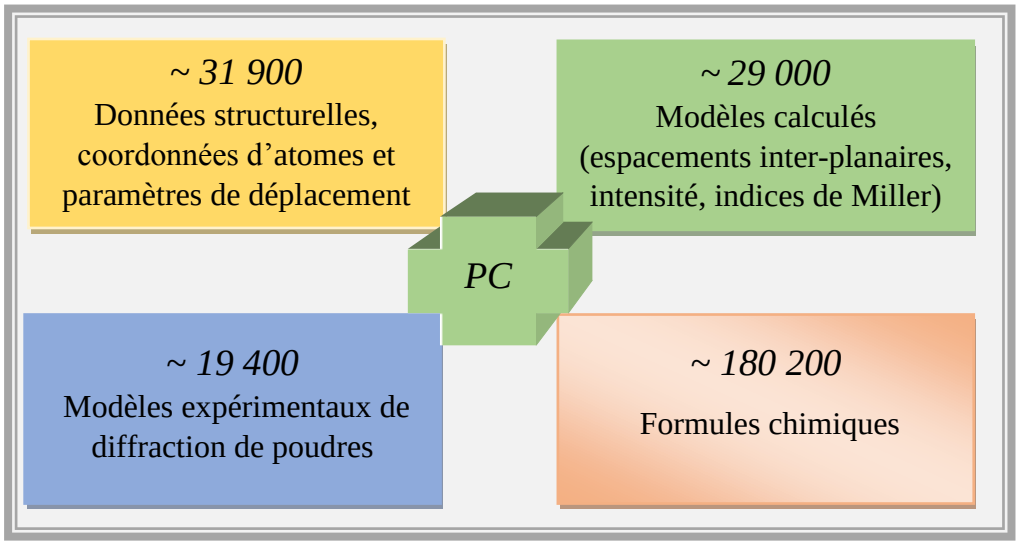


Figure II.4 : Base de données Pearson's Crystal Data (2018/19)

1.6 OQMD (The Open Quantum Materials Database) :

OQMD est une base de données de matériaux inorganique, développé et mis à jour à l'université de Northwestern par des membres du groupe de recherche « Chris Wolverton ». Elle contient environ 300 000 de données calculées par la théorie fonctionnelle de la densité (DFT) de plusieurs structures élémentaires, binaires et ternaires. Environ 10% de ces structures proviennent de la base de données ICSD. Pour chaque élément donné, OQMD contient les énergies totales et de formation, le gap, le moment magnétique, le groupe d'espace et le volume cristallin. Cette base de données a été utilisée par les chercheurs dans la découverte de nouveaux matériaux efficaces pour la fabrication des cellules solaires [6].

1.7 Citrination :

Citrination a été créé par Citrine Informatics dans le but d'avoir un accès facile et simple aux chercheurs sur des données de matériaux. Citrination fournit des outils basés sur l'intelligence artificielle et l'exploration de donnée qui permettent d'extraire de nouvelles informations à partir de données [7] .

Figure II.5 : Base de données Citrination

2 Bibliothèques Python sur le ML:

2.1 Scikit-Learn :

C'est une bibliothèque d'apprentissage statistique dans le langage de programmation python, basée sur d'autres bibliothèques python : NumPy, SciPy et matplotlib. Au début Scikit-learn était un projet « Google summer of code » de David Cournapeau en 2007. En 2010, l'INRIA, l'institut français de recherche en informatique et en automatisme, a commencé à développer ce projet et a publié la première version le 1er février 2010 [8]. Le projet repose aujourd'hui sur un effort mondial en code source ouvert rassemblant plus de 200 contributeurs.

Scikit-learn fournit des algorithmes pour les taches d'apprentissage statistique, notamment la classification, la régression, la réduction de dimension et le clustering. L'application des algorithmes d'apprentissage automatique n'était pas à la portée des utilisateurs qui pourraient

en tirer le plus grand profit : chercheurs biologistes, climatologues, physiciens expérimentaux d'où vient l'intérêt de Scikit-learn grâce à sa simple documentation. De nombreux algorithmes de Scikit-learn sont rapides et adaptables à tous les ensembles de données, les développeurs peuvent donc les adapter aux méthodes choisies en ne modifiant que quelques lignes de code [9].

L'installation de la version stable actuelle de Scikit-learn nécessite :

- Python (>= 2.7 ou >= 3.4),
- Numpy (>= 1.8.2),
- Scipy (>= 0.13.3).

Remarque : Scikit-learn 0.20 est la dernière version compatible avec python 2.7 et python 3.4. Scikit-learn 0.21 nécessite python 3.5+.

L'installation la plus simple sera l'installation pip :

```
pip install -U scikit-learn
```

Cependant, pour les systèmes Linux, il est recommandé d'utiliser le conda pour éviter les processus de génération possibles.

```
conda install scikit-learn
```

Pour vérifier que vous avez Scikit-learn, exécutez en Shell :

```
python -c 'import sklearn; print(sklearn.__version__)'
```

2.2 Matminer :

C'est une bibliothèque dans le langage de programmation python utilisé pour l'exploration de données (data mining) en science des matériaux. Matminer permet aux utilisateurs d'analyser et de gérer des données représentant des propriétés de matériaux car il possède des modules

permettant de récupérer des ensembles de données volumineux à partir de base de données externes tel que « Materials Project » et « Citrination ».

Les rôles principaux de matminer sont décrits dans la figure II.6 : d'abord, matminer récupère des ensembles de données importants à partir de bases de données présentes, ensuite il transforme les données brutes en représentations adaptées à un apprentissage automatique, de plus il produit des visualisations claire et explicatives des données pour pouvoir enfin faire une analyse exploratoire significative. Matminer n'implémente pas lui-même les algorithmes d'apprentissage automatique courants ; Des outils standards de l'industrie comme Scikit-learn ou Keras sont déjà développés pour cette raison. Au lieu de cela, le rôle de matminer est de connecter ces outils avancés d'apprentissage automatique au domaine des matériaux. Matminer implémente une bibliothèque de méthodes de génération d'options «featurizers» pour une grande variété de propriétés des matériaux (compositions, structures cristallines et structures électronique…etc.), ainsi que des outils facilitant la récupération et la visualisation des données. Matminer contient trois composants principaux :

- Récupération des données (data retrieval) : La première étape de l'exploration de données consiste à obtenir un ensemble de données idéalement volumineux et diversifié. Plusieurs efforts sont en cours pour constituer de telles bases de données sur les propriétés des matériaux. Cependant, l'utilisation de ces sources de données est compliquée par le fait que chaque base de données implémente une API et un schéma différent. Une des fonctions essentielles de matminer est de fournir une API cohérente autour de différentes bases de données et de renvoyer leurs contenus sous une forme adaptée à une utilisation dans les outils d'exploration de données [10].
- Caractérisation des données (data featurization) : Cette étape convertit les données d'un format brut en une représentation numérique lisible par les logiciels de visualisation ou d'apprentissage automatique. Matminer contient environ 7O fonctionnalités (Avril 2019) qui prennent en charge la génération de fonctionnalités pour divers types de données de matériaux. Chacune de ces fonctionnalités peut produire de nombreuses caractéristiques individuelles, de sorte qu'il est possible de générer des milliers de caractéristiques totales. avec le code matminer. Par exemple « ElectronegativityDiff » calcule les caractéristiques des différences d'électronégativité entre les anions et les cations [10].

- Visualisation des données (data visualization) : Bien qu'il existe plusieurs excellentes bibliothèques de graphiques en Python (par exemple, matplotlib et seaborn]), ces bibliothèques ne sont pas conçues pour générer des graphiques interactifs faciles à partager et à sérialiser en format de données brutes. Pour accélérer la visualisation, Matminer inclut son propre module, FigRecipes, qui fournit un ensemble de méthodes prédéfinies pour la création de figures communes bien formatées. Plotly a été sélectionné comme l'arrière-plan de FigRecipes. Il contient sept types de tracé : les tracés x-y, les matrices de dispersion, les histogrammes, les diagrammes à barres, les cartes thermiques, les tracés parallèles et les tracés de violon [10].

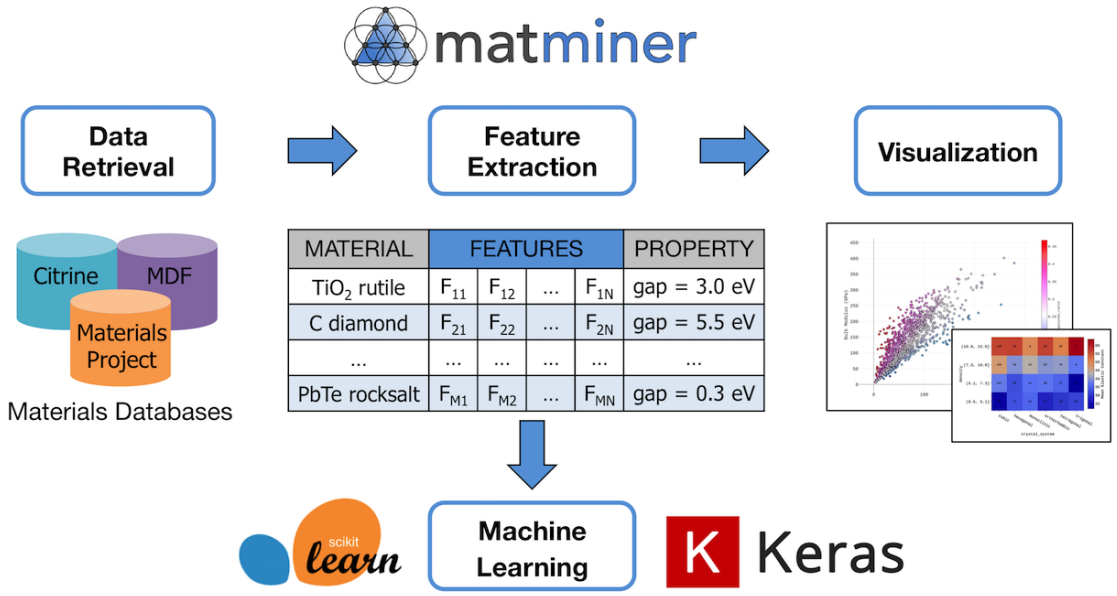


Figure II.6 : Vue d'ensemble des capacités de matminer

L'installation de matminer nécessite :

- Python (>= 3.6),
- Pymatgen,
- Sympy,

L'installation par pip :

```
$ pip install matminer
```

Ou, pour installer matminer dans votre dossier personnel, exécutez la commande suivante :

```
$ pip install matminer --user
```

Pour mettre à jour matminer, tapez simplement .pip install --upgrade matminer [11].

Bibliographies

[1] https://atomwork-adv.nims.go.jp/en/service.html

[2] https://icsd.fiz-karlsruhe.de/index.xhtml;jsessionid=605FB57E2579E65AC850DC43045626BE

[3] https://materialsproject.org/about

[4] https://omdb.mathub.io/

[5] http://www.crystalimpact.com/pcd/

[6] http://oqmd.org/

[7] https://citrination.org/

[8] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel (...) Mathieu Blondel. «Scikit-learn: Machine Learning in python». Journal of Machine Learning Research 12, Octobre 2011.

[9] Gavin Hackeling. «Mastering Machine Learning with scikit-learn Second Edition ». Packt Publishing Ltd, Livery Place, UK, Juin 2017.

[10] Logan Ward, Alexander Dunnc, Alireza Faghaninia, Nils E.R. Zimmermann, Saurabh Bajaj, Qi Wang (...) Joseph Montoya. «Matminer: An open source toolkit for materials data mining». Computational Materials Science Volume 152, Pages 60-69, September 2018. DOI : https://doi.org/10.1016/j.commatsci.2018.05.018

[11] https://pypi.org/project/matminer/0.1.1/

CHAPITRE III

Résultats

Chapitre III Résultats

1 Prédiction du gap expérimental par Machine Learning :

Afin de pouvoir utiliser un matériau dans n'importe quelle application, il est nécessaire de connaitre ses propriétés électroniques qui dépendent du gap. Ce dernier est difficile à calculer par les méthodes classiques. Par conséquent, la prédiction du gap représente un défi pour la communauté scientifique mais faisable par ML.

D'abord, on a rassemblé les énergies de gap expérimentales à partir de Citrine en utilisant la bibliothèque Matminer, et les énergies de gap théoriques à partir de Materials Project en utilisant la bibliothèque Pymatgen. Ensuite, on a appliqué un apprentissage sur notre base de données créée en utilisant la méthode des forêts aléatoires. Cet outil permet non seulement de prédire avec précision le gap expérimental pour n'importe quelle composition, mais la rapidité de la prédiction basée uniquement sur la composition et son gap théorique.

Comparer et tracer les énergies de gap expérimentales à partir de Citrine et les énergies de gap calulculées à partir de MP

Importer les bibliothèques

In [ ]:
```
import numpy as np
import pandas as pd

# Définir les options de vue des pandas
pd.set_option('display.width', 1000)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

# filtrer les messages d'avertissement du notebook
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
from matminer.data_retrieval.retrieve_Citrine import CitrineDataRetrieval

api_key = [redacted] # Définir votre clé API Citrine.
c = CitrineDataRetrieval(api_key) # Créer un adaptateur de la base de données de Citrine.

df = c.get_dataframe(criteria={'data_type': 'EXPERIMENTAL', 'max_results': 1483},
                     properties=['Band gap'],
                     common_fields=['chemicalFormula'])
```

```
100%|[redacted]| 1483/1483 [00:52<00:00, 28.83it/s]
```

```
all available fields:
['Photoluminescence-conditions', 'Color-dataType', 'Thermoluminescence-conditions', 'chemicalFormula', 'Temperature derivative
of band gap-conditions', 'Band gap-conditions', 'Structure', 'Lasing-conditions', 'Mechanical luminescence', 'Synthesis metho
d', 'Cathodoluminescence-conditions', 'Cathodoluminescence-dataType', 'Temperature derivative of band gap-methods', 'Thermolumi
nescence', 'Band gap-units', 'Electroluminescence-conditions', 'Mechanical luminescence-dataType', 'Band gap', 'category', 'ui
d', 'Band gap-dataType', 'Photoluminescence', 'Temperature derivative of band gap-units', 'Temperature derivative of band gap-d
ataType', 'references', 'Morphology', 'Thermoluminescence-dataType', 'Cathodoluminescence', 'Electroluminescence', 'Electrolumi
nescence-dataType', 'Mechanical luminescence-conditions', 'Crystallinity', 'Lasing-dataType', 'Phase_1', 'Photoluminescence-dat
aType', 'Phase', 'Color', 'Temperature derivative of band gap', 'Band gap-methods', 'Phase_2', 'Lasing', 'Color-conditions']

suggested common fields:
['chemicalFormula', 'references', 'Band gap', 'Band gap-conditions', 'Band gap-dataType', 'Band gap-methods', 'Band gap-units',
'Cathodoluminescence', 'Cathodoluminescence-conditions', 'Cathodoluminescence-dataType', 'Color', 'Color-conditions', 'Color-da
taType', 'Crystallinity', 'Electroluminescence', 'Electroluminescence-conditions', 'Electroluminescence-dataType', 'Lasing', 'L
asing-conditions', 'Lasing-dataType', 'Mechanical luminescence', 'Mechanical luminescence-conditions', 'Mechanical luminescence
-dataType', 'Morphology', 'Phase', 'Phase_1', 'Phase_2', 'Photoluminescence', 'Photoluminescence-conditions', 'Photoluminescenc
e-dataType', 'Structure', 'Synthesis method', 'Temperature derivative of band gap', 'Temperature derivative of band gap-conditi
ons', 'Temperature derivative of band gap-dataType', 'Temperature derivative of band gap-methods', 'Temperature derivative of b
and gap-units', 'Thermoluminescence', 'Thermoluminescence-conditions', 'Thermoluminescence-dataType']
```

In [3]:
```
df.rename(columns={'Band gap': 'Experimental band gap'}, inplace=True)
df.head()
```

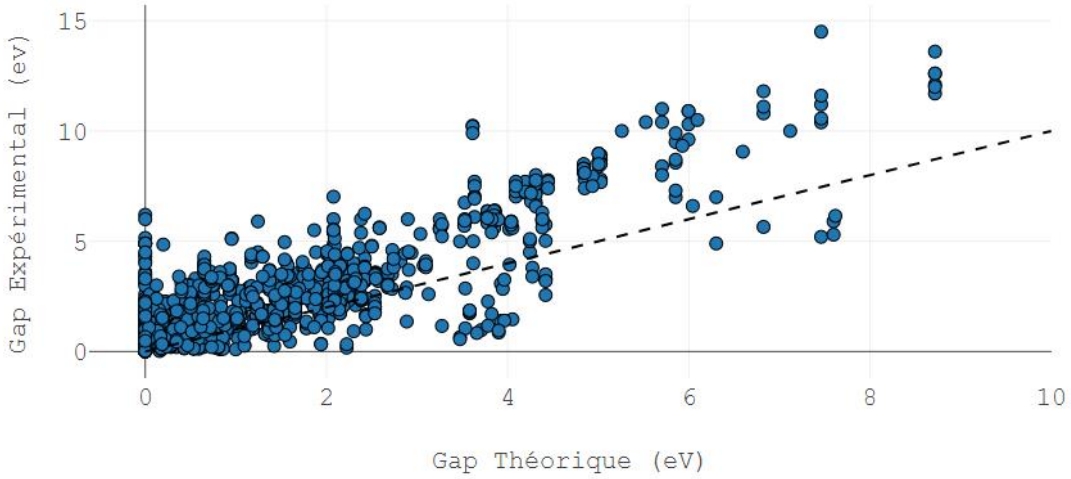
Pour chaque formule chimique avec un gap expérimental, on recherche le gap théorique dans la base de données de Materials Project

In [9]:
```
%%time
from pymatgen import MPRester, Composition
mpr = MPRester()

def get_MP_bandgap(formula):
    """Pour une formule chimique donnée, on recherche l'energie de gap

    Argument:
        composition (string) - Formule Chimique
    Retour:
        (réel) Energie de gap"""

    reduced_formula = Composition(formula).get_integer_formula_and_factor()[0]
    struct_lst = mpr.get_data(reduced_formula)

    if struct_lst:
        return sorted(struct_lst, key=lambda e: e['energy_per_atom'])[0]['band_gap']

df['Computed band gap'] = df['chemicalFormula'].apply(get_MP_bandgap)
```

```
CPU times: user 14.9 s, sys: 1.16 s, total: 16.1 s
Wall time: 13min 36s
```

Tracer la courbe des énergies de gap expérimentales en fonction des énergies de gap théoriques

In [11]:
```
from matminer.figrecipes.plot import PlotlyFig

pf = PlotlyFig(df, x_title='Gap Théorique (eV)',
               y_title='Gap Expérimental (ev)',mode='notebook',
               fontsize=20, ticksize=15)
pf.xy([('Computed band gap', 'Experimental band gap'), ([0, 10], [0, 10])],
      modes=['markers', 'lines'], lines=[{}, {'color': 'black', 'dash': 'dash'}],
      labels='chemicalFormula', showlegends=False)
```



Construire un modèle de Machine Learning

In [12]:
```
df.head(2)
```

Out[12]:

	Experimental band gap	Band gap-conditions	Band gap-dataType	Band gap-methods	Band gap-units	chemicalFormula	Computed band gap
1	0.153	{'name': 'Temperature', 'scalars': [{'value': ...	EXPERIMENTAL	{'name': 'Thermal activation'}	eV	Bi2Te3	0.5336
2	0.567	[{'name': 'Transition', 'scalars': [{'value': ...	EXPERIMENTAL	{'name': 'Absorption'}	eV	Mg2Ge1	0.1689

In [13]:
```
data = df[['Experimental band gap','Computed band gap','chemicalFormula']]
```

In [14]:
```
data = data.rename(columns={'Experimental band gap': 'Gap.Exp', 'Computed band gap':'Gap.Th',
                            'chemicalFormula': 'composition'})
```

In [15]:
```
from matminer.featurizers.conversions import StrToComposition
```

In [16]:
```
data = StrToComposition(target_col_id='composition_obj').featurize_dataframe(data, 'composition')
```

In [17]:
```
for k in ['Gap.Exp']:
    data[k] = pd.to_numeric(data[k])
```

In [18]:
```
original_count = len(data)
data = data[~ data['Gap.Exp'].isnull()]
print('Entrées nulles %d/%d effacées'%(original_count - len(data), original_count))
```

```
Entrées nulles 0/1483 effacées
```

In [20]:
```
from matminer.featurizers import composition as cf
from matminer.featurizers.base import MultipleFeaturizer
```

In [21]:
```
feature_calculators = MultipleFeaturizer([cf.Stoichiometry(), cf.ElementProperty.from_preset("magpie"),
                                          cf.ValenceOrbital(props=['avg']), cf.IonProperty(fast=True)])
```

In [22]:
```
feature_labels = feature_calculators.feature_labels()
```

In [23]:
```
%%time
data = feature_calculators.featurize_dataframe(data, col_id='composition_obj');
```

```
CPU times: user 506 ms, sys: 205 ms, total: 711 ms
Wall time: 3.41 s
```

Chapitre III Résultats

In [24]:
```
print('%d caractéristiques générées'%len(feature_labels))
print("Taille de l'ensemble des Tests:", 'x'.join([str(x) for x in data[feature_labels].shape]))
```

```
145 caractéristiques générées
Taille de l'ensemble des Tests: 1483x145
```

In [25]:
```
original_count = len(data)
data = data[~ data[feature_labels].isnull().any(axis=1)]
print('Entrées nulles %d/%d effacées'%(original_count - len(data), original_count))
```

```
Entrées nulles 0/1483 effacées
```

In [26]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score, cross_val_predict, GridSearchCV, ShuffleSplit, KFold
```

In [27]:
```
model = GridSearchCV(RandomForestRegressor(n_estimators=200, n_jobs=-1),
                     param_grid=dict(max_features=range(8,15)),
                     scoring='neg_mean_squared_error',cv=ShuffleSplit(n_splits=1, test_size=0.1))
```

In [28]:
```
model.fit(data[feature_labels], data['Gap.Exp'])
```

Out[28]:
```
GridSearchCV(cv=ShuffleSplit(n_splits=1, random_state=None, test_size=0.1, train_size=None),
             error_score='raise-deprecating',
             estimator=RandomForestRegressor(bootstrap=True, criterion='mse',
                                             max_depth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators=200, n_jobs=-1,
                                             oob_score=False, random_state=None,
                                             verbose=0, warm_start=False),
             iid='warn', n_jobs=None, param_grid={'max_features': range(8, 15)},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='neg_mean_squared_error', verbose=0)
```

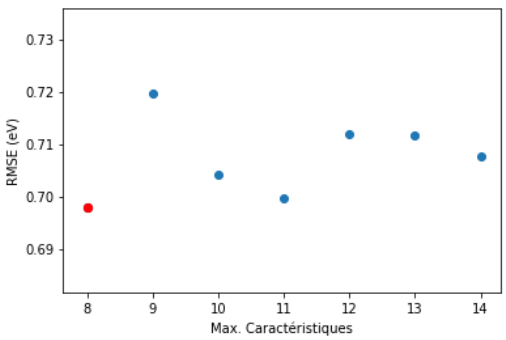
In [29]:
```
model.best_score_
```

Out[29]:
```
-0.4869939648722116
```

In [30]:
```
from matplotlib import pyplot as plt
```

In [31]:
```
fig, ax = plt.subplots()

ax.scatter(model.cv_results_['param_max_features'].data,
           np.sqrt(-1 * model.cv_results_['mean_test_score']))
ax.scatter([model.best_params_['max_features']], np.sqrt([-1*model.best_score_]), marker='o', color='r', s=40)
ax.set_xlabel('Max. Caractéristiques')
ax.set_ylabel('RMSE (eV)')
```

Out[31]:
```
Text(0, 0.5, 'RMSE (eV)')
```



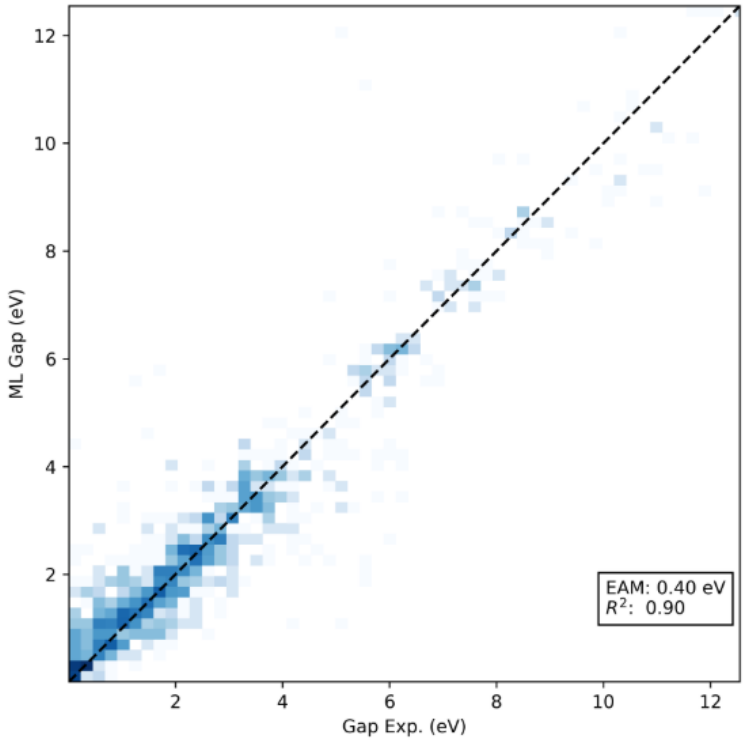
In [32]:
```
from sklearn import metrics
```

In [33]:
```
cv_prediction = cross_val_predict(model, data[feature_labels], data['Gap.Exp'], cv=KFold(40, shuffle=True))
```

In [ ]:
```
for scorer in ['r2_score', 'mean_absolute_error', 'mean_squared_error']:
    score = getattr(metrics,scorer)(data['Gap.Exp'], cv_prediction)
    print(scorer, score)
```

Entrée [30]:
```
from matplotlib.colors import LogNorm
```

Entrée [31]:
```
fig, ax = plt.subplots()

ax.hist2d(pd.to_numeric(data['Gap.Exp']), cv_prediction, norm=LogNorm(), bins=64, cmap='Blues', alpha=1)

ax.set_xlim(ax.get_ylim())
ax.set_ylim(ax.get_xlim())

mae = metrics.mean_absolute_error(data['Gap.Exp'], cv_prediction)
r2 = metrics.r2_score(data['Gap.Exp'], cv_prediction)
ax.text(0.8, 0.1, 'EAM: {:.2f} eV\n$R^2$:  {:.2f}'.format(mae, r2),
        transform=ax.transAxes,
       bbox={'facecolor': 'w', 'edgecolor': 'k'})

ax.plot(ax.get_xlim(), ax.get_xlim(), 'k--')

ax.set_xlabel('Gap Exp. (eV)')
ax.set_ylabel('ML Gap (eV)')

fig.set_size_inches(6, 6)
fig.tight_layout()
fig.savefig('Gap_Exp.png', dpi=600)
```



2 Prédiction de nouveaux matériaux par Machine Learning :

Dans cette partie nous reproduisons le calcul effectué par Ward et *al.* [1]. Sachant que la théorie de la fonctionnelle de la densité DFT est un outil fondamentale pour découvrir de nouveaux matériaux mais malheureusement elle nécessite un cout de calcul très élevé. Dans l'article [1], Ward et *al.* ont utilisé des données obtenues par la DFT pour créer des modèles performants par ML pouvant être utilisés pour développer des nouvelles recherches. Le principe de ce travail était de classifier les enthalpies de formation calculées par la DFT en un ensemble d'attributs contenant deux types distincts :

i. Attributs dépendants de la composition des propriétés des atomes constituant le matériau.
ii. Attributs obtenus par application de la tessellations de Voronoi sur la structure cristalline des composés.

Les modèles créés à l'aide de cette méthode présentent une erreur de validation croisée réduite de moitié par rapport aux modèles créés en utilisant la fonction de distribution radiale et la matrice de Coulomb. Prenant un ensemble de données de 435 000 énergies de formations extraites de la base de données OQMD, ce modèle atteint une erreur absolue moyenne de 80 mev/atome. Cette dernière est inférieure à l'erreur approximative entre les enthalpies de formation calculées par DFT et ceux mesurées expérimentalement. Ils ont également démontré que cette méthode est capable d'estimer avec précision l'énergie de formation des matériaux non incluant dans l'ensemble de données et peut être utilisée, par la suite, pour identifier des matériaux ayant des enthalpies de formation élevés [1].

Chapitre III Résultats

Entrée [1]:
```
%matplotlib inline
from matplotlib import pyplot as plt
from matminer.datasets import load_dataset
from matminer.featurizers.base import MultipleFeaturizer
from matminer.featurizers.composition import ElementProperty, Stoichiometry, ValenceOrbital, IonProperty
from matminer.featurizers.structure import (SiteStatsFingerprint, StructuralHeterogeneity,
                                            ChemicalOrdering, StructureComposition, MaximumPackingEfficiency)
from matminer.featurizers.conversions import DictToObject
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import ShuffleSplit, train_test_split
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from scipy import stats
from tqdm import tqdm_notebook as tqdm
import numpy as np
```

Créer une caractéristique

Ward et al. utilisent différentes variétés caractéristiques

Entrée [2]:
```
crt = MultipleFeaturizer([
    SiteStatsFingerprint.from_preset("CoordinationNumber_ward-prb-2017"),
    StructuralHeterogeneity(),
    ChemicalOrdering(),
    MaximumPackingEfficiency(),
    SiteStatsFingerprint.from_preset("LocalPropertyDifference_ward-prb-2017"),
    StructureComposition(Stoichiometry()),
    StructureComposition(ElementProperty.from_preset("magpie")),
    StructureComposition(ValenceOrbital(props=['frac'])),
    StructureComposition(IonProperty(fast=True))
])
```

Entrée [3]:
```
%%time
data = load_dataset("flla")
print('La base contient {} entrées'.format(len(data)))
```

```
La base contient 3938 entrées
CPU times: user 6.17 s, sys: 240 ms, total: 6.41 s
Wall time: 6.44 s
```

Entrée [4]:
```
data.head()
```

Out[4]:

	material_id	e_above_hull	formula	nsites	structure	formation_energy	formation_energy_per_atom
0	mp-10	0.107405	{'As': 1.0}	2	[[1.11758409 0.79025129 1.93571242] As, [3.352...	0.214810	0.107405
1	mp-10000	0.000000	{'S': 1.0, 'Hf': 2.0}	6	[[0. 0. 8.84051822] S, [0. ...	-7.520110	-1.253352
2	mp-10004	0.000000	{'P': 1.0, 'Mo': 3.0}	16	[[1.62101142 2.03208456 2.71687137] P, [2.9549...	-5.429887	-0.339368
3	mp-10010	0.000000	{'Al': 1.0, 'Si': 2.0, 'Co': 2.0}	5	[[0. 0. 0.] Al, [-2.29850896e-08 2.26918935e+...	-2.684175	-0.536835
4	mp-10021	0.004781	{'Ga': 1.0}	2	[[2.39728273 0.77565449 2.62420188] Ga, [0.393...	0.009563	0.004781

Entrée [5]:
```
data.columns
```

Out[5]:
```
Index(['material_id', 'e_above_hull', 'formula', 'nsites', 'structure',
       'formation_energy', 'formation_energy_per_atom'],
      dtype='object')
```

Entrée [6]:
```
dto = DictToObject(target_col_id='structure', overwrite_data=True)
data = dto.featurize_dataframe(data, 'structure')
```

DictToObject  100% 3938/3938 [00:05<00:00, 736.16it/s]

Entrée [7]:
```
data.head()
```

Out[7]:

	material_id	e_above_hull	formula	nsites	formation_energy	formation_energy_per_atom	structure
0	mp-10	0.107405	{'As': 1.0}	2	0.214810	0.107405	[[1.11758409 0.79025129 1.93571242] As, [3.352...
1	mp-10000	0.000000	{'S': 1.0, 'Hf': 2.0}	6	-7.520110	-1.253352	[[0. 0. 8.84051822] S, [0. ...
2	mp-10004	0.000000	{'P': 1.0, 'Mo': 3.0}	16	-5.429887	-0.339368	[[1.62101142 2.03208456 2.71687137] P, [2.9549...
3	mp-10010	0.000000	{'Al': 1.0, 'Si': 2.0, 'Co': 2.0}	5	-2.684175	-0.536835	[[0. 0. 0.] Al, [-2.29850896e-08 2.26918935e+...
4	mp-10021	0.004781	{'Ga': 1.0}	2	0.009563	0.004781	[[2.39728273 0.77565449 2.62420188] Ga, [0.393...

Entrée [8]:
```
%%time
print('Nombre total de caractéristiques:', len(crt.featurize(data['structure'][0])))
print('Nombre de sites dans une structures:', len(data['structure'][0]))
```

```
Nombre total de caractéristiques: 273
Nombre de sites dans une structures: 2
CPU times: user 764 ms, sys: 937 ms, total: 1.7 s
Wall time: 453 ms
```

Caractériser l'ensemble des données

Entrée [ ]:
```
%%time
X = crt.featurize_many(data['structure'], ignore_errors=True)
```

Convert `X` to a full array

Entrée [ ]:
```
X = np.array(X)
print('Forme de données d''entrée:', X.shape)
```

Vérifier le nombre d'échec sur la tessellation

Entrée [ ]:
```
import pandas as pd
echec = np.any(pd.isnull(X), axis=1)
print('Nombre d''échec: {}/{}'.format(np.sum(echec), len(echec)))
```

Apprentissage par Machine Learning

Entrée [ ]:
```
modele = Pipeline([
    ('imputer', SimpleImputer()), # Pour les structures défaillantes
    ('model', RandomForestRegressor(n_estimators=150, n_jobs=-1))])
```

Apprentissage sur l'ensemble des données

Entrée [46]:
```
%%time
modele.fit(X, data['formation_energy_per_atom'])
```

```
CPU times: user 2min 3s, sys: 226 ms, total: 2min 4s
Wall time: 17.3 s
```

Out[46]:
```
Pipeline(memory=None,
         steps=[('imputer',
                 SimpleImputer(add_indicator=False, copy=True, fill_value=None,
                               missing_values=nan, strategy='mean',
                               verbose=0)),
                ('model',
                 RandomForestRegressor(bootstrap=True, criterion='mse',
                                       max_depth=None, max_features='auto',
                                       max_leaf_nodes=None,
                                       min_impurity_decrease=0.0,
                                       min_impurity_split=None,
                                       min_samples_leaf=1, min_samples_split=2,
                                       min_weight_fraction_leaf=0.0,
                                       n_estimators=150, n_jobs=-1,
                                       oob_score=False, random_state=None,
                                       verbose=0, warm_start=False))],
         verbose=False)
```

Evaluation de la précision de l'apprentissage

Entrée [47]:
```
maes = []
rs = ShuffleSplit(train_size=3000, n_splits=20, random_state=None)
for train_ids, test_ids in rs.split(X):
    # Diviser la base en deux ensembles
    train_X = X[train_ids, :]
    train_y = data['formation_energy_per_atom'].iloc[train_ids]
    test_X = X[test_ids, :]
    test_y = data['formation_energy_per_atom'].iloc[test_ids]

    # Apprentissage
    model.fit(train_X, train_y)

    # Tester lz modèle, calculer la précision
    predict_y = model.predict(test_X)
    maes.append(np.abs(test_y - predict_y).mean())
```

Entrée [48]:
```
print('Erreur Absolue Moyenne: {:.3f}+/-{:.3f} eV/atom'.format(np.mean(maes), stats.sem(maes)))
```

```
Erreur Absolue Moyenne: 0.176+/-0.002 eV/atom
```

3 Etude Statistique:

Entrée [3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Set pandas view options
pd.set_option('display.width', 1000)
pd.set_option('display.max_columns', 30)
pd.set_option('display.max_rows', 1000)

# filter warnings messages from the notebook
import warnings
warnings.filterwarnings('ignore')
```

Entrée [4]:
```
#lecture du fichier 'Oxide.csv'
data_All_N = pd.read_csv('Oxide.csv', delimiter=';')
```

Entrée [6]:
```
data_N = data_All_N.loc[:,['chemicalFormula', 'Band Gap', 'Density', 'Enthalpy of Formation', 'Space Group', 'Volume',
                           'stability']]
```

Entrée [8]:
```
data_N.head()
```

Out[8]:

	chemicalFormula	Band Gap	Density	Enthalpy of Formation	Space Group	Volume	stability
0	Cu2 O	2.137	6.000	NaN	NaN	NaN	NaN
1	Lu2 O3	5.500	NaN	NaN	NaN	NaN	NaN
2	In2 O3	3.000	7.179	NaN	NaN	NaN	NaN
3	Ta2 O5	3.800	NaN	NaN	NaN	NaN	NaN
4	La2 O3	4.300	6.510	NaN	P-3m1, No. 164	NaN	NaN

Entrée [9]:
```
data_N.rename(columns={'Band Gap':'Band_Gap','Enthalpy of Formation':'Enthalpy_of_Formation','Space Group':'Space_Group'},
              inplace = True)
```

Entrée [10]:
```
data_N['Band_Gap'] = data_N['Band_Gap'].astype(float)
data_N['Volume'] = data_N['Volume'].astype(float)
```

Entrée [11]:
```
data_N2 = data_N.assign(Alloy=data_N['chemicalFormula'].str.count(' '))
```

Entrée [12]:
```
data_N2 = data_N2.assign(Alloy_id=data_N2['Alloy'])
```

Entrée [15]:
```
data_N2 = data_N2.replace({'Alloy':0.0},'Mono')
data_N2 = data_N2.replace({'Alloy':1.0},'binaire')
data_N2 = data_N2.replace({'Alloy':2.0},'ternaire')
data_N2 = data_N2.replace({'Alloy':3.0},'quaternaire')
data_N2 = data_N2.replace({'Alloy':4.0},'quintaine')
data_N2 = data_N2.replace({'Alloy':5.0},'plus')
data_N2 = data_N2.replace({'Alloy':6.0},'plus')
data_N2 = data_N2.replace({'Alloy':7.0},'plus')
```

Entrée [16]:
```
space_groupe = pd.read_csv('spacegroup_traite.csv', delimiter=';',usecols = ["Space_Group", "Crystal_System"])
```

Entrée [17]:
```
data_N2 = data_N2.join(space_groupe.set_index('Space_Group'), on='Space_Group')
```

Entrée [18]:
```
data_N2['Crystal_System'] = data_N2['Crystal_System'].str.title()
```

Entrée [19]:
```
data_N2.loc[(data_N2.Crystal_System == 'Triclinic'), 'N_of_Crystal_System'] = '1'
data_N2.loc[(data_N2.Crystal_System == 'Monoclinic'), 'N_of_Crystal_System'] = '2'
data_N2.loc[(data_N2.Crystal_System == 'Orthorhombic'), 'N_of_Crystal_System'] = '3'
data_N2.loc[(data_N2.Crystal_System == 'Tetragonal'), 'N_of_Crystal_System'] = '4'
data_N2.loc[(data_N2.Crystal_System == 'Trigonal'), 'N_of_Crystal_System'] = '5'
data_N2.loc[(data_N2.Crystal_System == 'Hexagonal'), 'N_of_Crystal_System'] = '6'
data_N2.loc[(data_N2.Crystal_System == 'Cubic'), 'N_of_Crystal_System'] = '7'
```

Entrée [20]:
```
data_N2.N_of_Crystal_System = pd.to_numeric(data_N2.N_of_Crystal_System)
```

Entrée [21]:
```
data_N2.to_csv('Oxide_traite.csv', sep=';')
```

Entrée [22]:
```
data_N2.head()
```

Out[22]:

	chemicalFormula	Band_Gap	Density	Enthalpy_of_Formation	Space_Group	Volume	stability	Alloy	Alloy_id	Crystal_System	N_of_Crystal_System
0	Cu2 O	2.137	6.000	NaN	NaN	NaN	NaN	binaire	1	NaN	NaN
1	Lu2 O3	5.500	NaN	NaN	NaN	NaN	NaN	binaire	1	NaN	NaN
2	In2 O3	3.000	7.179	NaN	NaN	NaN	NaN	binaire	1	NaN	NaN
3	Ta2 O5	3.800	NaN	NaN	NaN	NaN	NaN	binaire	1	NaN	NaN
4	La2 O3	4.300	6.510	NaN	P-3m1, No. 164	NaN	NaN	binaire	1	NaN	NaN

Entrée [23]:
```
print('Nombre de Binaire {:f}'.format(data_N2['Alloy'].str.count('binaire').sum()))
```

```
Nombre de Binaire 385.000000
```

Entrée [24]:
```
print('Nombre de Ternaire {:f}'.format(data_N2['Alloy'].str.count('ternaire').sum()))
```

```
Nombre de Ternaire 1203.000000
```

Entrée [25]:
```
print('Nombre de Quaternaire {:f}'.format(data_N2['Alloy'].str.count('quaternaire').sum()))
```

```
Nombre de Quaternaire 544.000000
```

```
#data_N2 = data_N2.join(df.set_index('Crystal_System'), on='Crystal_System')
```

Entrée [26]:
```
#data_N2 = data_N2.dropna(subset=['Band_Gap'])
#data_N2 = data_N2.dropna(subset=['N_of_Crystal_System'])
```

Entrée [27]:
```
data_N2_bin = data_N2.loc[(data_N2['Alloy']=='binaire') & (data_N2['Band_Gap']!=0)]
```

Entrée [28]:
```
data_N2_bin.N_of_Crystal_System = pd.to_numeric(data_N2_bin.N_of_Crystal_System)
```

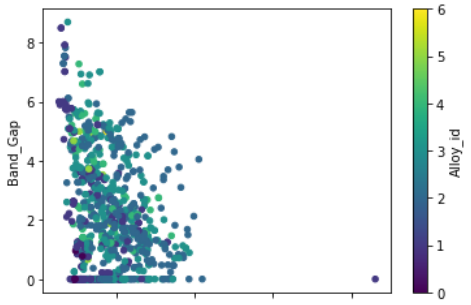
Entrée [29]:
```
print('Nombre de binaire avec un gap non nul {:d}'.format(data_N2_bin.shape[0]))
```

```
Nombre de binaire avec un gap non nul 272
```

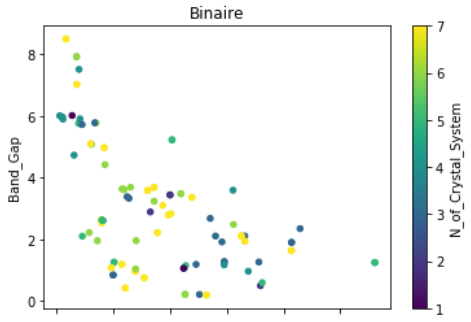
Entrée [30]:
```
print('Nombre de binaire avec un gap nul {:d}'.format(data_N2.loc[(data_N2['Alloy']=='binaire') &
                                                                  (data_N2['Band_Gap']== 0)].shape[0]))
```

```
Nombre de binaire avec un gap nul 113
```

Entrée [31]:
```
fig1 = data_N2.plot.scatter(x='Volume',
                            y='Band_Gap',
                            c='Alloy_id',
                            colormap='viridis')
```



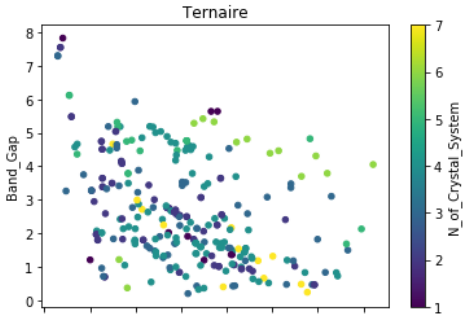
Entrée [32]:
```
fig2 = data_N2_bin.plot.scatter(x='Volume',
                                y='Band_Gap',
                                c='N_of_Crystal_System',
                                colormap='viridis',
                                title = 'Binaire')
```



Entrée [33]:
```
data_N2_ter = data_N2.loc[(data_N2['Alloy']=='ternaire') & (data_N2['Band_Gap']!=0)]
```

Entrée [34]:
```
data_N2_ter.N_of_Crystal_System = pd.to_numeric(data_N2_ter.N_of_Crystal_System)
```

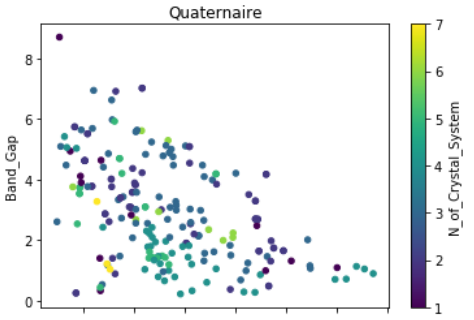
Entrée [35]:
```
fig3 = data_N2_ter.plot.scatter(x='Volume',
                                y='Band_Gap',
                                c='N_of_Crystal_System',
                                colormap='viridis',
                                title = 'Ternaire')
```



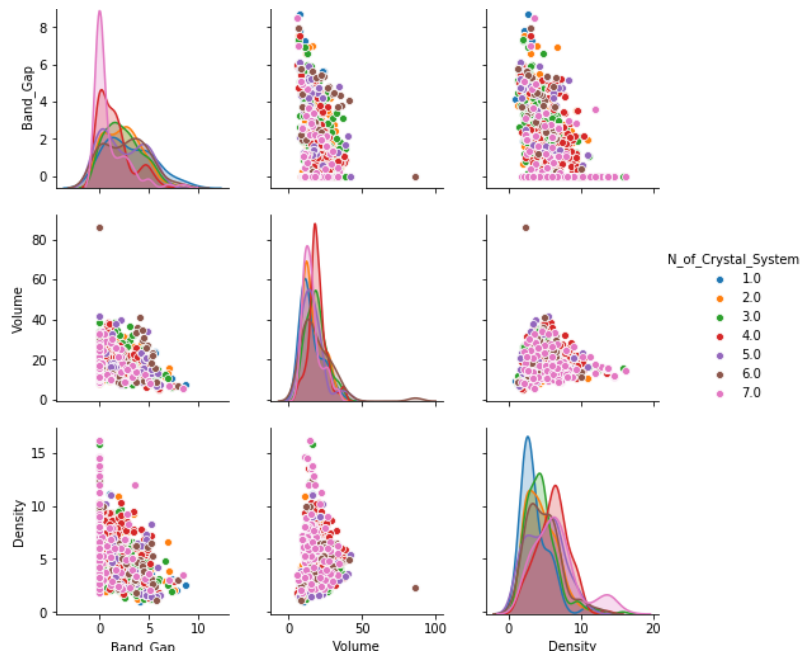
Entrée [36]:
```
data_N2_qua = data_N2.loc[(data_N2['Alloy']=='quaternaire') & (data_N2['Band_Gap']!=0)]
```

Entrée [37]:
```
data_N2_qua.N_of_Crystal_System = pd.to_numeric(data_N2_qua.N_of_Crystal_System)
```

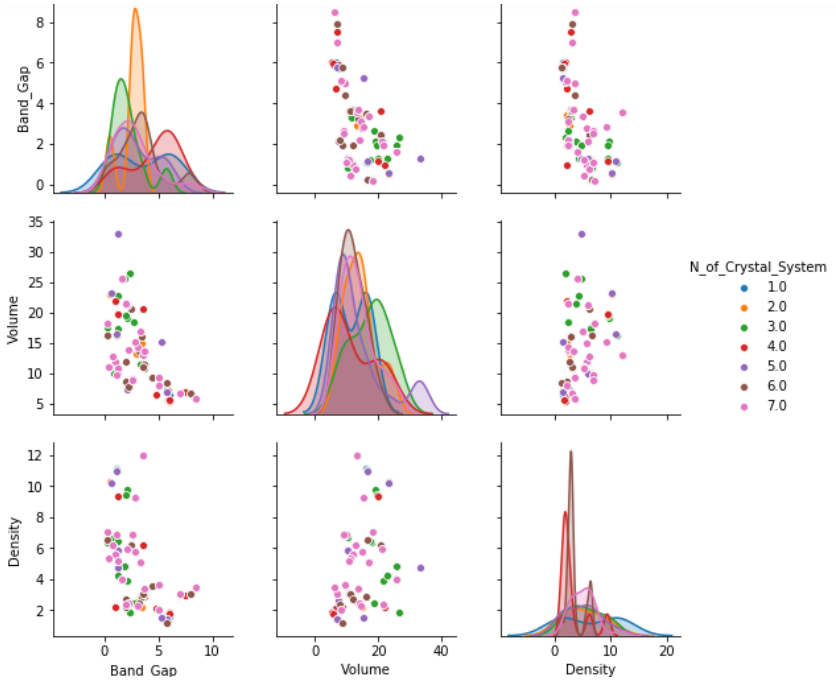
Entrée [38]:
```
fig4 = data_N2_qua.plot.scatter(x='Volume',
                                y='Band_Gap',
                                c='N_of_Crystal_System',
                                colormap='viridis',
                                title='Quaternaire')
```



Entrée [39]:
```
fig5 = sns.pairplot(data_N2, vars = ['Band_Gap', 'Volume','Density'],hue="N_of_Crystal_System",dropna= True)
```




```
Entrée [40]: fig6 = sns.pairplot(data_N2_bin, vars = ['Band_Gap', 'Volume','Density'],hue="N_of_Crystal_System", dropna= True)
```



Chapitre III Résultats

Entrée [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
# Set pandas view options
pd.set_option('display.width', 1000)
pd.set_option('display.max_columns', 30)
pd.set_option('display.max_rows', 1000)

# filter warnings messages from the notebook
import warnings
warnings.filterwarnings('ignore')
```

Entrée [2]:
```
data_Oxide = pd.read_csv('Oxide_traite.csv', delimiter=';')
```

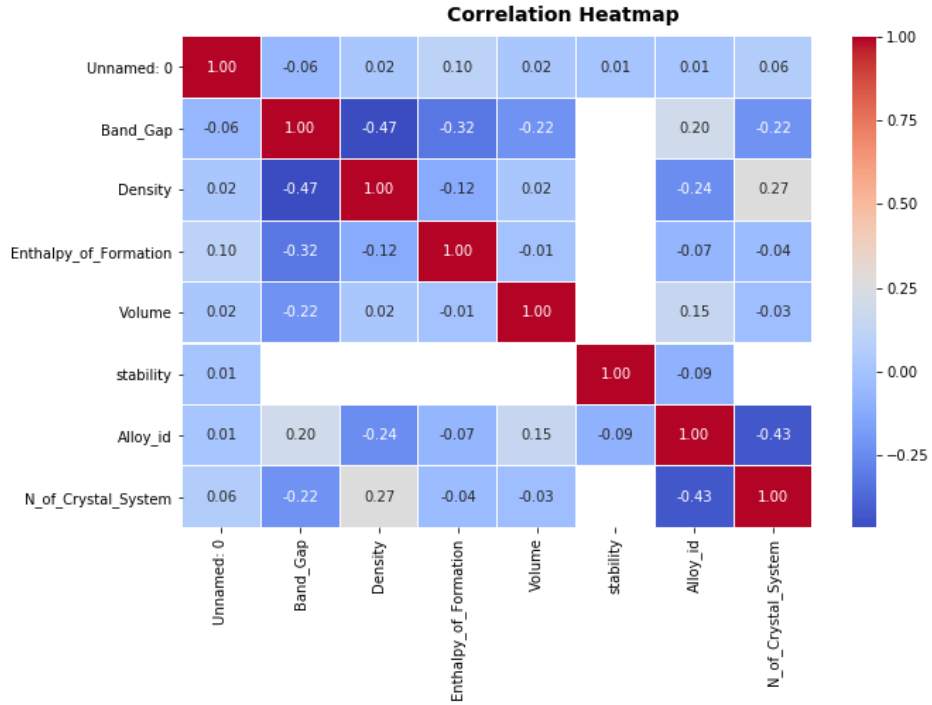
Entrée [3]:
```
data_Cuivre = pd.read_csv('Cuivre_traite.csv', delimiter=';')
```

Entrée [4]:
```
data_Nitrite = pd.read_csv('Nitrite_traite.csv', delimiter=';')
```

Entrée [5]:
```
corr = data_Oxide.corr()
```

Entrée [6]:
```
fig3, (ax) = plt.subplots(1, 1, figsize=(10,6))

hm = sns.heatmap(corr,
                 ax=ax,
                 cmap="coolwarm",
                 annot=True,
                 fmt='.2f',
                 #annot_kws={"size": 14},
                 linewidths=.05)

fig3.subplots_adjust(top=0.93)
fig3.suptitle('Correlation Heatmap',
              fontsize=14,
              fontweight='bold')
```

Out[6]:
```
Text(0.5, 0.98, 'Correlation Heatmap')
```



Bibliographies

[1] Logan Ward, Ruoqian Liu, Amar Krishna, Vinay I. Hegde, Ankit Agrawal, Alok Choudhary, and Chris Wolverton. « Including crystal structure attributes in machine learning models of formation energies via Voronoi tessellations ». Université Northwestern, Evanston, Illinois 60208, États-Unis.14 Juillet 2017. DOI : 10.1103/PhysRevB.96.024104

Conclusion et Perspective

Conclusion et Perspective

Dans ce travail, nous avons exploré ce que l'apprentissage automatique peut apporter à la compréhension et à l'amélioration dans le domaine des sciences des matériaux, notamment la prédiction de nouveaux matériaux sans utiliser les techniques de la DFT, gourmandes en calcul numérique. Néanmoins, afin d'avoir un très bon résultat ou une bonne prédiction, il est nécessaire d'avoir une large base de données.

Cette nouvelle technique d'investigation est rendue possible grâce au développement et le partage open-source des données à l'image de Citrination, Materials Project, The Open Quantum Materials Database et d'autres bases de données qui sont entrain de voir le jour dans des équipes de simulation ab-initio à travers le monde. Ainsi, le développement de nouvelles bibliothèques sous python, par exemple scikit-learn ou matminer, a grandement facilité l'application de l'intelligence artificielle dans le domaine physique en général et dans les sciences des matériaux en particulier, surtout pour les non-spécialistes.

Nous croyons que l'utilisation de l'apprentissage automatique pour l'analyse de grandes quantités de données, surtout expérimentales, peut être utilisée pour découvrir une nouvelle physique, puisque le machine Learning n'a pas d'idées préconçues. La suite de ce travail consistera à créer et enrichir une base de données des propriétés physiques des matériaux des équipes de recherche locales et d'implémenter la technique de l'apprentissage profond.

Abstract :

In the last years, the materials science community has made considerable efforts to use informatics to accelerate the development and discovery of new materials. The algorithms of machine learning analyze material properties data to extract new knowledge or to predictive models representing the behavior of materials from existing databases in materials science. This technique is less expensive in computing time than traditional ab-initio codes. In this master's thesis, we have implemented the algorithms of machine learning, in python, using Scikit-learn to extract data from platforms such as Materials Project and Citrination.

Keywords: machine learning, database, python, Scikit-learn, gap.

Résumé :

Au cours des dernières années, la communauté des sciences de matériaux a déployé des efforts considérables pour utiliser l'informatique afin d'accélérer le développement et la découverte de nouveaux matériaux. Les algorithmes d'apprentissage automatique analysent les données relatives aux propriétés des matériaux afin d'extraire de nouvelles connaissances ou des modèles prédictifs représentant leur comportement à partir de bases de données existantes en science des matériaux. Cette technique est moins coûteuse en temps de calcul que les codes ab-initio traditionnels. Dans ce mémoire master, nous avons implémenté les algorithmes d'apprentissage automatique, en python, en utilisant Scikit-learn pour extraire des informations à partir des bases de données telles que Materials Project et Citrination.

Mots clés : apprentissage statistique, base de données, python, Scikit-learn, gap.

ملخص:

في السنوات الأخيرة، بذل مجتمع علوم المادة جهودًا كبيرة لاستخدام المعلوماتية بهدف تسريع عملية تطوير واكتشاف مواد جديدة. تقوم خوارزميات التعلم الآلي بتحليل بيانات تحتوي على خصائص المواد لاستخراج معلومات جديدة تمثل سلوك المواد من خلال قواعد البيانات الموجودة في علوم المادة. هذه التقنية أقل تكلفة في وقت الحوسبة من رموز ab-initio التقليدية. في أطروحة هذا الماستر، قمنا بتنفيذ خوارزميات التعلم الآلي، في بايثون، باستخدام Scikit-Learn لاستخراج المعلومات من قواعد البيانات مثل Materials Project وCitrination.

الكلمات المفتاحية: تعلم الآلة، قاعدة بيانات، بيثون، Scikit-Learn، gap.