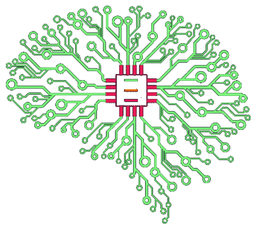
Intelligence Artificielle

CPGE – TSI → Sciences Industrielles de l'Ingénieur (SII)

Wikipédia – L'intelligence artificielle (IA) est « l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence ».

Introduction

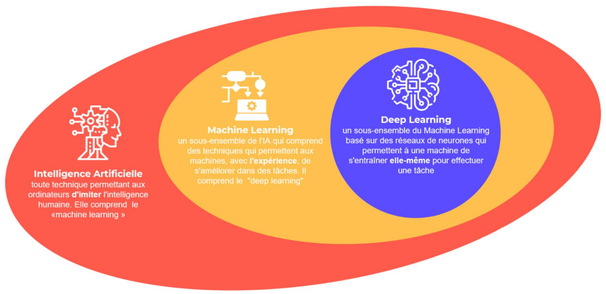
L'objectif de l'IA n'est plus de simuler l'intelligence humaine mais de créer des **machines capables d'apprendre**



Au programme de SII, on désigne par le terme d'Intelligence Artificielle des méthodes permettant d'effectuer une prise de décision (aussi appelée **prédiction**) à partir de la reproduction d'un comportement observé sur des données réelles, et non à partir d'un modèle de connaissance choisi par un humain.

Machine Learning

L'apprentissage automatique est un des domaines de l'IA. Il enregistre depuis quelques années une forte croissance grâce l'accès aux grandes quantités de données et à une puissance de calcul élevée et peu couteuse.

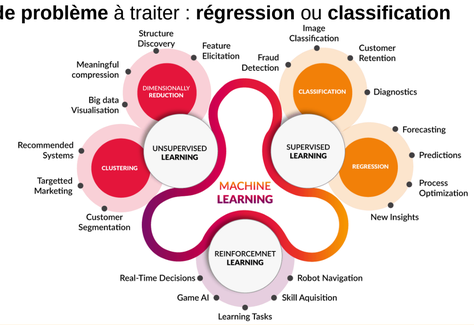


A partir d'un grand nombre de données, en se basant sur des statistiques, des reconnaissances de motifs, l'apprentissage automatique va pouvoir effectuer des prédictions.

Classification des algorithmes

Les algorithmes ne sont pas tous destinés aux mêmes usages. On les classe selon deux critères :

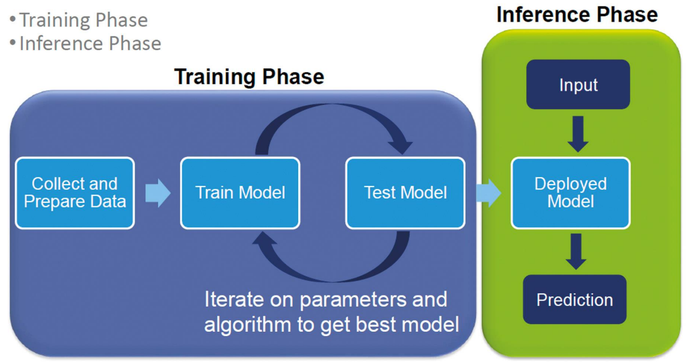
- Le **mode d'apprentissage** : algorithmes **supervisés** ou **non-supervisés**
- Le **type de problème** à traiter : **régression** ou **classification**



Les étapes d'apprentissage

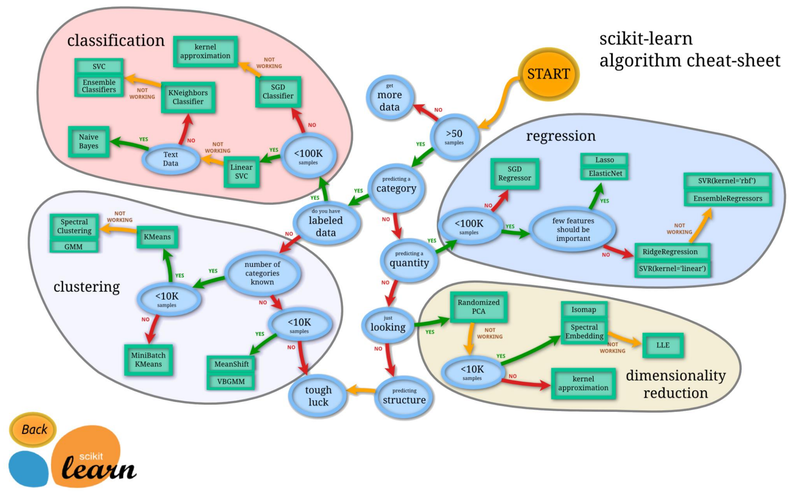
Tous les algorithmes d'apprentissage automatique fonctionnent en plusieurs étapes :

- **Première étape :** récolter les **données** et les mettre en forme (matrice X). On **sépare** ensuite les données en deux parties : un jeu dit d'entrainement (80%) et un jeu test (20%). Dans le cas d'un apprentissage supervisé, pour chacun de ces jeux de données, la valeur de la sortie est connue (vecteur Y).
- **Deuxième étape :** Phase d'apprentissage ou d'**entraînement** qui permet à la machine d'apprendre un modèle prédictif à partir du **jeu d'entraînement**. A chaque itération, on **évalue** le modèle avec **le jeu test** afin de valider le meilleur modèle.
- **Troisième étape :** phase d'**inférence** pendant laquelle le modèle déjà entraîné est utilisé pour reproduire le comportement appris sur des nouvelles données.



SciKit-Learn

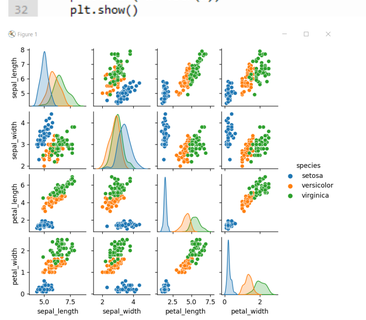
La bibliothèque Python scikit-learn propose une classification de divers algorithmes en fonction des apprentissages automatiques.

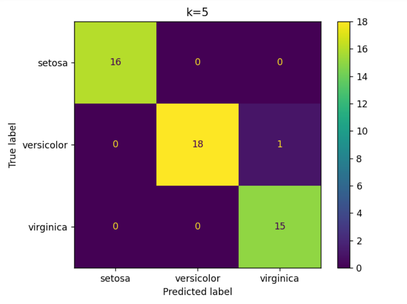


Exemple : KNN

L'algorithme des k plus proches voisins (aussi appelé k-nearest neighbors algorithm – k-NN) permet de réaliser des opérations de régression et de classification sur un ensemble de données.

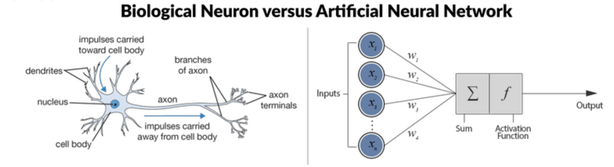
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split # découpage des données
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

data = pd.read_csv("iris.csv")
print(data.head())
data.info()
sns.pairplot(data, hue="species",height=1.5)
plt.show()

iris_X = data[['sepal_length','sepal_width','petal_length','petal_width']]
iris_Y = data['species']

X_train, X_test, Y_train, Y_test = train_test_split(iris_X,iris_Y,test_size=1/3,random_state=0)
# Création du classifieur

for k in [4,5]:
    knn = KNeighborsClassifier(n_neighbors=k) # KNeighborsClassifier(n_neighbors=k) pour choisir k
    # Entraînement du classifieur
    knn.fit(X_train, Y_train)
    # Prédiction sur le jeu de test (à comparer avec Y_test)
    Y_pred=knn.predict(X_test)

    cm=confusion_matrix(Y_test,Y_pred,labels=['setosa', 'versicolor', 'virginica'])
    disp=ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['setosa', 'versicolor', 'virginica'])
    disp.plot()
    plt.title('k='+str(k))
    plt.show()
```





Deep Learning

Le **Deep Learning** est basé sur la mise en œuvre de **réseaux de neurones**.



$$y_p = f(z) = f\left(\boldsymbol{b} + \sum_i \boldsymbol{w_i}.\boldsymbol{x_i}\right)$$

```python
from sklearn.neural_network import MLPClassifier
# Définition du modèle
mlp = MLPClassifier(hidden_layer_sizes=(50, ), max_iter=10, solver='sgd', verbose=10, learning_rate_init=.1)
# Entrainement
model_final = mlp.fit(X_train, y_train)
 # Calcul des précisions
print(f"Précision sur les données d'entraînement : {mlp.score(X_train, y_train):.3f}")
print(f"Précision sur les données de test : {mlp.score(X_test, y_test):.3f}")
```