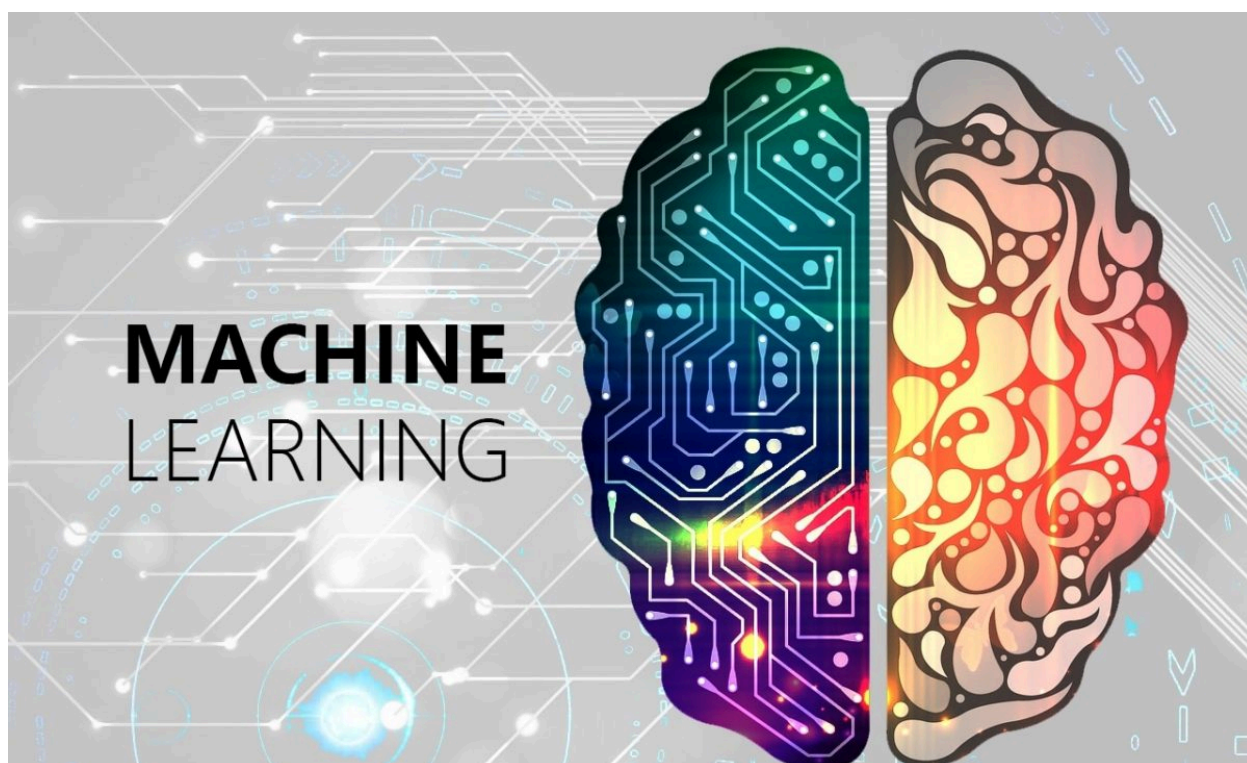


# Note d'application



**Réalisation :** ANGLADE Alexis  
**Cliente :** Mme. EL RACHKIDY Nancy (LIMOS)  
**Tuteur industriel :** M. FICKINGER Pascal  
**Professeur référent :** M. LANDRAULT Alexis



---

## Sommaire

<b>Introduction.....</b>	<b>6</b>
1. Apprentissage supervisé.....	7
2. Apprentissage non supervisé.....	8
3. Apprentissage semi-supervisé.....	9
4. Apprentissage par renforcement.....	10
5. Apprentissage auto-supervisé.....	11
6. Choix de la famille adaptée au projet.....	12
7. k-Nearest Neighbors (kNN).....	13
8. Naïve Bayes.....	15
9. Decision Tree.....	17
10. Random Forest.....	19
11. Support Vector Machine (SVM).....	21
<b>Conclusion.....</b>	<b>23</b>



---

## Table des figures

Figure 1 : Fonctionnement de l'algorithme kNN.....	14
Figure 2 : Fonctionnement de l'algorithme Naïve Bayes.....	16
Figure 3 : Arbre de décision.....	18
Figure 4 : Fonctionnement de l'algorithme Random Forest.....	20
Figure 5 : Fonctionnement de l'algorithme Support Vector Machine.....	22
Figure 6 : Précisions de chaque algorithme pour notre projet.....	23



---

## Introduction

Dans le cadre de notre projet de fin d'études à Polytech Clermont, nous avons travaillé sur l'hypothèse qu'il était possible d'identifier l'émetteur d'un signal utilisant le protocole de communication LoRa uniquement à partir d'une caractéristique physique de la trame : le CFO (pour Carrier Frequency Offset).

L'objectif principal du projet a été de démontrer que cette hypothèse était vraie. Pour cela, nous avons eu à notre disposition deux modules permettant l'émission de trames LoRa ainsi qu'une passerelle configurable permettant de recevoir les trames et de calculer le CFO de chacune d'entre elles.

Pour prouver la véracité de l'hypothèse, notre démarche nous a amené à constituer une base de données comprenant les CFO, les SF (pour Spreading Factor, une autre caractéristique du protocole LoRa) et les émetteurs d'un certain nombre de trames émises et reçues pour pouvoir ensuite réaliser du Machine Learning avec cette base de données.

Dans le cadre de ce projet, le présent document a donc pour but de faire deux états de l'art. Le premier présente les différentes familles d'algorithmes de Machine Learning existantes, puis après avoir expliqué le choix de la famille que nous avons retenue pour notre cas, le deuxième état de l'art présente les spécificités des cinq algorithmes testés.

# 1. Apprentissage supervisé

## 1.1. Description

L'apprentissage supervisé repose sur l'utilisation de données étiquetées pour entraîner un modèle à prédire une sortie à partir d'entrées. Le modèle apprend ainsi à associer des **caractéristiques** avec des **étiquettes** en minimisant une fonction de perte.

Selon la nature de la sortie attendue, on distingue deux principales sous-catégories. La **classification**, qui consiste à prédire des catégories discrètes, comme la détection de spam ou la reconnaissance d'images et la **régression**, qui, quant à elle, est utilisée pour prédire des valeurs continues, par exemple l'estimation du prix d'une maison.

### Exemples d'algorithmes :

- k-Nearest Neighbors (kNN)
- Naïve Bayes
- Decision Tree
- Random Forest
- Support Vector Machine (SVM)

## 1.2. Avantages

L'apprentissage supervisé est donc très performant lorsqu'il existe une relation claire entre les entrées et les sorties, ce qui permet d'obtenir des prédictions fiables. De plus, il offre souvent des résultats facilement interprétables, notamment à travers l'utilisation d'arbres de décision qui permettent de visualiser les choix effectués par le modèle. Grâce à sa flexibilité et à son efficacité, il est utilisé dans un large éventail d'applications, allant de la vision par ordinateur au traitement du langage naturel (NLP), en passant par la finance et bien d'autres domaines.

## 1.3. Inconvénients

En revanche, il présente certains inconvénients, notamment la nécessité d'un grand volume de données étiquetées, ce qui peut engendrer des coûts élevés en annotation. De plus, il est souvent sensible au sur-apprentissage, ce qui signifie que le modèle peut mémoriser les données d'entraînement au lieu de généraliser à de nouvelles données. Enfin, ses performances restent limitées lorsqu'il est appliqué à des problèmes nécessitant une exploration non supervisée, car il repose sur des correspondances prédéfinies entre entrées et sorties.



## 2. Apprentissage non supervisé

### 2.1. Description

L'apprentissage non supervisé est utilisé lorsque les données ne sont pas étiquetées. Le modèle cherche alors à **détecter des structures** ou des **relations cachées** dans les données.

Les algorithmes de cette famille se déclinent en plusieurs approches. Le **clustering** est l'une des plus courantes : il regroupe les données en ensembles homogènes appelés "clusters", où les éléments d'un même cluster partagent des caractéristiques similaires. Une autre approche est la **réduction de dimensionnalité**, qui permet de projeter les données dans un espace réduit tout en conservant l'information essentielle, facilitant ainsi leur interprétation et leur traitement. Enfin, l'**apprentissage pour la détection d'anomalies** vise à identifier les points de données atypiques qui diffèrent significativement du reste des observations.

#### Exemples d'algorithmes :

- k-Means
- DBSCAN
- Algorithme de regroupement hiérarchique
- Analyse en composantes principales (PCA)
- Autoencodeurs

### 2.2. Avantages

L'apprentissage non supervisé est donc particulièrement utile pour explorer des données non étiquetées, car il permet d'identifier des structures et des tendances cachées. Il est notamment efficace pour la segmentation et la découverte de patterns dans les données, facilitant ainsi leur analyse et leur interprétation. De plus, il peut être utilisé comme une étape de prétraitement avant un apprentissage supervisé, en extrayant des caractéristiques pertinentes qui amélioreront les performances des modèles supervisés.

### 2.3. Inconvénients

En revanche, les résultats obtenus en apprentissage non supervisé sont souvent plus difficiles à interpréter que ceux issus d'un apprentissage supervisé, car l'absence d'étiquettes rend l'évaluation plus complexe. De plus, il n'existe pas de métrique universelle, comme "l'accuracy" en classification, pour mesurer efficacement les performances du modèle. Enfin, ce type d'apprentissage est fortement influencé par le choix des hyperparamètres, tels que le nombre de clusters dans un algorithme de regroupement, ce qui peut nécessiter plusieurs expérimentations avant d'obtenir un résultat optimal.

---

## 3. Apprentissage semi-supervisé

### 3.1. Description

L'apprentissage semi-supervisé utilise un **petit ensemble de données étiquetées** et un **grand ensemble de données non étiquetées** pour entraîner un modèle.

**Exemples d'algorithmes :**

- Self-training
- Co-training
- Graph-based learning
- Réseaux de neurones pré-entraînés (ex. GPT fine-tuné)

### 3.2. Avantages

L'apprentissage semi-supervisé permet donc de réduire considérablement le coût d'annotation des données, car il exploite un grand ensemble de données non étiquetées en complément d'un petit ensemble étiqueté.

Cette approche peut donc améliorer la performance des modèles par rapport à un apprentissage strictement supervisé, notamment lorsque la quantité de données étiquetées est limitée.

De ce fait, il est largement utilisé dans des domaines tels que le traitement du langage naturel (NLP) et la vision par ordinateur, où l'annotation manuelle des données est particulièrement coûteuse et chronophage.

### 3.3. Inconvénients

En revanche, il peut être instable si les données non étiquetées ne sont pas bien exploitées.

---

## 4. Apprentissage par renforcement

### 4.1. Description

L'apprentissage par renforcement repose sur un **agent** qui interagit avec un **environnement** en effectuant des actions et en recevant des **récompenses** positives ou négatives.

#### Exemples d'algorithmes :

- Q-Learning
- Deep Q-Networks (DQN)
- Policy Gradient Methods (REINFORCE, PPO)
- Actor-Critic

### 4.2. Avantages

Ce type d'apprentissage est ainsi très performant pour les tâches de prise de décision (ex. jeux vidéo, robotique, finance).

### 4.3. Inconvénients

En revanche, il est très coûteux en calcul.

---

## 5. Apprentissage auto-supervisé

### 5.1. Description

L'apprentissage auto-supervisé est une variante récente où un modèle génère ses propres labels à partir des données, souvent en construisant une tâche auxiliaire (ex. prédire des parties manquantes d'une image ou d'un texte).

#### Exemples d'algorithmes :

- BERT (NLP)
- SimCLR, MoCo (vision)

### 5.2. Avantages

Ce type d'apprentissage réduit donc drastiquement la dépendance aux labels humains.

### 5.3. Inconvénients

En revanche, il demande également une grande puissance de calcul pour le pré-entraînement.

---

## 6. Choix de la famille adaptée au projet

### 6.1. Objectif du Machine Learning

Pour choisir au mieux la famille de Machine Learning la plus adaptée au projet, il faut d'abord rappeler l'objectif de l'étape de Machine Learning au sein du projet.

L'objectif était de valider l'hypothèse, pré-validée par les tests ayant précédés la création de la base de données, que l'on peut identifier l'émetteur d'une trame LoRa uniquement à partir de son CFO.

### 6.2. Base de données

La base de données créée comprend donc, pour chaque trame, le CFO spécifique à la trame, le SF utilisé et l'émetteur de la trame.

### 6.3. Famille la plus adaptée

Étant donné que nous avons donc des données étiquetées, la famille la plus adaptée, au vu des avantages et inconvénients de chaque famille, est celle de l'apprentissage supervisé.

Les parties suivantes vont donc détailler les caractéristiques des cinq algorithmes d'apprentissage supervisé sélectionnés pour notre étape de Machine Learning.

## 7. k-Nearest Neighbors (kNN)

### 7.1. Fonctionnement de l'algorithme

L'algorithme k-Nearest Neighbors (kNN) est un algorithme de classification supervisée. Il fonctionne en calculant les distances entre un point de test et les points d'entraînement, puis en attribuant la classe la plus fréquente parmi les k voisins les plus proches. La distance la plus courante utilisée est la distance euclidienne, mais d'autres métriques comme la distance Manhattan ou la distance de Minkowski peuvent aussi être utilisées.

### 7.2. Spécificités de l'algorithme

L'algorithme k-Nearest Neighbors (kNN) est un modèle **non paramétrique**, ce qui signifie qu'il ne fait aucune hypothèse sur la distribution des données, contrairement à d'autres algorithmes comme Naïve Bayes ou la régression linéaire.

Il est **basé sur la proximité**, prenant des décisions en fonction de la similarité entre les points de test et leurs voisins les plus proches dans l'espace des caractéristiques. Cette approche permet de classer un point en fonction des classes des points voisins.

De plus, il offre la possibilité de **pondérer les voisins** en fonction de leur distance au point cible, ce qui signifie que les voisins plus proches peuvent avoir une influence plus grande sur la prédiction que ceux plus éloignés, améliorant ainsi la précision du modèle dans certains cas.

### 7.3. Algorithme

1. Choisir le nombre de voisins k.
2. Calculer la distance entre le point de test et tous les points d'entraînement.
3. Trier les points par distance croissante.
4. Sélectionner les k voisins les plus proches.
5. Assigner la classe la plus fréquente parmi ses k voisins.

### 7.4. Avantages

L'algorithme kNN est **simple à comprendre et à implémenter**, ce qui en fait une option accessible pour ceux qui débutent en apprentissage automatique.

De plus, il est particulièrement **efficace pour des ensembles de données de petite taille**, car les calculs nécessaires pour déterminer les voisins proches restent raisonnables.

Enfin, kNN est **capable de traiter des données multi-classes**, ce qui le rend adapté à une grande variété de problèmes de classification.

## 7.5. Inconvénients

En revanche, il présente plusieurs inconvénients. Il devient **très coûteux en termes de calcul** lorsque la taille du dataset est grande, car il nécessite de calculer les distances entre le point à classer et tous les autres points d'entraînement, ce qui peut être lent. Il est également **sensible au choix du paramètre  $k$** ; un  $k$  mal ajusté peut affecter négativement les performances du modèle.

De plus, kNN est **sensible à l'échelle des données**, ce qui signifie que les caractéristiques avec des échelles différentes peuvent dominer les calculs de distance, rendant le modèle moins performant. L'algorithme est aussi **sensible aux données bruyantes ou aux données aberrantes**, car ces dernières peuvent influencer les décisions de classification en affectant les voisins proches.

Enfin, kNN rencontre des difficultés avec des **données de grande dimension**, un problème connu sous le nom de "malédiction de la dimensionnalité", où la distance entre les points devient de plus en plus homogène et donc moins utile pour distinguer les classes.

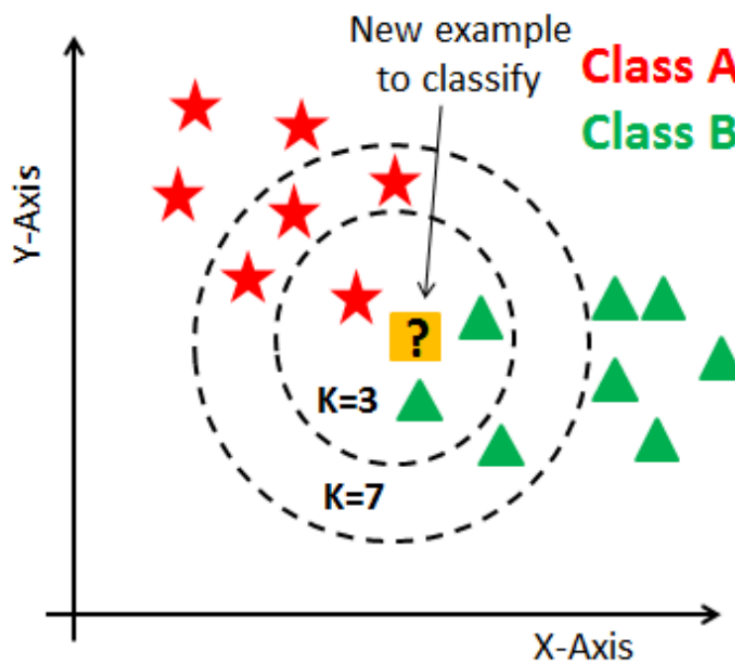


Figure 1 : Fonctionnement de l'algorithme kNN

## 8. Naïve Bayes

### 8.1. Fonctionnement de l'algorithme

Naïve Bayes est un algorithme probabiliste basé sur le théorème de Bayes, qui suppose que les caractéristiques sont indépendantes entre elles. L'algorithme utilise les probabilités conditionnelles pour estimer la classe d'un exemple donné.

### 8.2. Spécificités de l'algorithme

L'algorithme Naïve Bayes repose sur l'**indépendance conditionnelle**, supposant que toutes les caractéristiques sont indépendantes les unes des autres, ce qui simplifie considérablement les calculs, bien que cette hypothèse soit souvent irréaliste dans la réalité.

De plus, il est **basé sur des probabilités**, car il calcule la probabilité d'appartenance à une classe donnée en fonction des probabilités conditionnelles des caractéristiques observées. En combinant ces probabilités, l'algorithme peut prédire la classe la plus probable pour une nouvelle observation.

### 8.3. Algorithme

1. Calculer la probabilité a priori pour chaque classe.
2. Calculer les probabilités conditionnelles des caractéristiques pour chaque classe.
3. Utiliser la règle de Bayes pour estimer la probabilité d'une classe donnée les caractéristiques de l'exemple.
4. Assigner la classe avec la probabilité la plus élevée.

### 8.4. Avantages

L'algorithme Naïve Bayes est **très rapide**, ce qui le rend particulièrement adapté pour les **grandes bases de données**, où la rapidité de calcul est un atout majeur. Il reste également **efficace même avec peu de données**, ce qui en fait un bon choix lorsque les échantillons d'entraînement sont limités.

De plus, Naïve Bayes nécessite **peu de paramètres à ajuster**, ce qui simplifie son utilisation et l'optimisation du modèle. Cet algorithme **fonctionne bien pour des données de type texte**, comme dans le cas de la **classification de spam**, en raison de sa capacité à traiter efficacement des caractéristiques indépendantes comme les mots dans un document.



## 8.5. Inconvénients

En revanche, il repose sur une **hypothèse d'indépendance** entre les caractéristiques, ce qui est **irréaliste dans de nombreux cas** et constitue la raison pour laquelle il est qualifié de "naïf". Cette hypothèse rend l'algorithme **moins performant lorsque les caractéristiques sont fortement corrélées**, car il ne prend pas en compte ces relations.

De plus, Naïve Bayes **ne gère pas bien les relations complexes entre les variables**, ce qui peut limiter sa capacité à capturer des dépendances plus subtiles et à modéliser des structures plus complexes dans les données.

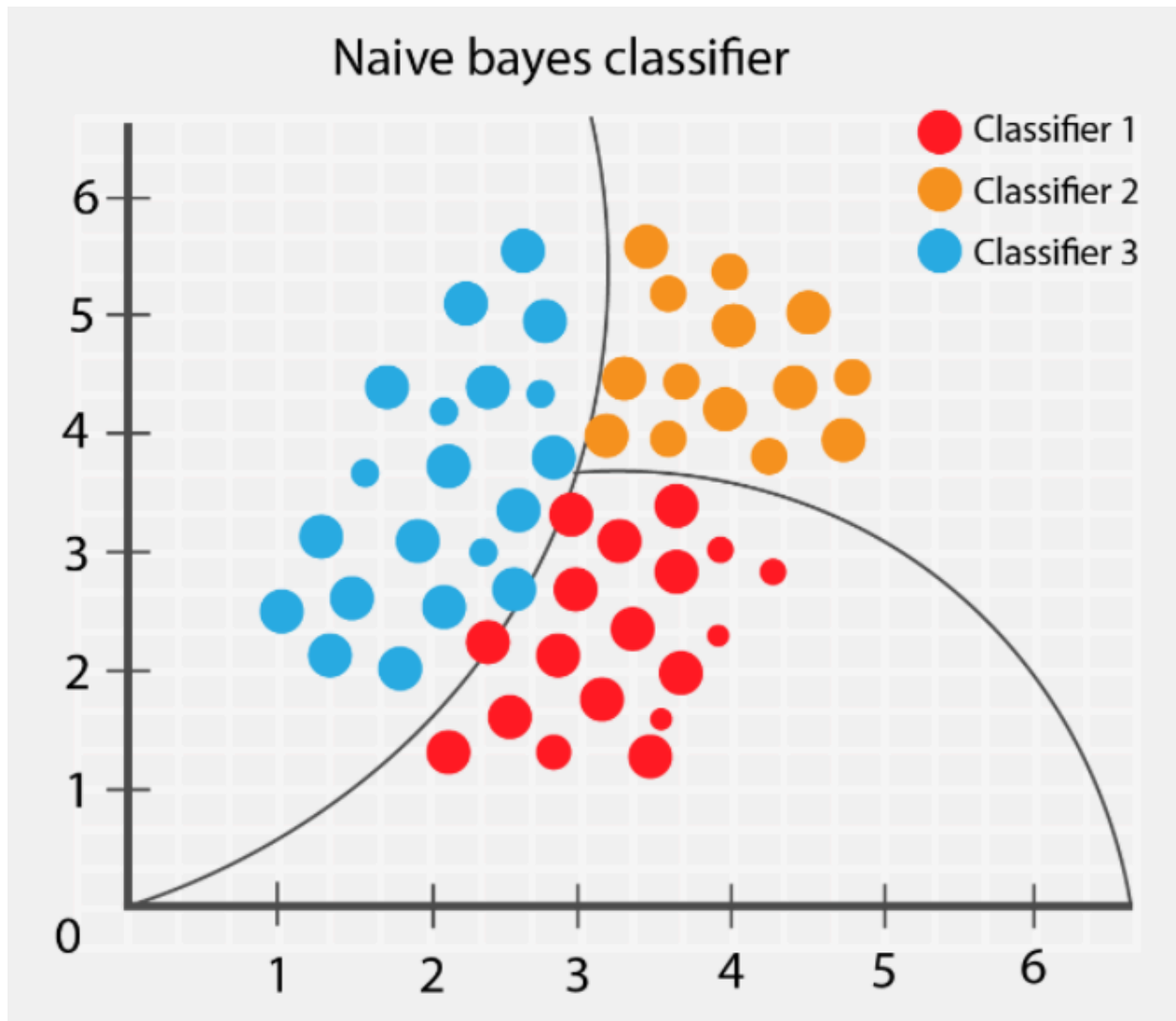


Figure 2 : Fonctionnement de l'algorithme Naïve Bayes

## 9. Decision Tree

### 9.1. Fonctionnement de l'algorithme

Un arbre de décision est un modèle en forme d'arbre qui prend des décisions basées sur les valeurs des caractéristiques. Chaque nœud de l'arbre représente un test sur une caractéristique, chaque branche représente l'issue de ce test, et chaque feuille représente une classe ou une valeur de régression.

### 9.2. Spécificités de l'algorithme

L'algorithme des arbres de décision repose sur la stratégie de **diviser et conquérir**, où l'arbre construit des sous-groupes de manière récursive en fonction des caractéristiques des données. À chaque étape, il choisit la meilleure caractéristique pour effectuer la division en fonction de critères comme l'**entropie** ou le **gain d'information**, qui permettent de maximiser la séparation des données et d'optimiser la pureté des sous-groupes obtenus. Ces critères aident l'algorithme à prendre des décisions sur la manière de diviser les données pour atteindre une classification ou une régression optimale.

### 9.3. Algorithme

1. Choisir la meilleure caractéristique pour séparer les données (en fonction de la mesure de la pureté comme l'entropie ou le gain d'information).
2. Diviser les données en sous-ensembles en fonction de cette caractéristique.
3. Répéter ce processus pour chaque sous-ensemble jusqu'à ce que tous les points soient classés ou qu'une condition d'arrêt soit remplie (par exemple, profondeur maximale ou pureté parfaite).

### 9.4. Avantages

L'algorithme des arbres de décision est **facile à comprendre et à interpréter**, ce qui en fait un choix populaire pour les modèles explicables. Il prend en charge à la fois des **données catégorielles et numériques**, offrant une grande flexibilité pour différents types de données.

De plus, il n'est **pas nécessaire de normaliser les données**, ce qui simplifie la préparation des données. L'arbre de décision est également capable de **capturer des relations non linéaires** entre les variables, ce qui lui permet de s'adapter à des problèmes complexes où les frontières de décision ne sont pas simplement linéaires.

## 9.5. Inconvénients

En revanche, il est **sensible aux petites variations dans les données**, ce qui peut conduire à un **surapprentissage**, où le modèle s'ajuste trop spécifiquement aux données d'entraînement et perd sa capacité de généralisation.

De plus, si l'arbre n'est pas limité, il peut devenir **très grand et complexe**, rendant le modèle difficile à interpréter. Enfin, l'algorithme peut être **instable** lorsqu'il est appliqué à des données bruitées, car de petites variations ou des erreurs dans les données peuvent entraîner des changements significatifs dans la structure de l'arbre, réduisant ainsi sa robustesse.

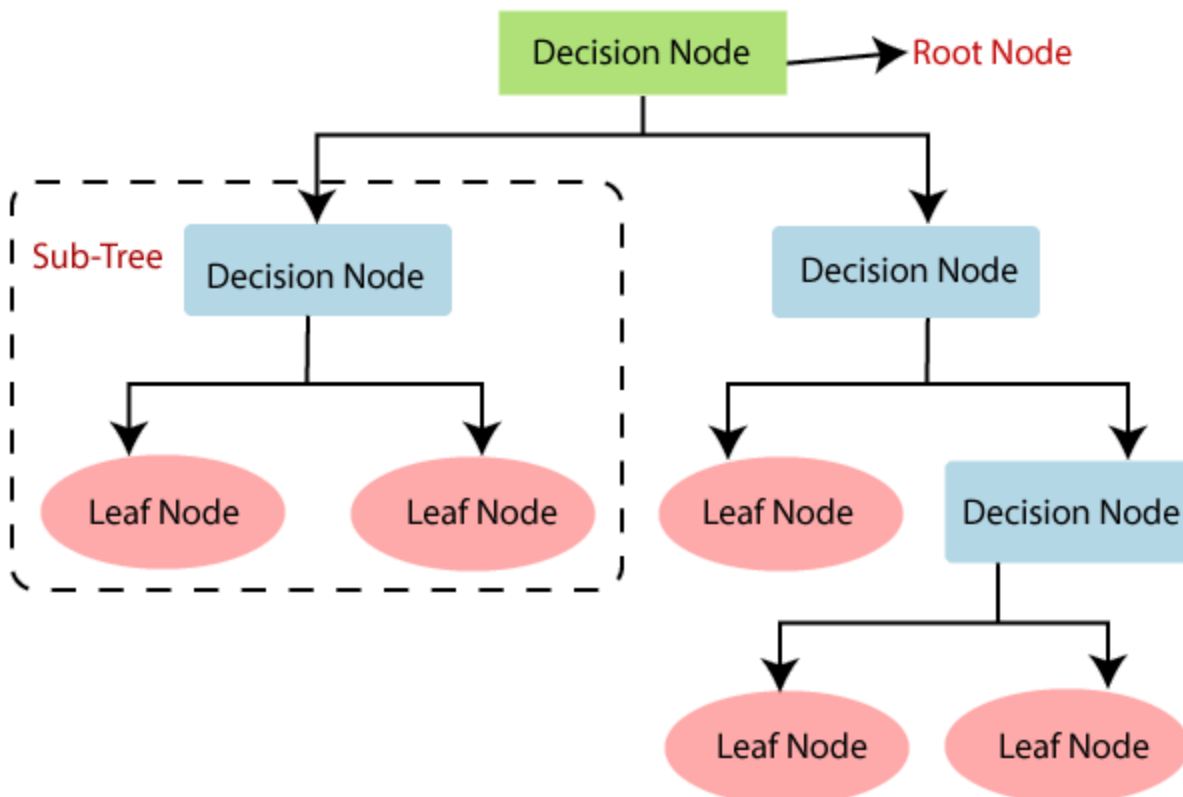


Figure 3 : Arbre de décision

## 10. Random Forest

### 10.1. Fonctionnement de l'algorithme

Random Forest est un ensemble d'arbres de décision, qui fonctionne en créant plusieurs arbres sur des sous-échantillons aléatoires des données et en prenant la moyenne (ou la majorité) des prédictions. Cela réduit la variance et améliore la performance par rapport aux arbres de décision individuels.

### 10.2. Spécificités de l'algorithme

L'algorithme Random Forest repose sur la méthode d'**ensemble learning**, qui combine plusieurs modèles pour améliorer la performance globale. Il utilise le **bagging**, une technique d'échantillonnage avec remplacement (bootstrap) pour créer des sous-ensembles de données différents, ce qui permet de réduire la variance et d'éviter le surapprentissage.

De plus, lors de la construction des arbres, une **sélection aléatoire des caractéristiques** est effectuée à chaque division, ce qui permet d'introduire de la diversité parmi les arbres et d'améliorer la robustesse du modèle en réduisant la corrélation entre eux.

### 10.3. Algorithme

1. Créer plusieurs arbres de décision indépendants sur des sous-échantillons des données.
2. Pour chaque arbre, faire des prédictions.
3. Prendre la prédiction la plus fréquente (classification) ou la moyenne (régression).

### 10.4. Avantages

L'algorithme Random Forest est donc **moins sensible à l'overfitting** que les arbres de décision simples, grâce à l'utilisation de plusieurs arbres qui réduisent la variance du modèle. Il **convient bien aux grands ensembles de données avec de nombreuses caractéristiques**, car il est capable de gérer efficacement des données complexes tout en conservant une bonne performance.

De plus, il est **plus robuste que les arbres de décision individuels**, en raison de la diversité introduite par l'ensemble d'arbres. Enfin, cet algorithme peut **donner des estimations de l'importance des caractéristiques**, permettant de comprendre quelles variables ont le plus d'influence sur les décisions du modèle.

## 10.5. Inconvénients

En revanche, il est **moins interprétable qu'un arbre de décision unique**, car il combine plusieurs arbres, rendant l'analyse du modèle plus complexe.

De plus, il est **plus coûteux en termes de calculs**, tant en termes de **temps que de mémoire**, en raison de la nécessité de construire et de maintenir plusieurs arbres. Si le nombre d'arbres est élevé, **le temps d'entraînement peut devenir considérablement plus long**, ce qui peut poser des problèmes pour des ensembles de données très volumineux ou des applications nécessitant des résultats rapides.

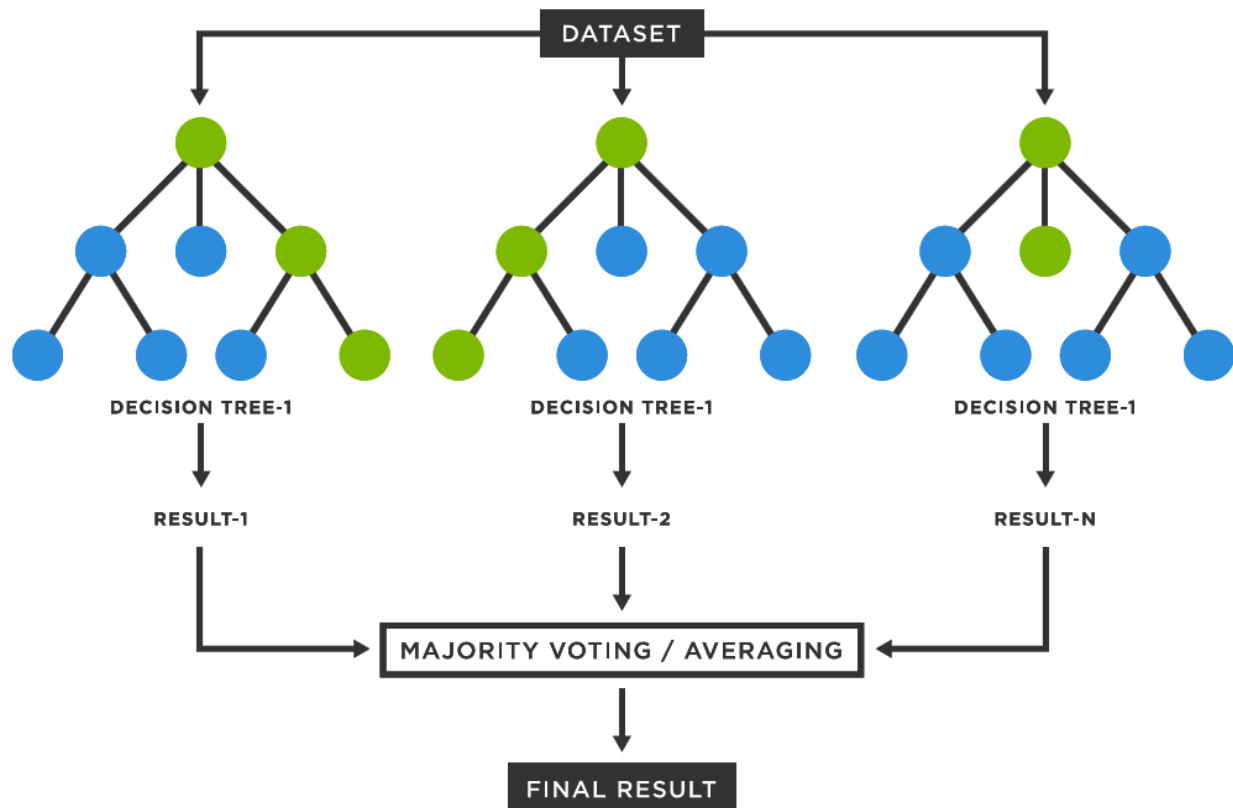


Figure 4 : Fonctionnement de l'algorithme Random Forest

## 11. Support Vector Machine (SVM)

### 11.1. Fonctionnement de l'algorithme

SVM est un algorithme de classification qui cherche à trouver l'hyperplan (dans un espace à  $n$  dimensions) qui sépare les différentes classes de manière optimale. L'objectif est de maximiser la marge, c'est-à-dire la distance entre l'hyperplan et les points les plus proches de chaque classe (appelés vecteurs de support).

### 11.2. Spécificités de l'algorithme

L'algorithme Support Vector Machine repose sur la **maximisation de la marge**, où l'hyperplan optimal est celui qui maximise la distance entre les classes, afin de garantir une séparation aussi large et robuste que possible.

Bien que SVM soit généralement linéaire, il peut également gérer des problèmes non linéaires en utilisant des **noyaux**, qui projettent les données dans un espace de dimension supérieure, où les classes peuvent être séparées de manière linéaire. Des noyaux comme le **noyau gaussien (RBF)**, le **noyau polynomial** ou le **noyau linéaire** peuvent être utilisés pour ajuster la séparation des classes et rendre l'algorithme plus flexible dans des situations complexes.

### 11.3. Algorithme

1. Choisir un noyau pour transformer les données.
2. Trouver l'hyperplan qui maximise la marge entre les classes.
3. Pour les nouvelles données, prédire la classe en fonction du côté de l'hyperplan où elles se trouvent.

### 11.4. Avantages

L'algorithme Support Vector Machine est **très efficace dans des espaces de grande dimension**, ce qui le rend particulièrement adapté aux problèmes où le nombre de caractéristiques est élevé.

Grâce à l'utilisation des **noyaux**, il est également **efficace même avec des données complexes et non linéaires**, permettant de trouver des hyperplans de séparation dans des espaces projetés de dimension supérieure.

De plus, SVM **fonctionne bien même avec un petit nombre d'exemples**, ce qui en fait une option intéressante lorsque les données disponibles sont limitées.

## 11.5. Inconvénients

L'algorithme Support Vector Machine (SVM) est **sensible au choix du noyau** et des **paramètres de régularisation**, ce qui nécessite une bonne optimisation pour obtenir de bonnes performances.

Il peut également être **peu performant avec de très grands ensembles de données**, car il demande une quantité importante de **mémoire et de temps de calcul** pour entraîner le modèle, ce qui peut rendre son utilisation difficile sur de très grands volumes de données.

Enfin, SVM est **moins facile à interpréter** que d'autres modèles, comme les arbres de décision, ce qui peut rendre l'analyse des résultats plus complexe et moins transparente.

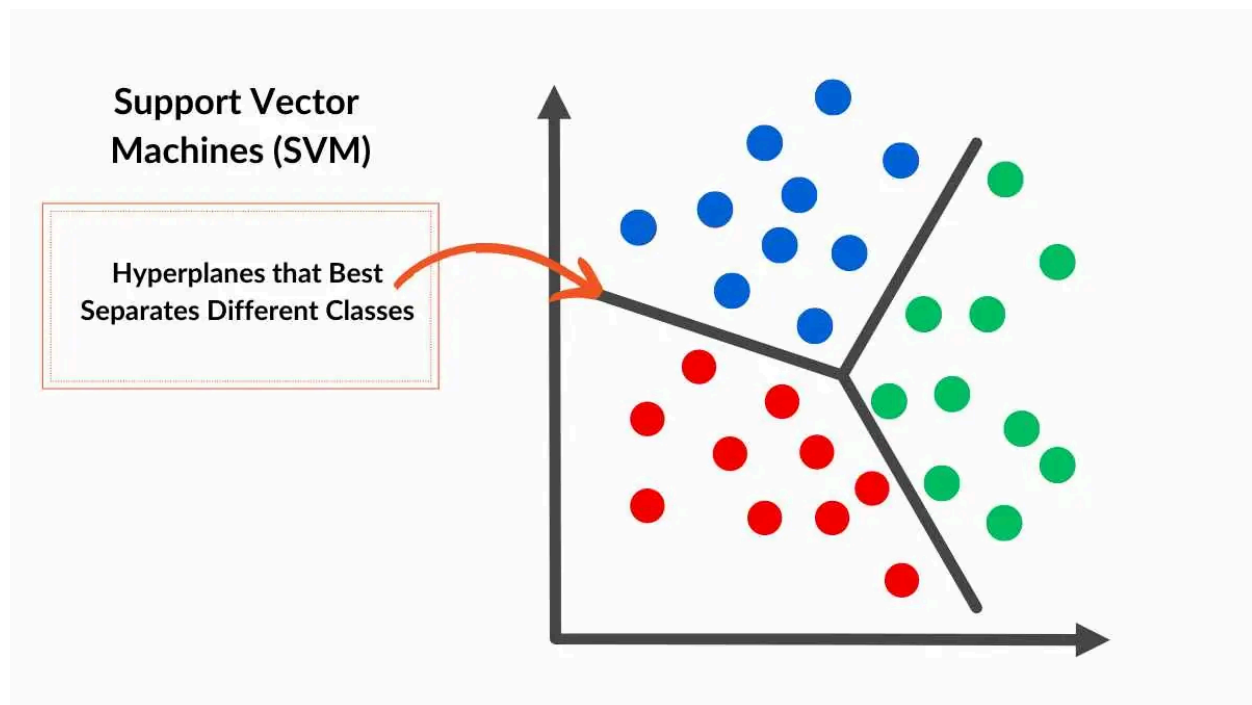


Figure 5 : Fonctionnement de l'algorithme Support Vector Machine

## Conclusion

Ce projet nous a donc permis d'appréhender l'usage du Machine Learning dans un cadre spécifique avec un objectif simple qui consistait à valider une hypothèse d'identification. Le Machine Learning était donc un bon moyen de confirmer ou non cette hypothèse puisque les algorithmes retenus nous ont permis de faire de la prédiction, d'émettre en fonction d'un CFO quelconque.

Au vu des précisions de chaque modèle, présenté dans le graphique ci-dessous, nous avons ainsi pu valider l'hypothèse énoncée en Introduction.

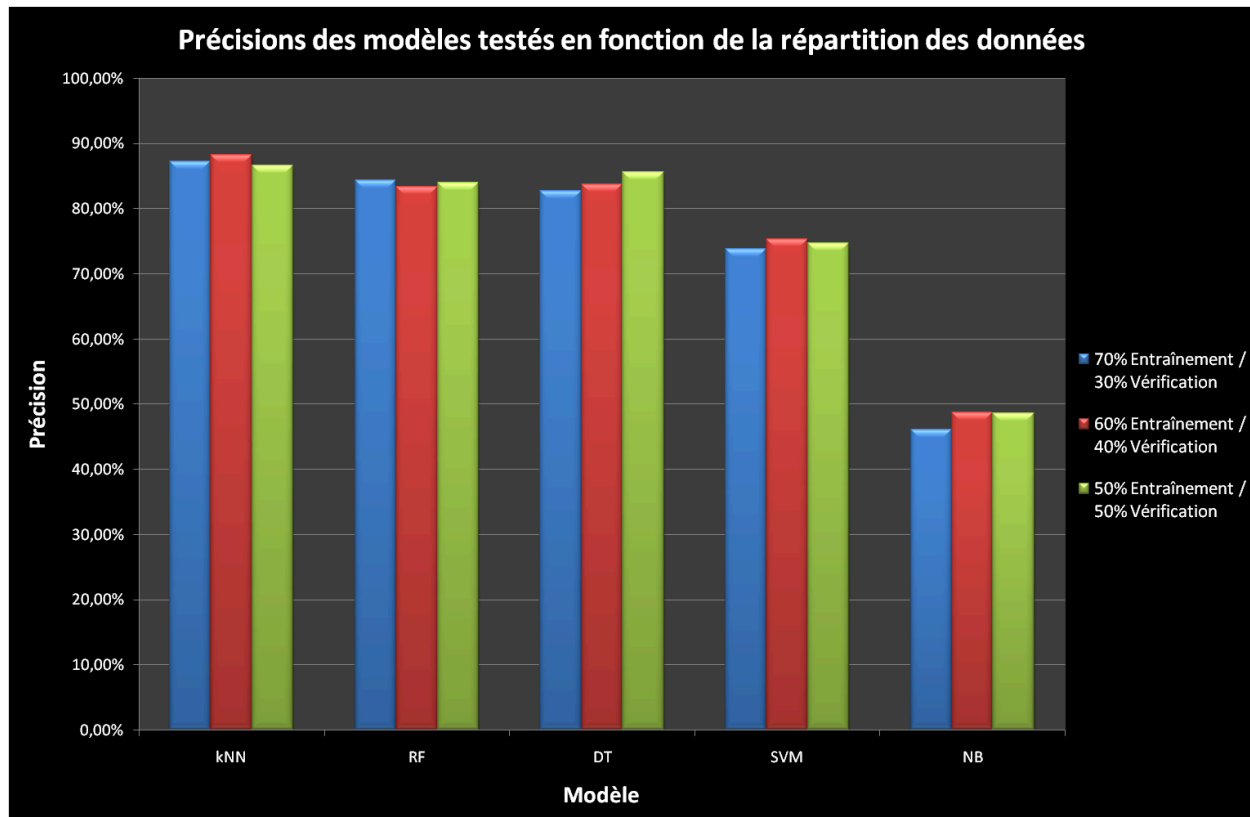


Figure 6 : Précisions de chaque algorithme pour notre projet

Les recherches sur les algorithmes sélectionnés ayant été faites une fois le projet terminé, ces résultats n'ont pas été étudiés pour déterminer s'ils étaient cohérents avec les avantages et inconvénients de chaque algorithme présentés dans le présent document.

Il s'agit donc d'une piste d'amélioration du projet, en plus de celle, évoquée dans le rapport de projet, qui consiste à augmenter le nombre de données de la base utilisée pour le Machine Learning pour voir si les précisions s'améliorent et si un autre algorithme que le kNN se détache du lot.