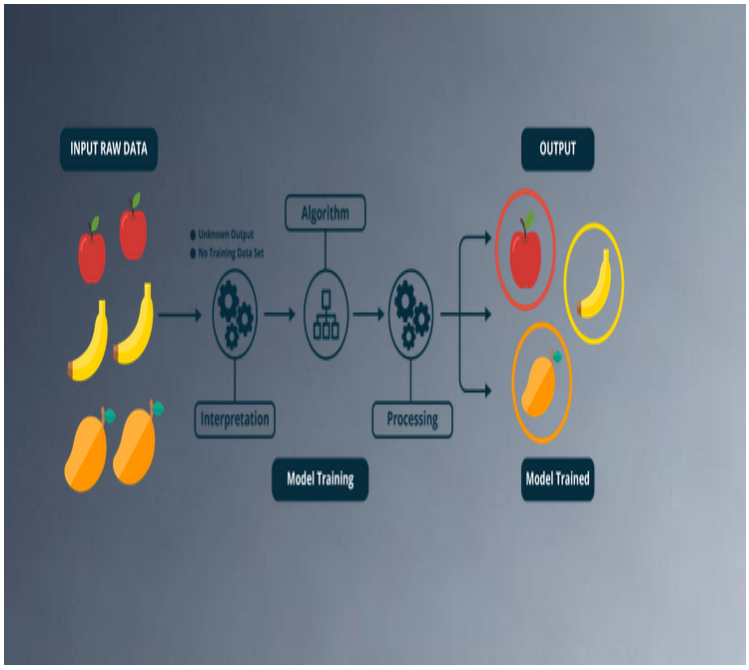
Rapport AI et modélisation

Réalisé par Ziad Bougrine

La première étape d'un projet d'apprentissage automatique est la récupération et l'importation des données. Malheureusement, dans des projets réels, nous obtenons régulièrement des fichiers incomplets, comportant des erreurs. Il est donc crucial, avant même de travailler sur des visualisations ou des algorithmes, de s'assurer d'obtenir des données correctes, et bien étiquetées. L'apprentissage automatique comporte généralement deux phases. La première consiste à estimer un modèle à partir de données, appelées observations, qui sont disponibles et en nombre fini, lors de la phase de conception du système. La seconde phase correspond à la mise en production : le modèle étant déterminé, de nouvelles données peuvent alors être soumises afin d'obtenir le résultat correspondant à la tâche souhaitée.

Selon les informations disponibles durant la phase d'entraînement, l'apprentissage est qualifié de différentes manières. Si les données sont étiquetées (c'est-à-dire que la réponse à la tâche est connue pour ces données), il s'agit d'un apprentissage supervisé. On parle de classification si les étiquettes sont discrètes, ou de régression si elles sont continues. Si le modèle est appris de manière incrémentale en fonction d'une récompense reçue par le programme pour chacune des actions entreprises, on parle d'apprentissages par renforcement. Dans le cas le plus général, sans étiquette, on cherche à déterminer la structure sous-jacente des données (qui peuvent être une densité de probabilité) et il s'agit alors d'apprentissage non supervisé.



Etude du problème

Nous voulons modéliser un modèle pour prédire des types de fleurs de la base de données iris et comparer entre trois modèles KNN, SVC, RandomForest

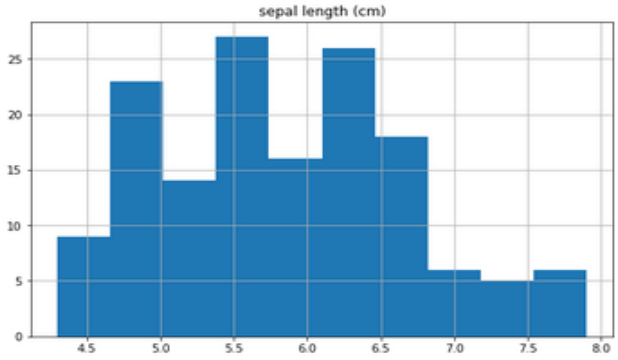
Prédiction avec le model KNN : **le modèle de prédiction le plus intuitif consiste à chercher pour chaque nouvelle plante le type palnte qui lui ressemble le plus parmi tous ceux connus. On appelle cette méthode la méthode des plus proches voisins. Le module scikit-learn implémente cet algorithme Nearest Neighbors et on pourra s'inspirer de l'exemple Nearest Neighbors regression.**

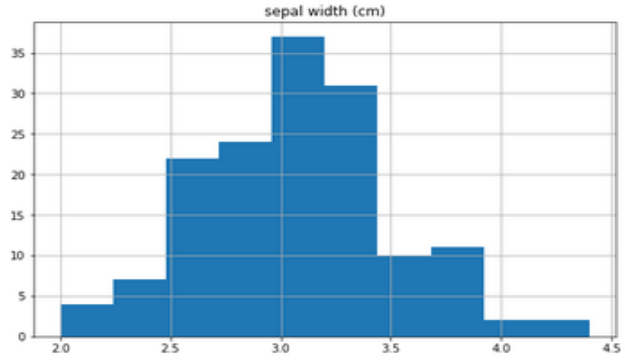
Le modèle RandomForest : **L'algorithme des forêts aléatoires (ou Random Forest en anglais) proposé par Leo Breiman en 2001 est l'un des algorithmes les plus utilisés aujourd'hui pour traiter des problème de classification. Il s'agit d'un algorithme qui comme son nom l'indique combine plusieurs arbres de décisions dans une approche de type bagging3. L'algorithme entraîne plusieurs arbres de décisions sur des samples sélectionnés aléatoirement. La prédiction de l'algorithme est calculée à partir de la moyenne des des prédictions de chaque arbre de décision construit avec des données sont quantitatives. Pour les données qualitatives, l'algorithme utilise la moyenne des prédiction des modèles indépendants et procède à un vote pour déterminer sa prédiction, comme l'illustre la figure II.2 ci-dessous.**

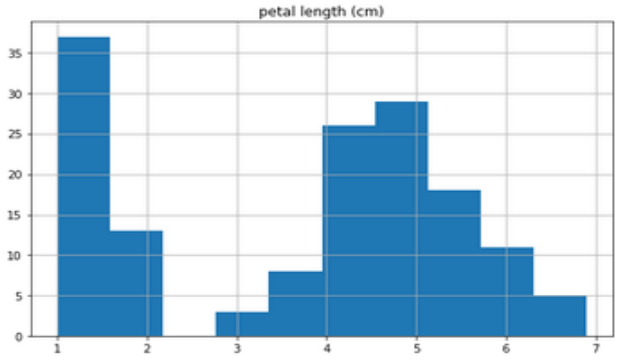
Le SVM **appartient à la catégorie des classificateurs linéaires (qui utilisent une séparation linéaire des données), et qui dispose de sa méthode à lui pour trouver la frontière entre les catégories. Pour que le SVM puisse trouver cette frontière, il est nécessaire de lui donner des données d'entraînement. En l'occurrence, on donne au SVM un ensemble de points, dont on sait déjà le type. Pour l'obtention de la droite optimale, un SVM va placer la frontière aussi loin que possible des types de fleurs presents dans notre exemple. Les points d'entraînement les plus proches de la frontière sont appelés vecteurs support, d'ou vien le nom : SVM signifie Support Vector Machine, ou Machines à Vecteur Support en français.**

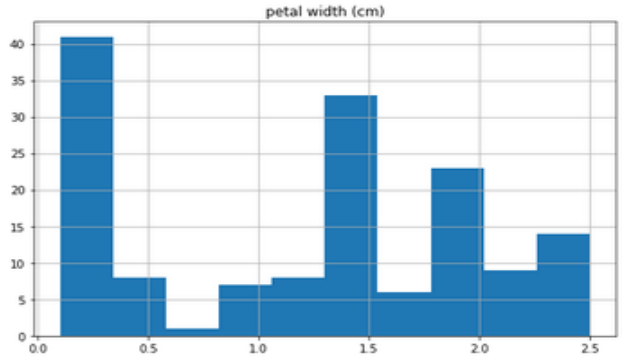
Statistiques histogramme pour bien décrire notre modèle

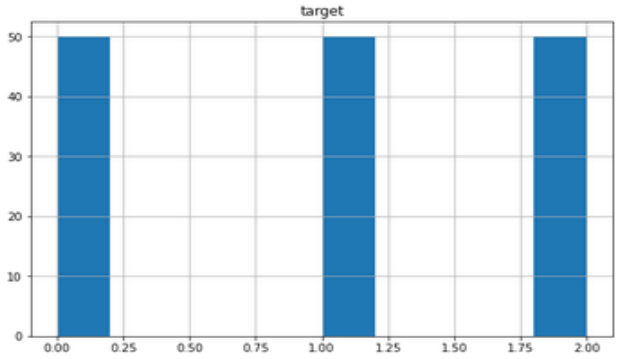
```
array([[<AxesSubplot:title={'center':'sepal length (cm)'}>,
        <AxesSubplot:title={'center':'sepal width (cm)'}>],
       [<AxesSubplot:title={'center':'petal length (cm)'}>,
        <AxesSubplot:title={'center':'petal width (cm)'}>],
       [<AxesSubplot:title={'center':'target'}>, <AxesSubplot:>]],
      dtype=object)
```



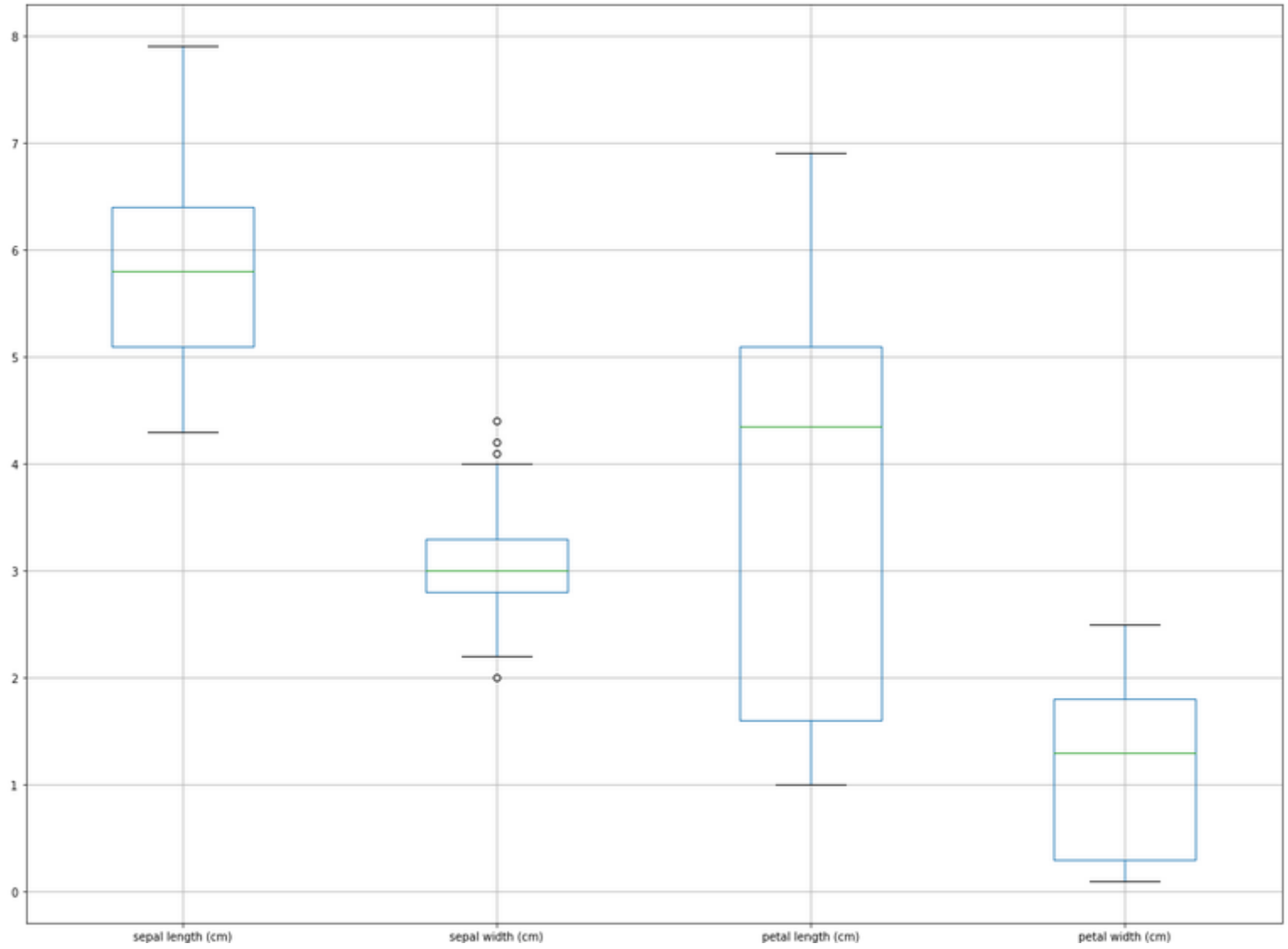








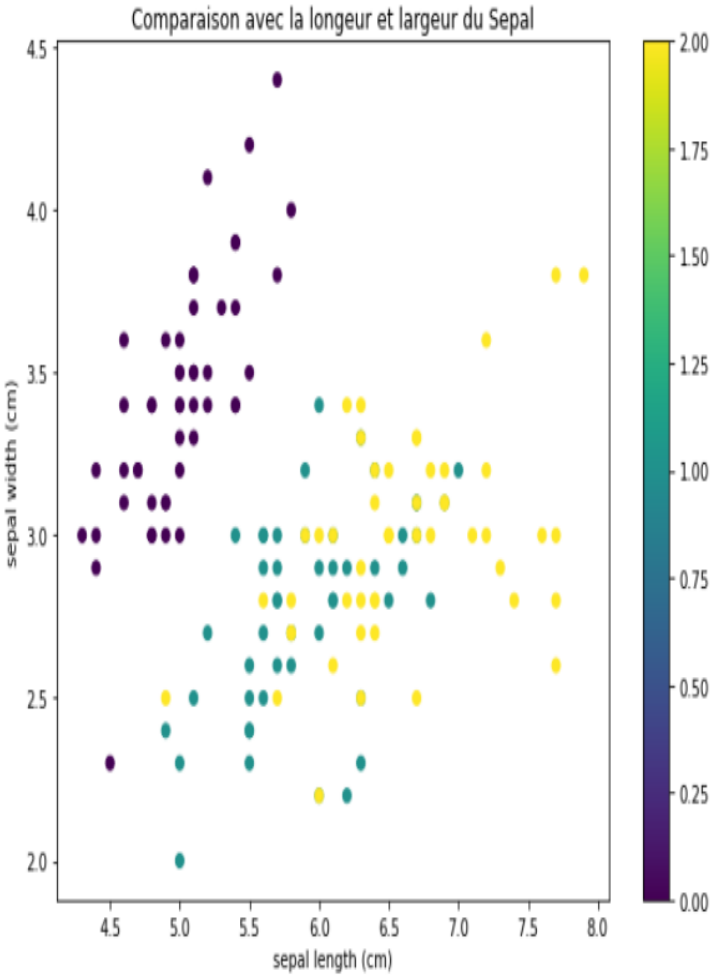
Statistiques de dispersion de nos données

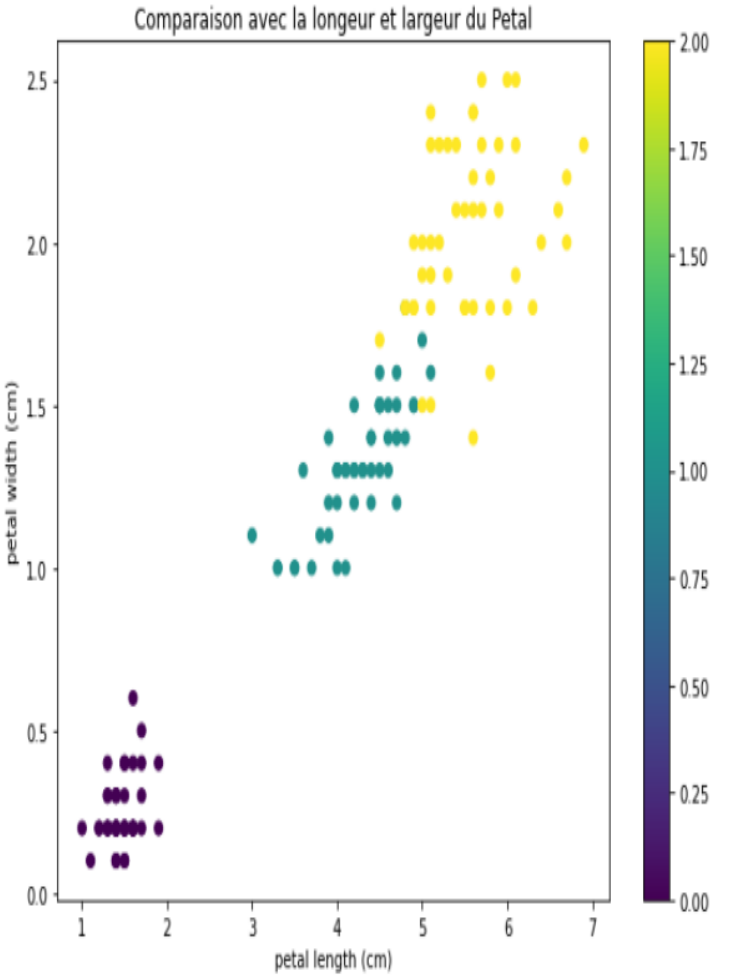


Statistiques numériques

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

Relation entre les caractéristiques des plantes





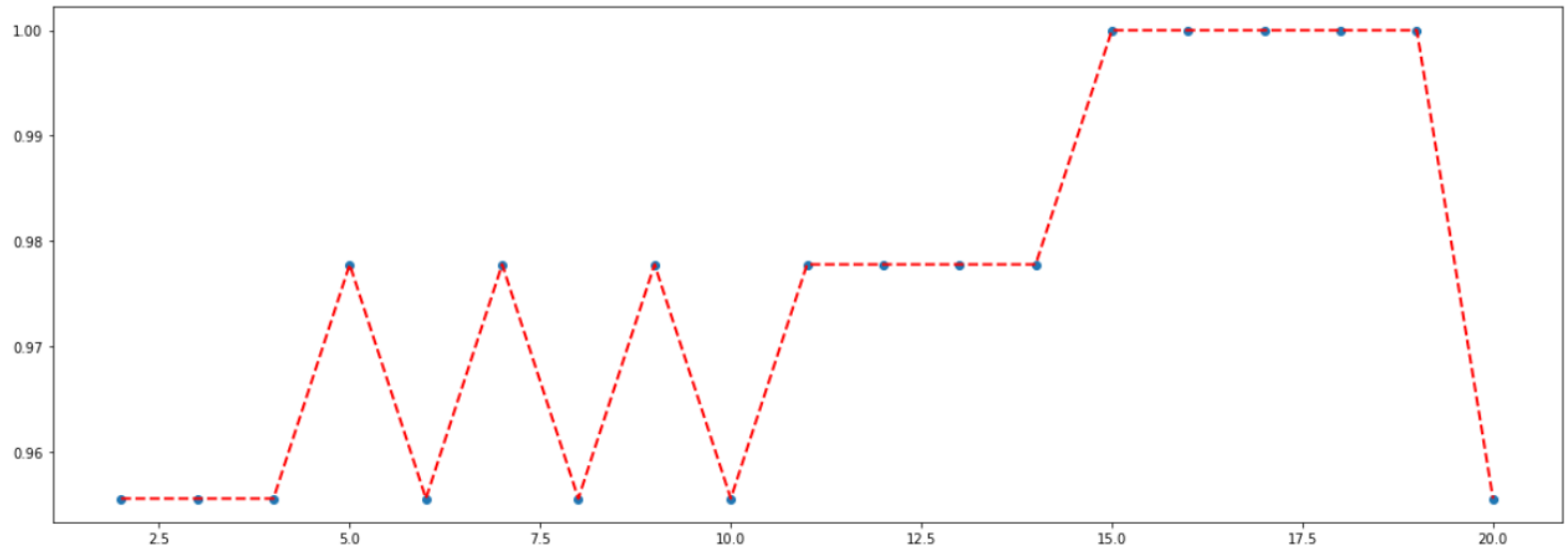
Techniques utilisées

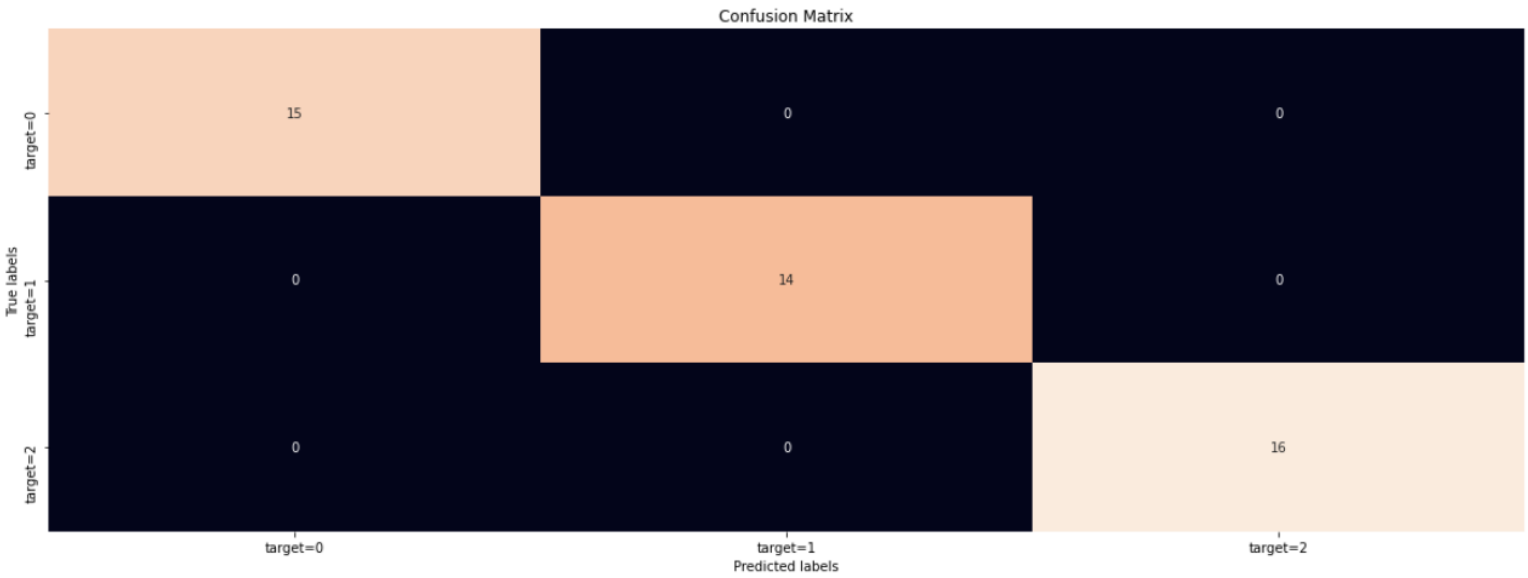
Pour entraîner mes modèles avant de faire la comparaison entre eux, il faut trouver le paramètre compatible pour chaque modèle entre eux, donc j ‘ai utilisé un algorithme qui boucle et essaye pour chaque modèle tous les combinaisons des paramètres compatibles

Modèle KNN

Pour le modèle KNN, nous avons besoin de trouver le meilleur nombre de voisins pour avoir un modèle puissant, mon cas, le k est égal à 15

```
[<matplotlib.lines.Line2D at 0x21d0ae0ea90>]
```



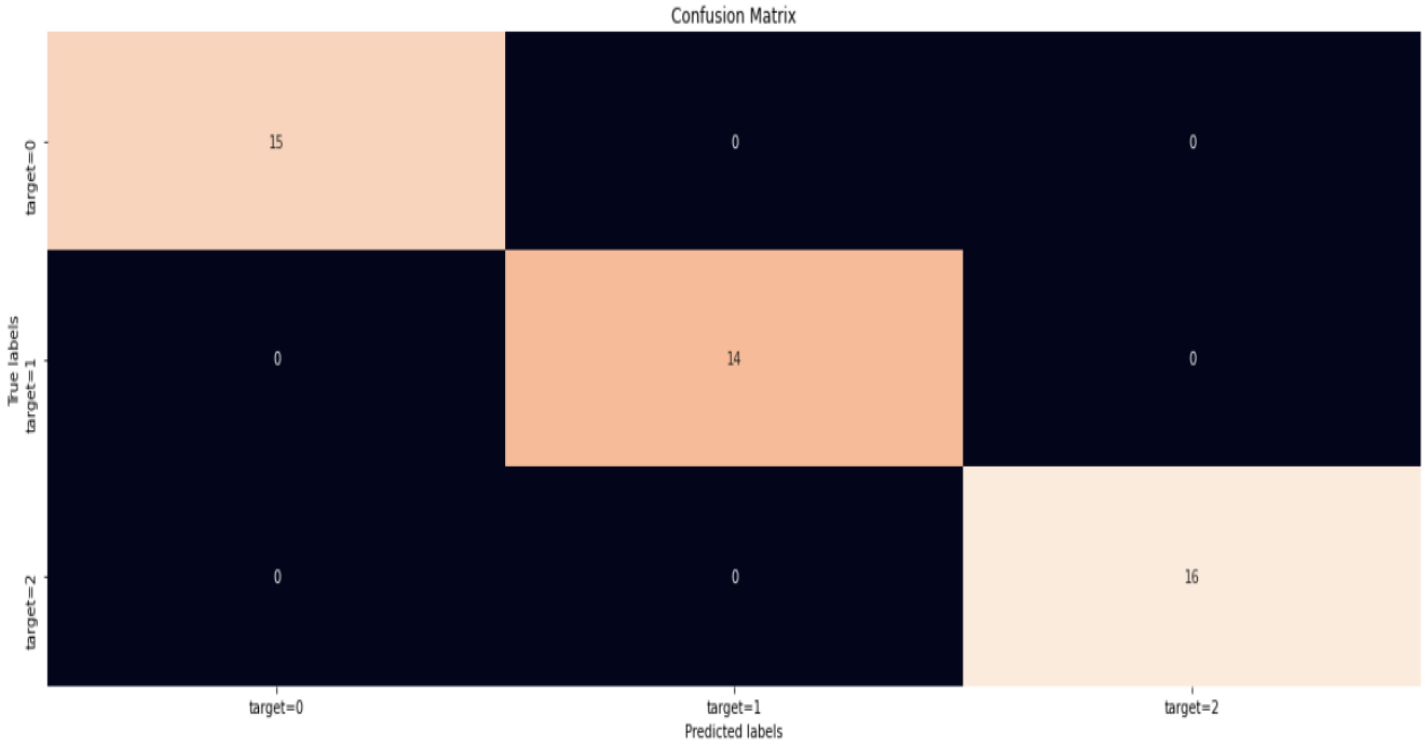


Modèle SVC

Pour notre Supported Vector Classifier, nous avons besoin d'un gamma très petit et d'un noyau pour bien estimé

```
le kernel linear a pour max gamma 0.001 et pour score 1.0
le kernel poly a pour max gamma 0.033 et pour score 1.0
le kernel rbf a pour max gamma 0.021 et pour score 0.9777777777777777
le kernel sigmoid a pour max gamma 0.003 et pour score 0.6444444444444445
```

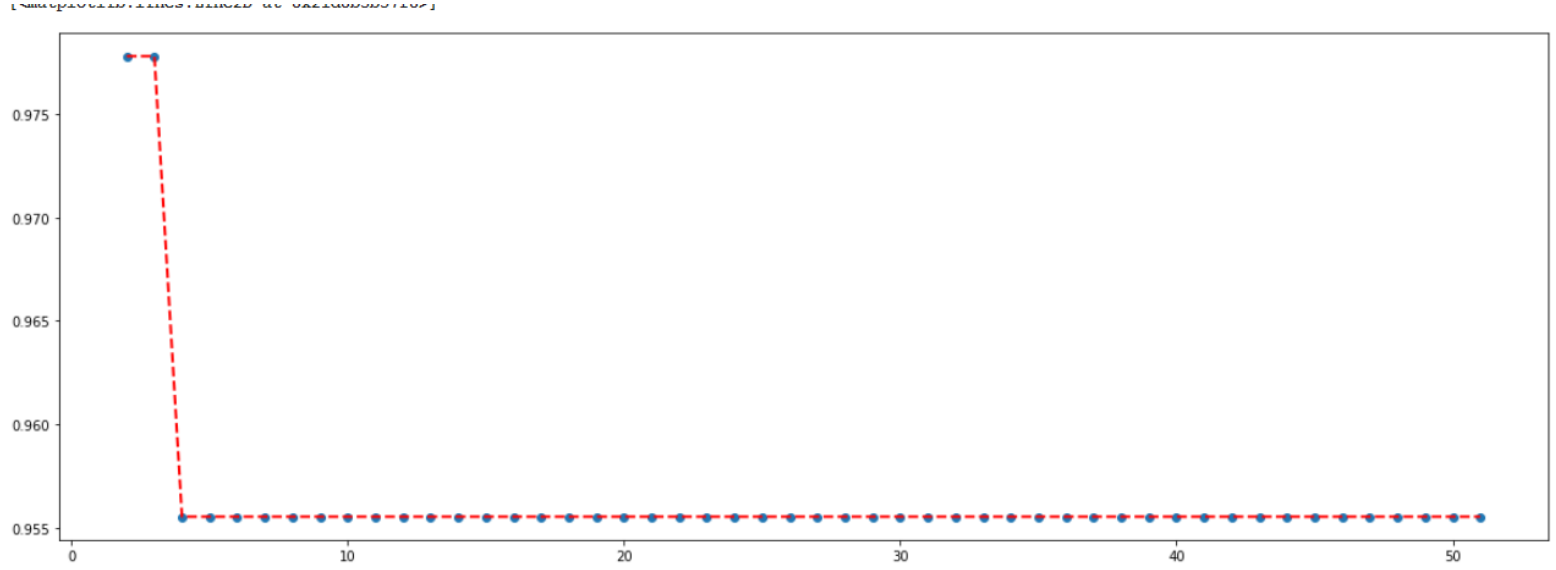
Le meilleur cas pour le Grid Search est gamma 0.008 et kernel est rbf, on peut aussi choisir le poly kernel

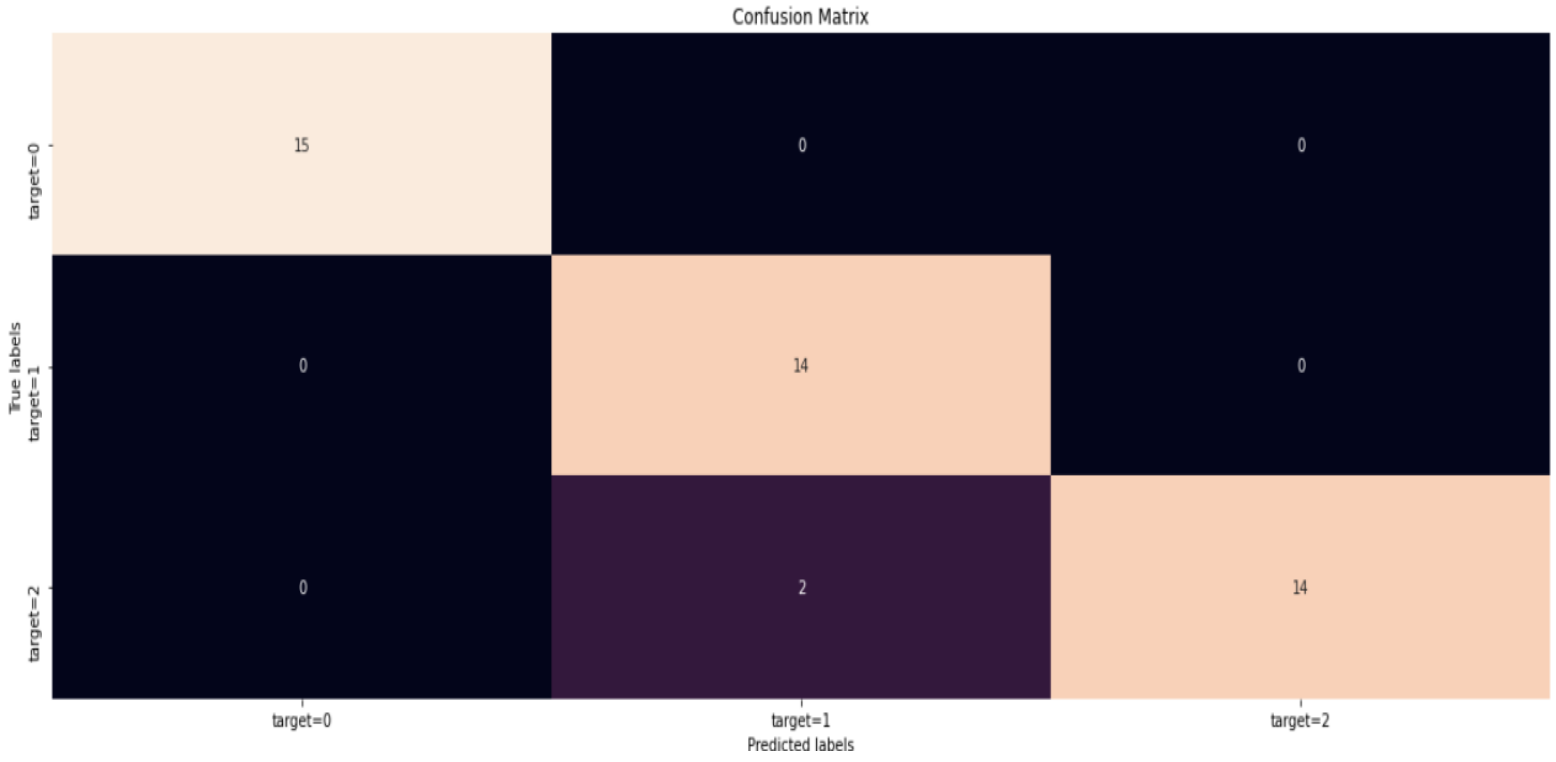


Modèle RandomForest

Pour notre RandomForest Classifier, nous avons besoin d'un nombre d'estimateur pour avoir une bonne presicion

Pour mon cas nombre estimateur = 2





Comparaison entre les modèles

KNN :

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	15
1.0	1.00	1.00	1.00	14
2.0	1.00	1.00	1.00	16
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

RandomForest

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	15
1.0	0.88	1.00	0.93	14
2.0	1.00	0.88	0.93	16
accuracy			0.96	45
macro avg	0.96	0.96	0.96	45
weighted avg	0.96	0.96	0.96	45

SVC

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	15
1.0	1.00	1.00	1.00	14
2.0	1.00	1.00	1.00	16
accuracy			1.00	45
macro avg	1.00	1.00	1.00	45
weighted avg	1.00	1.00	1.00	45

Conclusion :

Je prends le modèle SVC et le KNN car ils ont une meilleure précision mieux que le RandomForest

Neural Network

Un **réseau de neurones artificiels**, ou **réseau neuronal artificiel**, est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques. Les réseaux de neurones sont généralement optimisés par des méthodes d'apprentissage de type probabiliste, en particulier bayésien. Ils sont placés d'une part dans la famille des applications statistiques, qu'ils enrichissent avec un ensemble de paradigmes permettant de créer des classifications rapides (réseaux de Kohonen en particulier), et d'autre part dans la famille des méthodes de l'intelligence artificielle auxquelles ils fournissent un mécanisme perceptif indépendant des idées propres de l'implémentera, et des informations d'entrée au raisonnement logique formel