

Plan du cours

1) Introduction au machine learning

2) Régularisation et forêts aléatoires

3) Réseau de neurones

4) Réseau de neurones convolutifs

Pour chaque séance:

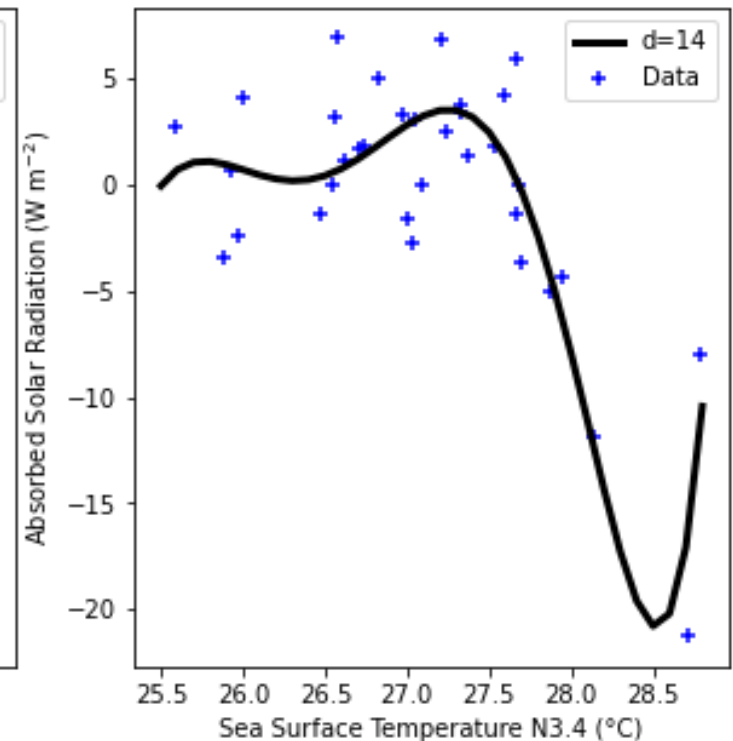
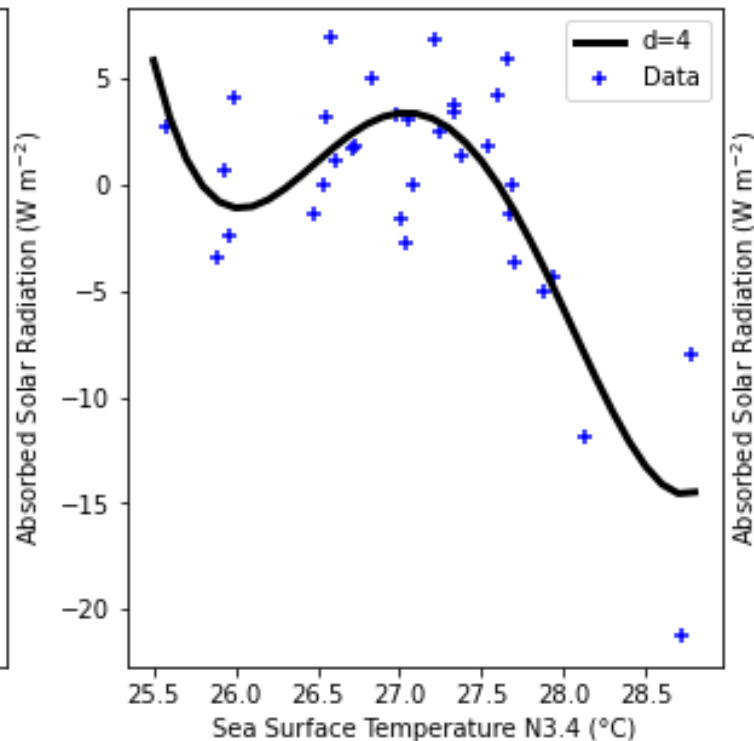
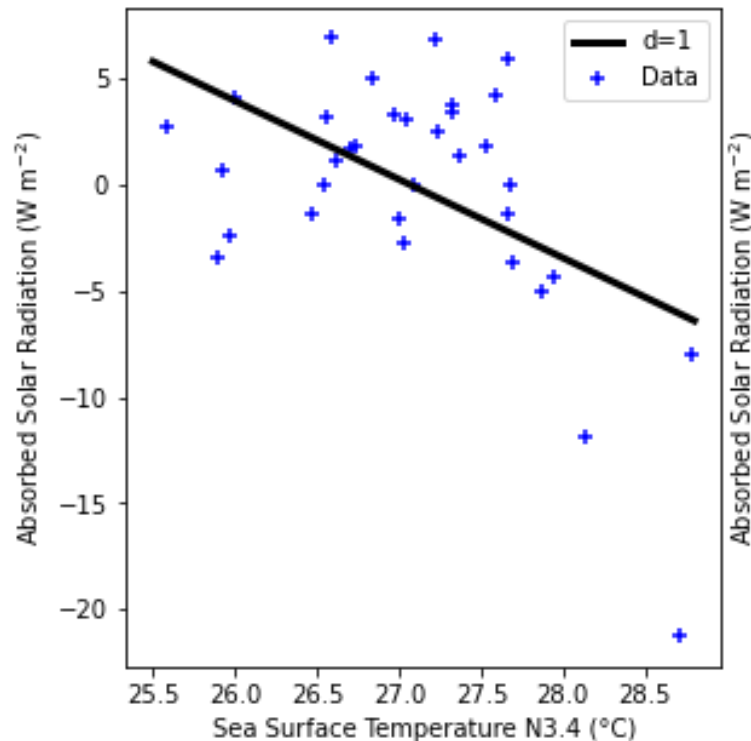
1h de cours / support transparent

2h de travaux pratiques (amener un ordinateur portable)

Régularisation

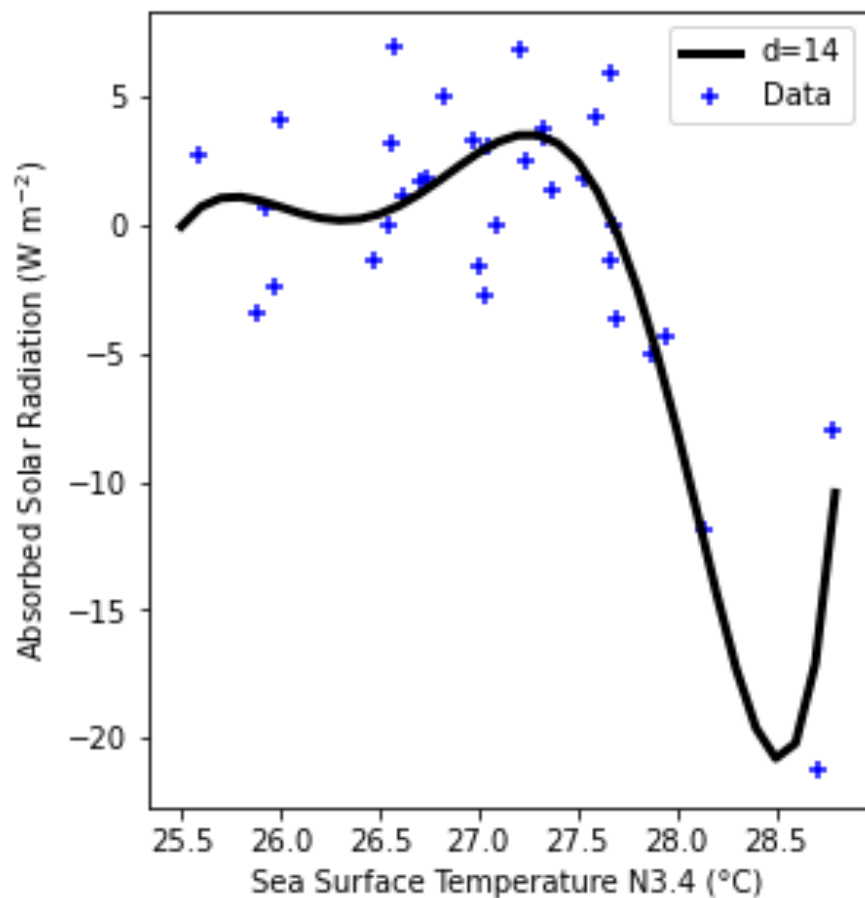
Reprenons l'exemple de la régression **polynomiale** :

$$\hat{y}_i = \theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \dots + \theta_d x_i^d$$



Régularisation

- Dans une situation de sur-apprentissage, le modèle cherche à expliquer la partie aléatoire du signal (le bruit).
- Les paramètres ont alors tendance à augmenter.



d	θ_1	θ_2	θ_3	θ_4	...
1	-3.7				
2	2.2e2	-4.2			
3	-1.7e3	6.8e1	-8.9e-1		
4	-2.4e5	1.3e4	-3.3e2	3.0	
...					

Régularisation

- La méthode des moindres carrés est basée sur la minimisation de: $\sum_{i=1}^n (\hat{y}_i - y_i)^2$
- En machine learning, on cherche ainsi à minimiser une fonction mathématique pour obtenir les paramètres θ . Celle-ci s'appelle **la fonction cout** et est notée $J(\theta)$.

$$\theta = \underset{\theta}{\operatorname{argmin}} J(\mathbf{y}, \mathbf{X}, \theta)$$

- Pour réduire le risque de sur-apprentissage, on peut effectuer une **régularisation**. L'idée est de *pénaliser* les modèles avec des grands paramètres.
- Ainsi, pour une régression polynomiale au lieu de chercher à réduire:

$$J(\theta) = \sum_{i=1}^n (\hat{y}_i - y_i)^2 = L_2(\hat{\mathbf{y}} - \mathbf{y})$$

On réduit:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 + \alpha P(\theta)$$

Régularisation

$$J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 + \alpha P(\boldsymbol{\theta})$$

Il existe plusieurs types de régularisation:

- Régularisation Ridge (L_2) avec : $P(\boldsymbol{\theta}) = \sum_{i=0}^n \theta_i^2$
- Régularisation Lasso (L_1) avec : $P(\boldsymbol{\theta}) = \sum_{i=0}^n |\theta_i|$
- Régularisation Elastic Net combinant Ridge et Lasso

α est un hyperparamètre.

Estimation des paramètres

En machine learning les paramètres sont le plus souvent estimés à l'aide d'une descente de gradient.

Il s'agit d'un problème d'optimisation, car on souhaite minimiser $J(\boldsymbol{\theta})$. Dans certains cas, il n'existe pas de solutions directes pour ce problème. Il faut alors faire une **descente de gradient**.

Pour cela, on calcule : $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial J}{\partial \theta_0} \\ \vdots \\ \frac{\partial J}{\partial \theta_m} \end{pmatrix}$

Estimation des paramètres

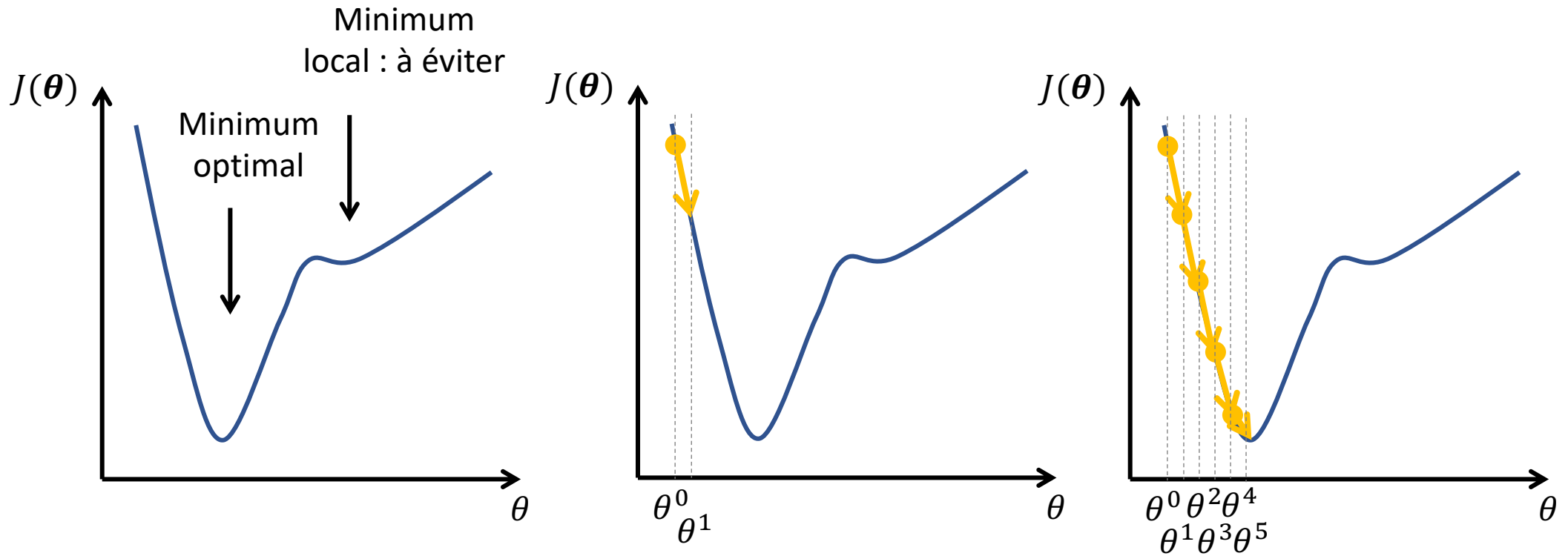
La descente de gradient est un processus itératif qui consiste à modifier successivement les paramètres θ .

On part de valeurs aléatoires pour les θ_i , appelés θ_i^0 .

On modifie alors les paramètres par : $\theta^{k+1} = \theta^k - \eta \nabla_{\theta} J(\theta)$

Le pas utilisé ici est noté η ($\eta > 0$) et est le *learning rate*.

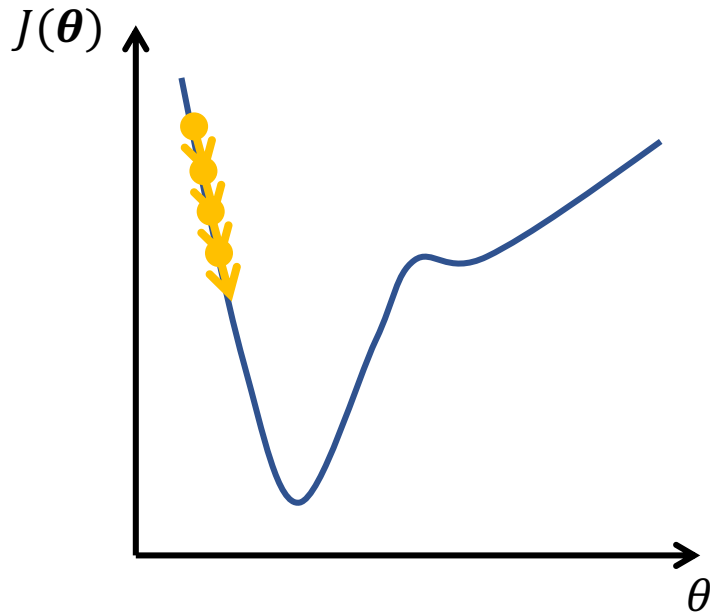
Illustration : descente de gradient



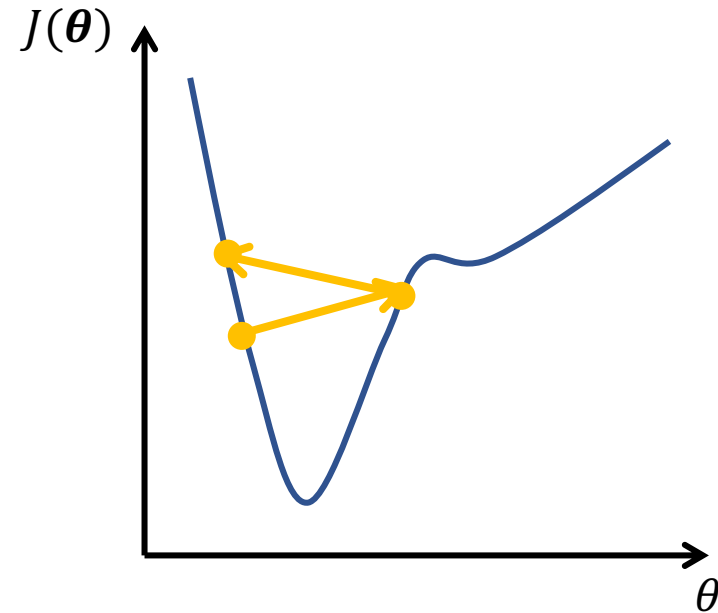
On a : $\frac{\partial J}{\partial \theta} < 0$ donc $-\eta \frac{\partial J}{\partial \theta} > 0$

Au bout de 5 itérations...

Illustration : descente de gradient



Si learning rate trop petit :
convergence plus lente et risque
d'obtenir un minimum local.



Si learning rate trop grand : plus de
convergence vers le minimum local.

Régularisation Ridge

$$J(\boldsymbol{\theta}) = J_{sr}(\boldsymbol{\theta}) + \alpha \sum_{i=0}^m \theta_i^2$$

↑
Fonction loss sans régularisation

La descente de gradient est alors implémentée avec :

$$\begin{aligned} \boldsymbol{\theta}^{k+1} &= \boldsymbol{\theta}^k - \eta (\nabla_{\boldsymbol{\theta}} J_0(\boldsymbol{\theta}) + 2\alpha \boldsymbol{\theta}^k) \\ \boldsymbol{\theta}^{k+1} &= \boldsymbol{\theta}_{sr} - 2\alpha \boldsymbol{\theta}^k \end{aligned}$$

↑
Param. sans régularisation

Régularisation Lasso

$$J(\boldsymbol{\theta}) = J_{sr}(\boldsymbol{\theta}) + \alpha \sum_{i=0}^m |\theta_i|$$

↑
Fonction loss sans régularisation

← Fonction non-dérivable
en $\theta_i = 0$

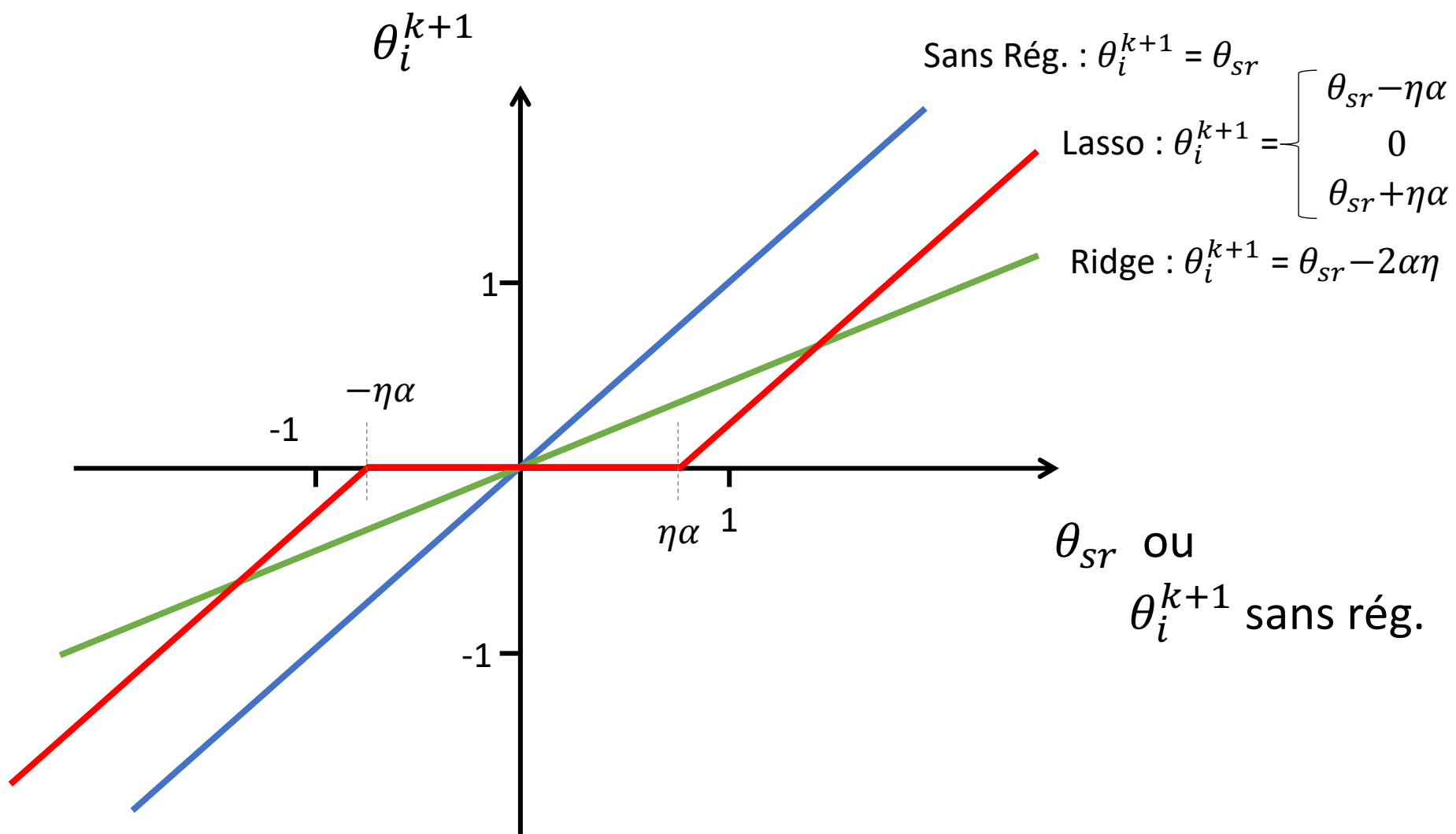
On appelle $\boldsymbol{\theta}_{sr}$ une itération des paramètres obtenus sans régularisation:

$$\boldsymbol{\theta}_{sr} = \boldsymbol{\theta}^k - \eta \nabla_{\boldsymbol{\theta}} J_{sr}(\boldsymbol{\theta})$$

La descente de gradient est alors implémentée avec :

$$\begin{aligned} \theta_i^{k+1} &= \theta_i^k - \eta \frac{\partial J_{sr}}{\partial \theta_i} - \eta \alpha = \theta_{sr} - \eta \alpha & \text{si } \theta_{sr} > \eta \alpha \\ \theta_i^{k+1} &= 0 & \text{si } -\eta \alpha < \theta_{sr} < \eta \alpha \\ \theta_i^{k+1} &= \theta_i^k - \eta \frac{\partial J_{sr}}{\partial \theta_i} + \eta \alpha = \theta_{sr} + \eta \alpha & \text{si } \theta_{sr} < -\eta \alpha \end{aligned}$$

Illustration



Régularisation

Utilisé en machine learning pour trouver les paramètres.

Ridge (L_2) :

- Evite le surapprentissage
- Utilise toutes les caractéristiques de X et peut être inutile dans le cas où X a beaucoup de caractéristiques.

Lasso (L_1) :

- Réduit le nombre de prédicteurs et donc de caractéristiques utilisés,
- Dans le cas où plusieurs caractéristiques sont corrélées, sélectionne arbitrairement une de celles-ci.

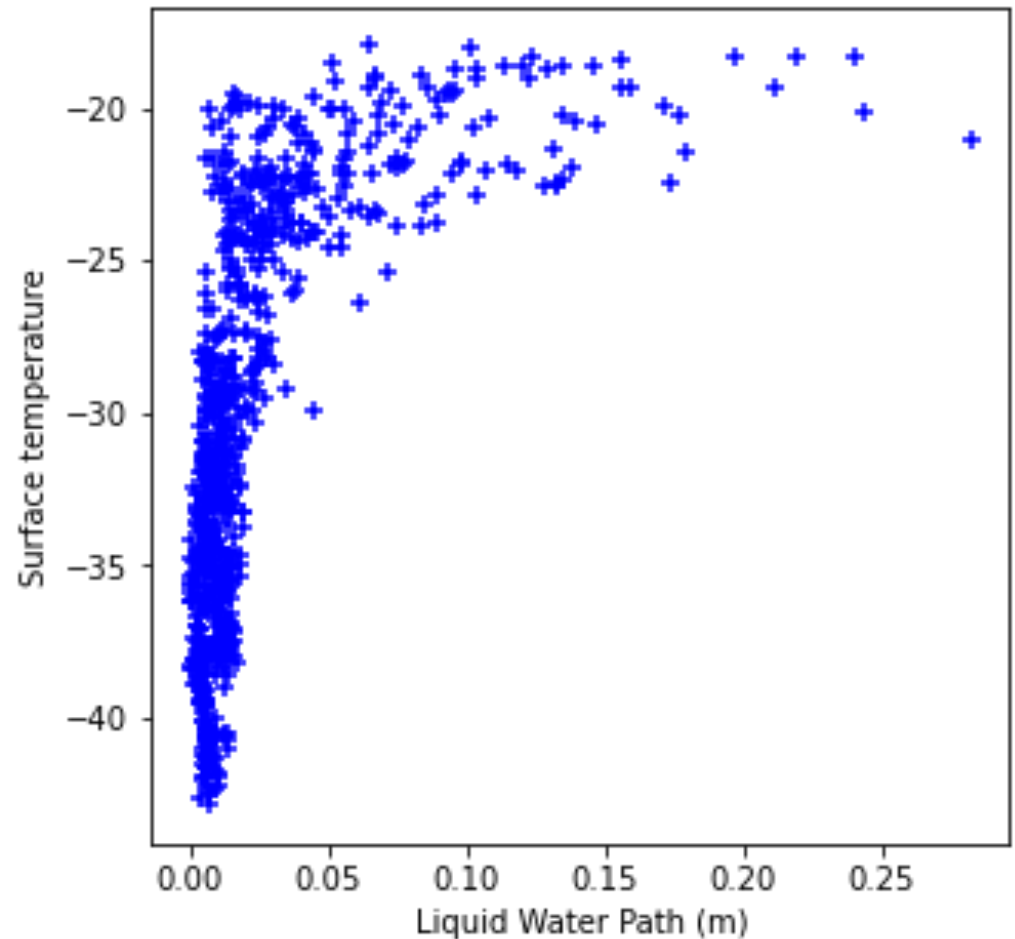
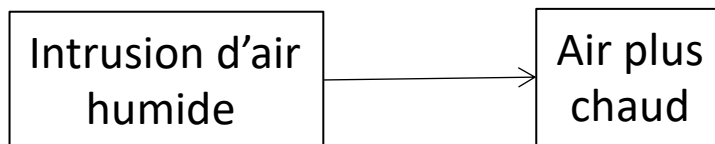
Forêt aléatoire

- Méthode pouvant utiliser des variables qualitative et quantitatives en tant que features. Peut être utilisé pour de la classification ou de la régression.
- Méthode basée sur des arbres de décision.

Illustration

Données: campagne SHEBA en Arctique.

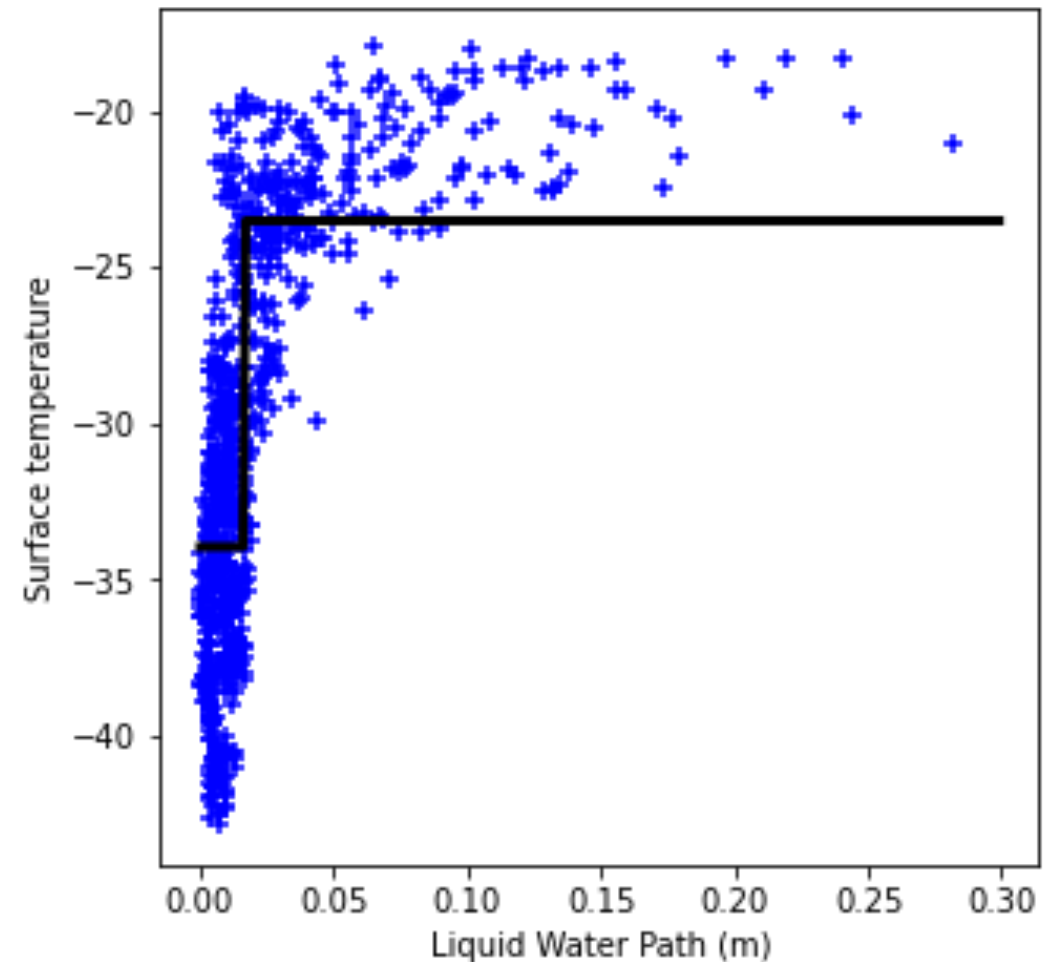
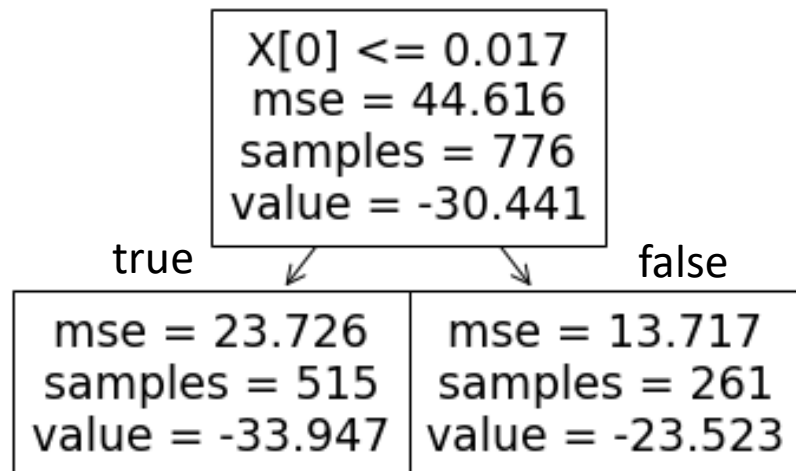
- X = Contenne en vapeur d'eau de l'atmosphère.
- y = Température de surface observée



Illustration

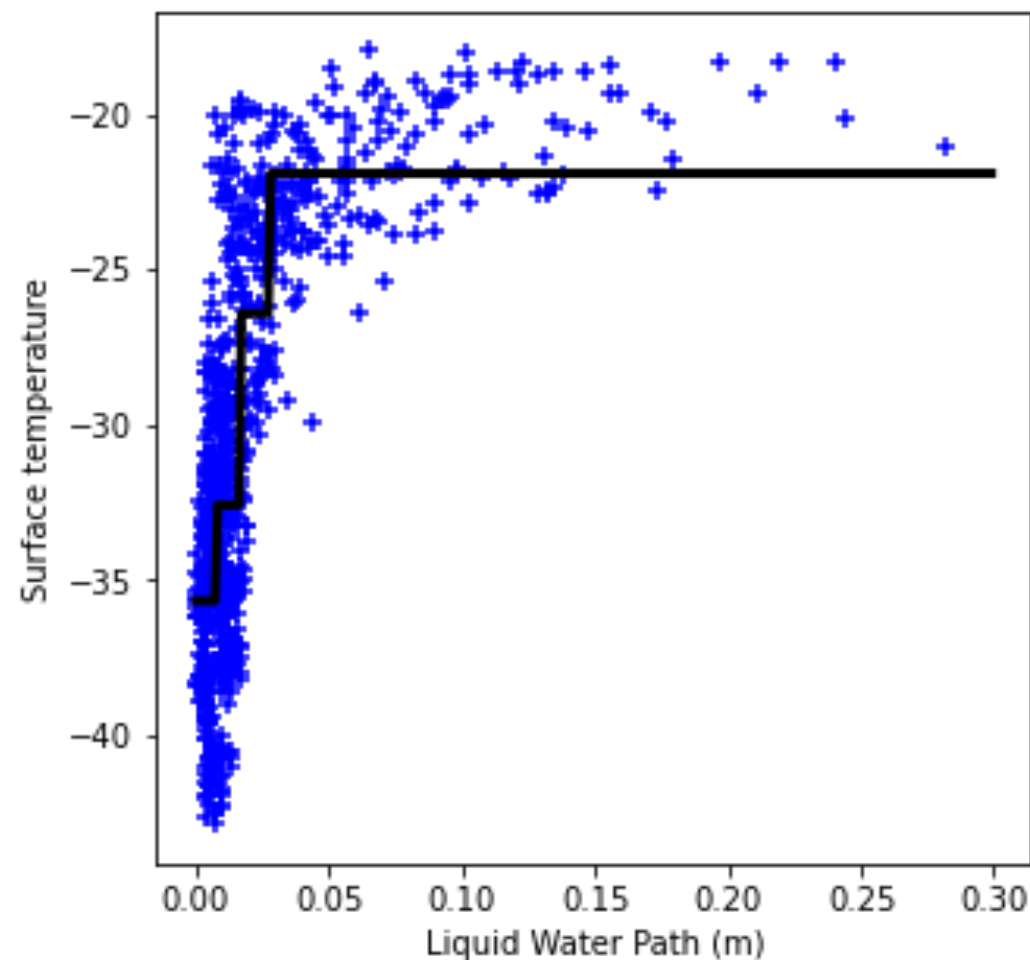
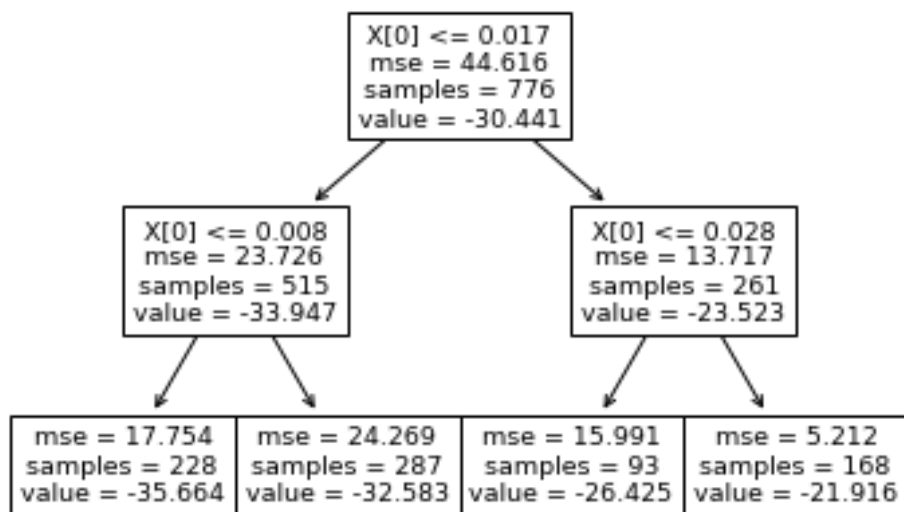
On construit un arbre de décision à partir des valeurs de X.

-> on les construit à partir de l'Entropie et de la réduction d'information effectuée.



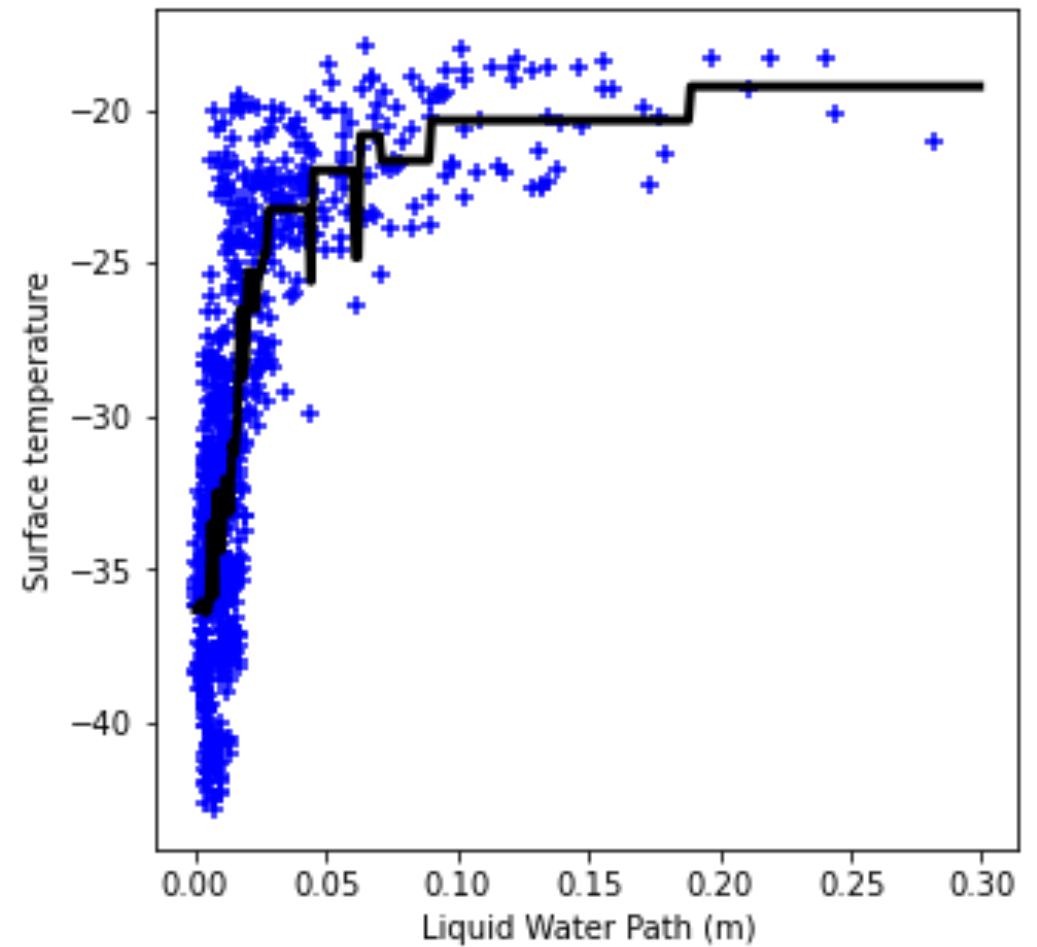
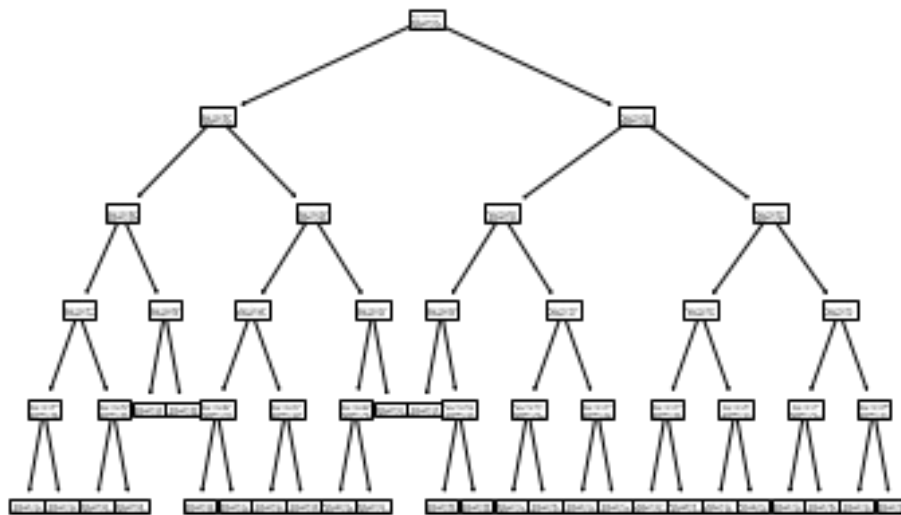
Illustration

Augmentons la profondeur de l'arbre à 2.



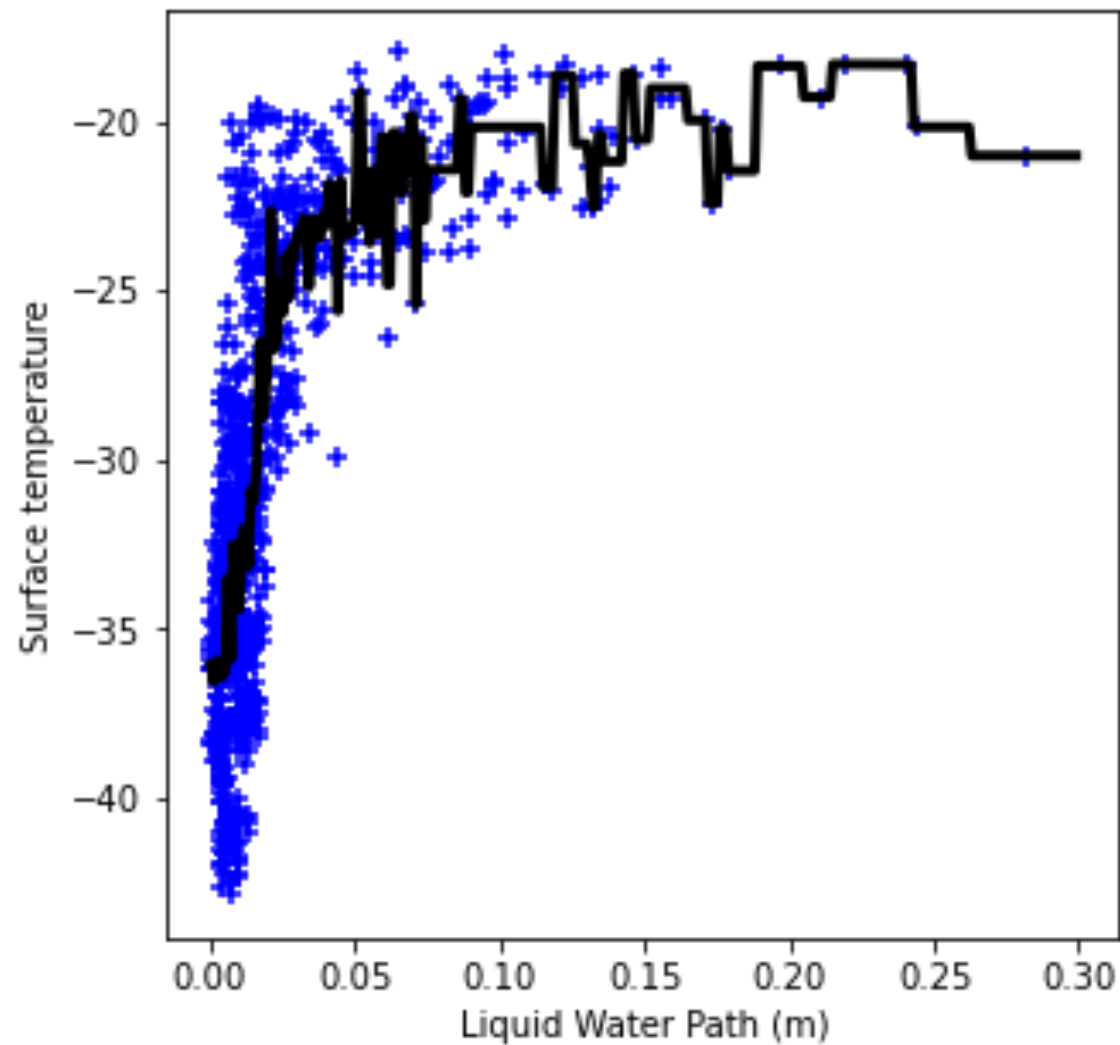
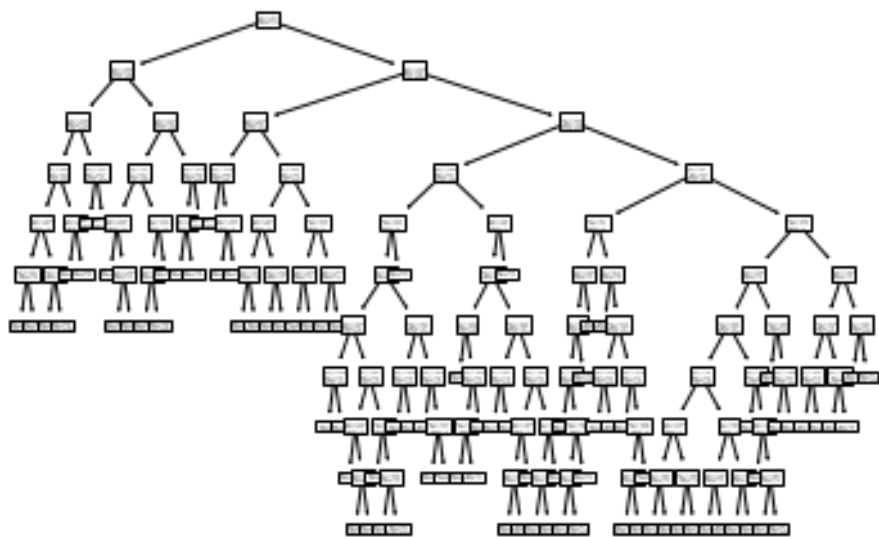
Illustration

Profondeur = 5



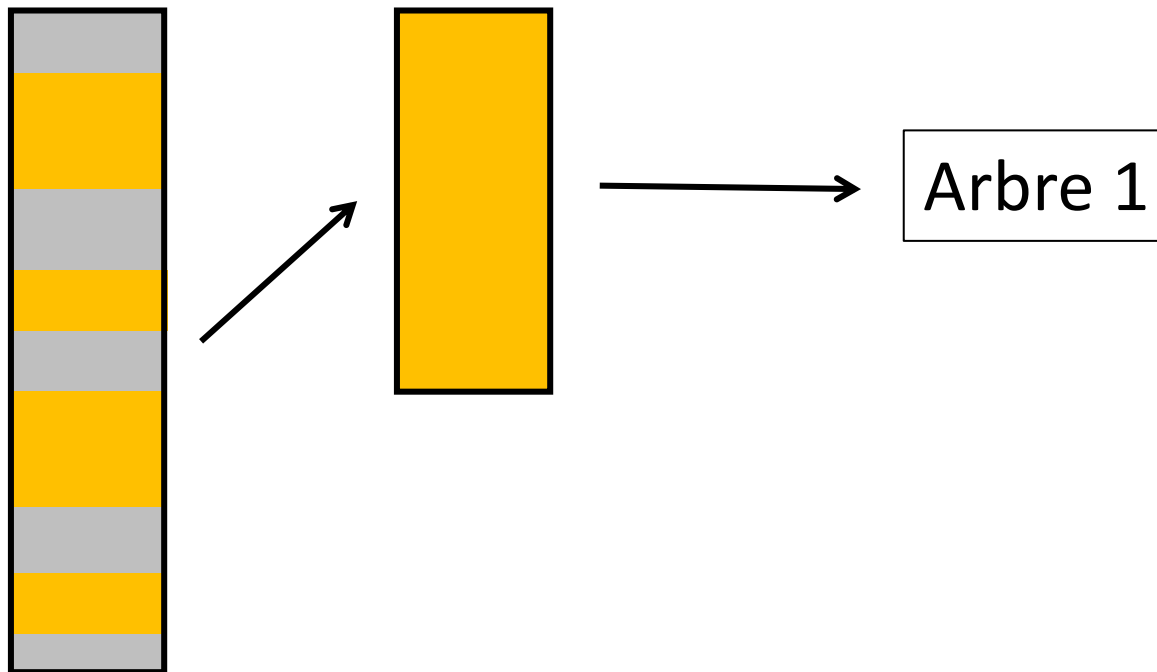
Illustration

Profondeur = 10



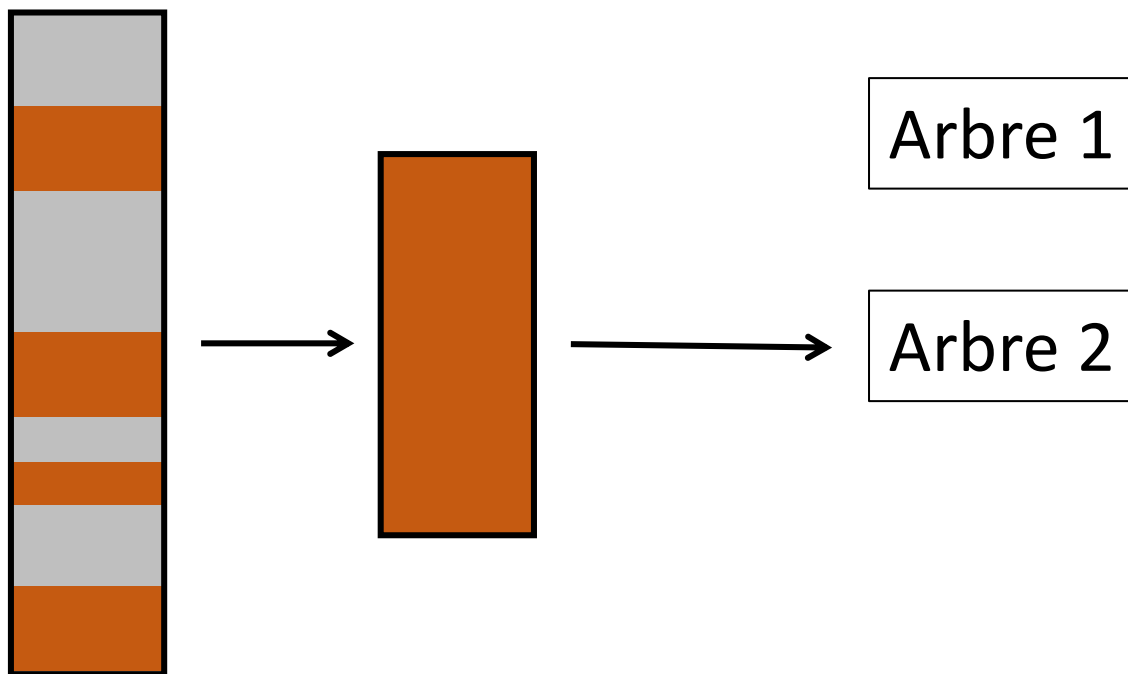
Arbre de décision et forêt aléatoire

- Les arbres décisionnels ont l'inconvénient de surapprendre très vite.
- La solution est la **forêt aléatoire**.



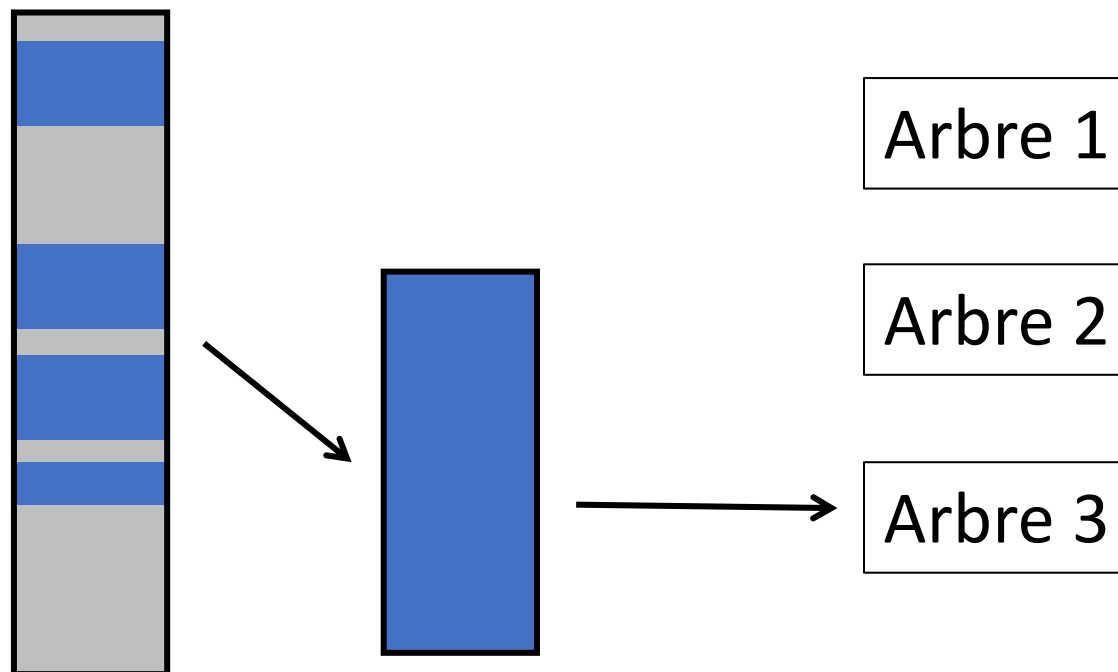
Arbre de décision et forêt aléatoire

- Les arbres décisionnels ont l'inconvénient de surapprendre très vite.
- La solution est la **forêt aléatoire**.



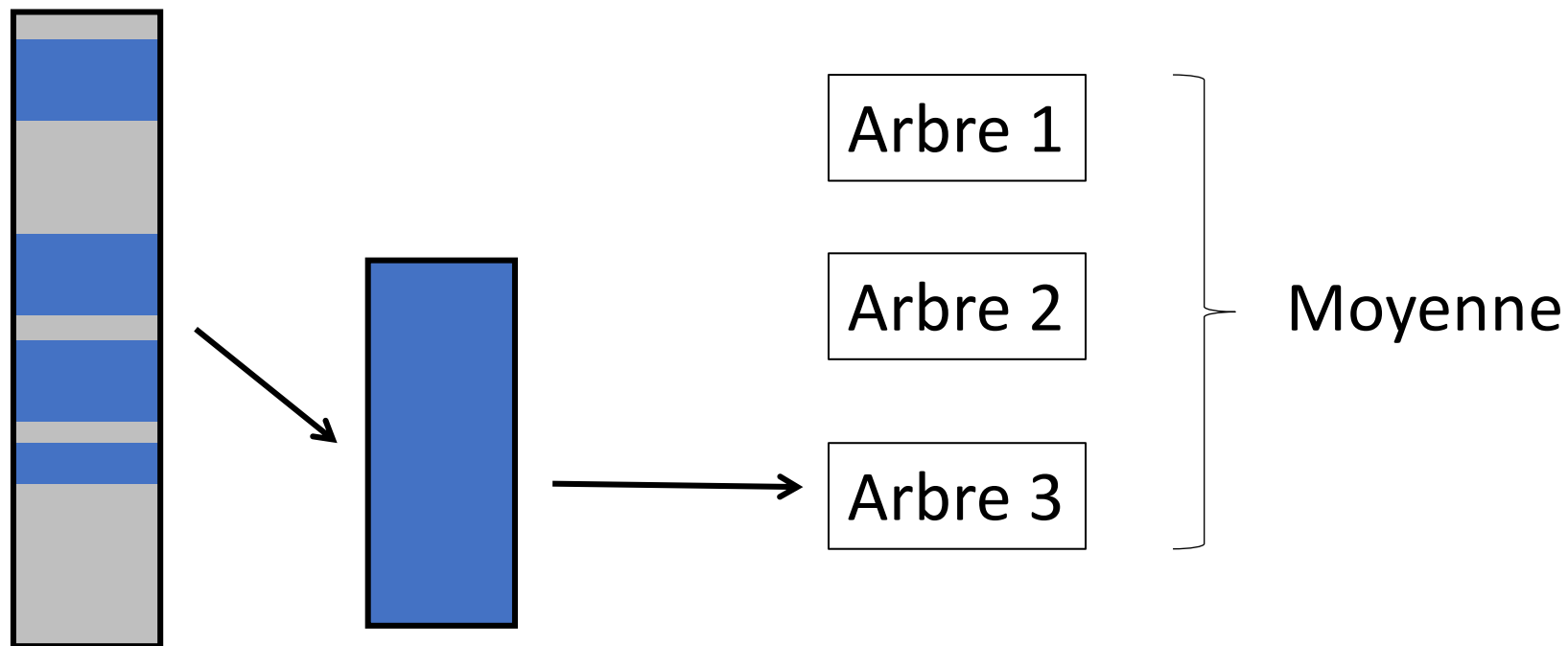
Arbre de décision et forêt aléatoire

- Les arbres décisionnels ont l'inconvénient de surapprendre très vite.
- La solution est la **forêt aléatoire**.



Arbre de décision et forêt aléatoire

- Les arbres décisionnels ont l'inconvénient de surapprendre très vite.
- La solution est la **forêt aléatoire**.

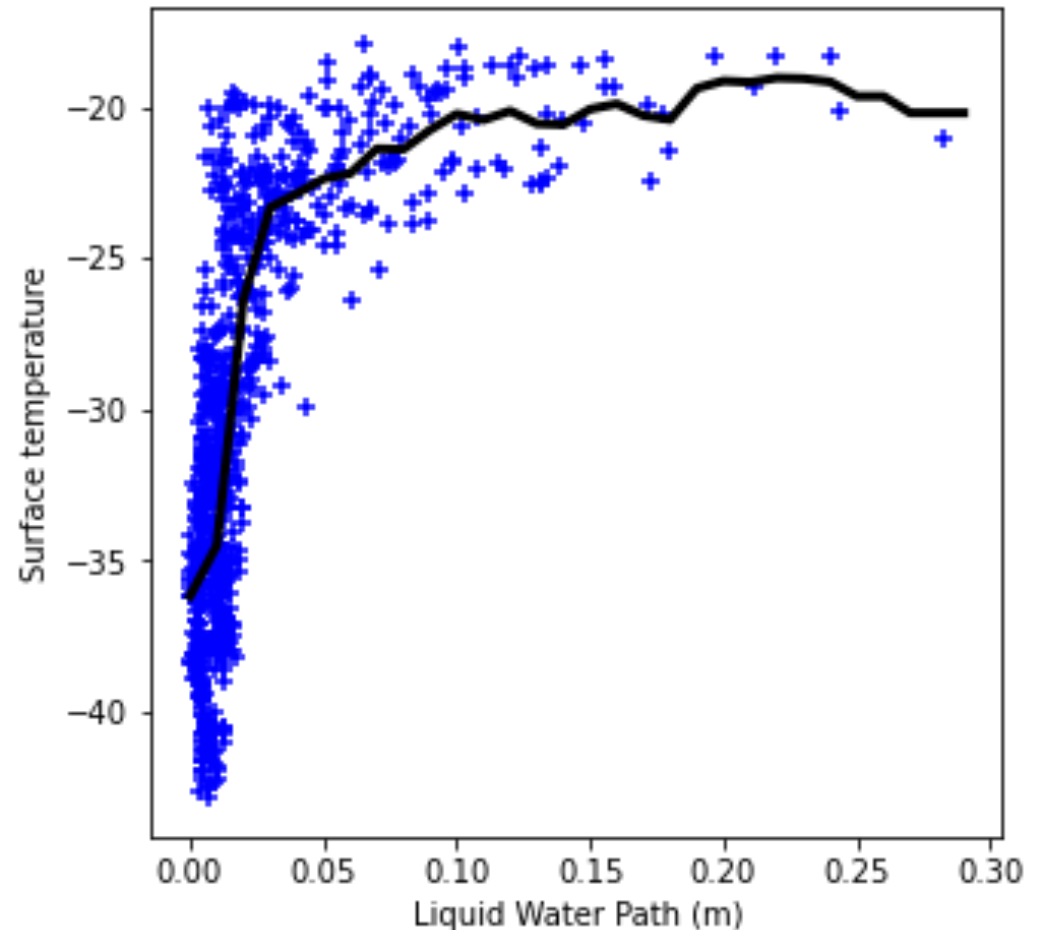


Illustration

Le résultat de la forêt aléatoire dépend de :

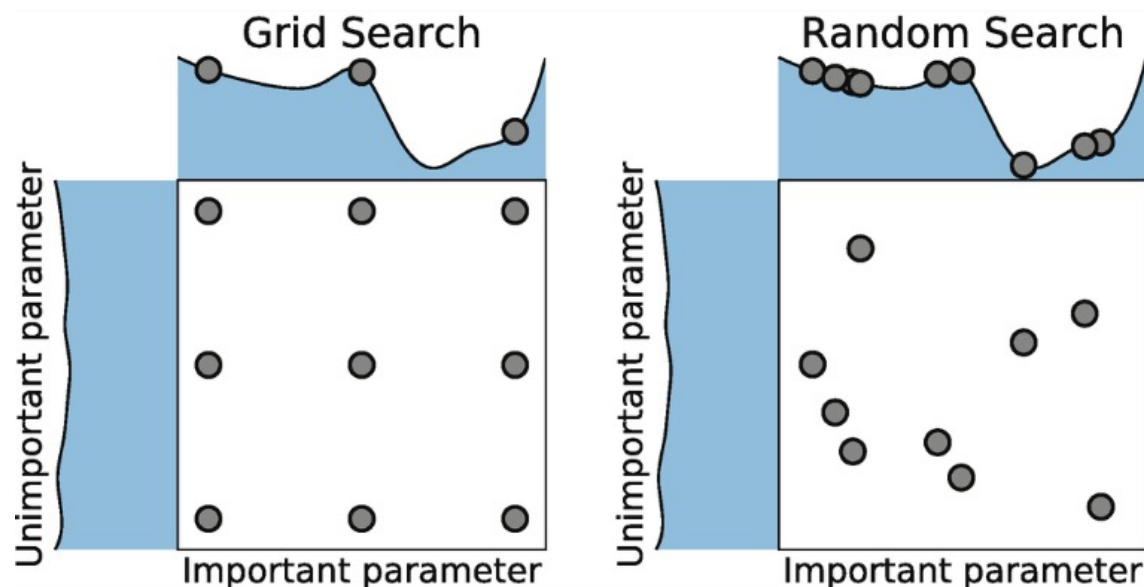
- du nombre d'arbres utilisé (`n_estimator`),
- du nombre de features utilisé pour chaque étape (`max_features`). Une valeur grande réduit l'erreur, mais augmente le risque de surapprentissage.
- du nombre de donnée minimal à utiliser pour réaliser une étape (`min_samples`). Un nombre de 2 réalise des arbres pleinement développés. Un petit nombre augmente le risque de surapprentissage.

Ce sont les hyperparamètres.



Choisir les hyperparamètres

- Il faut les choisir sur une base de donnée de validation ou en utilisant une validation croisée.
- On peut spécifier une liste d'hyperparamètre à utiliser, et garder le meilleur modèle avec **grid** ou **random Search**.



Grid Search

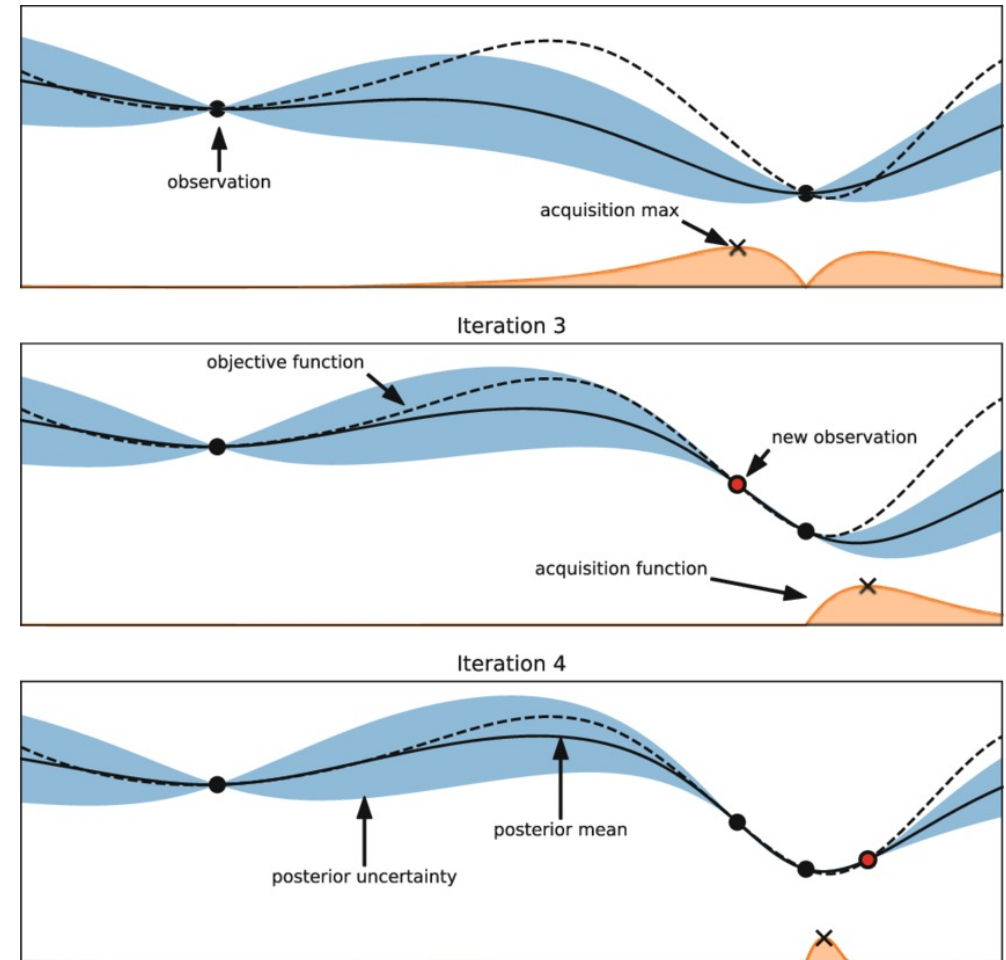
- Entraînement d'un modèle pour chaque combinaison d'hyperparamètre.
 - La meilleur combinaison évaluée avec validation croisée est retenue
-
- Recherche exhaustive, peut être mené en parallèle
 - Peu adapté pour certain hyperparamètres quantitatifs,
 - Couteux (par exemple pour un choix de 8 hyperparamètres parmi 8 : $8^8 = 16\,777\,216$ modèles à entraîner.

Random Search

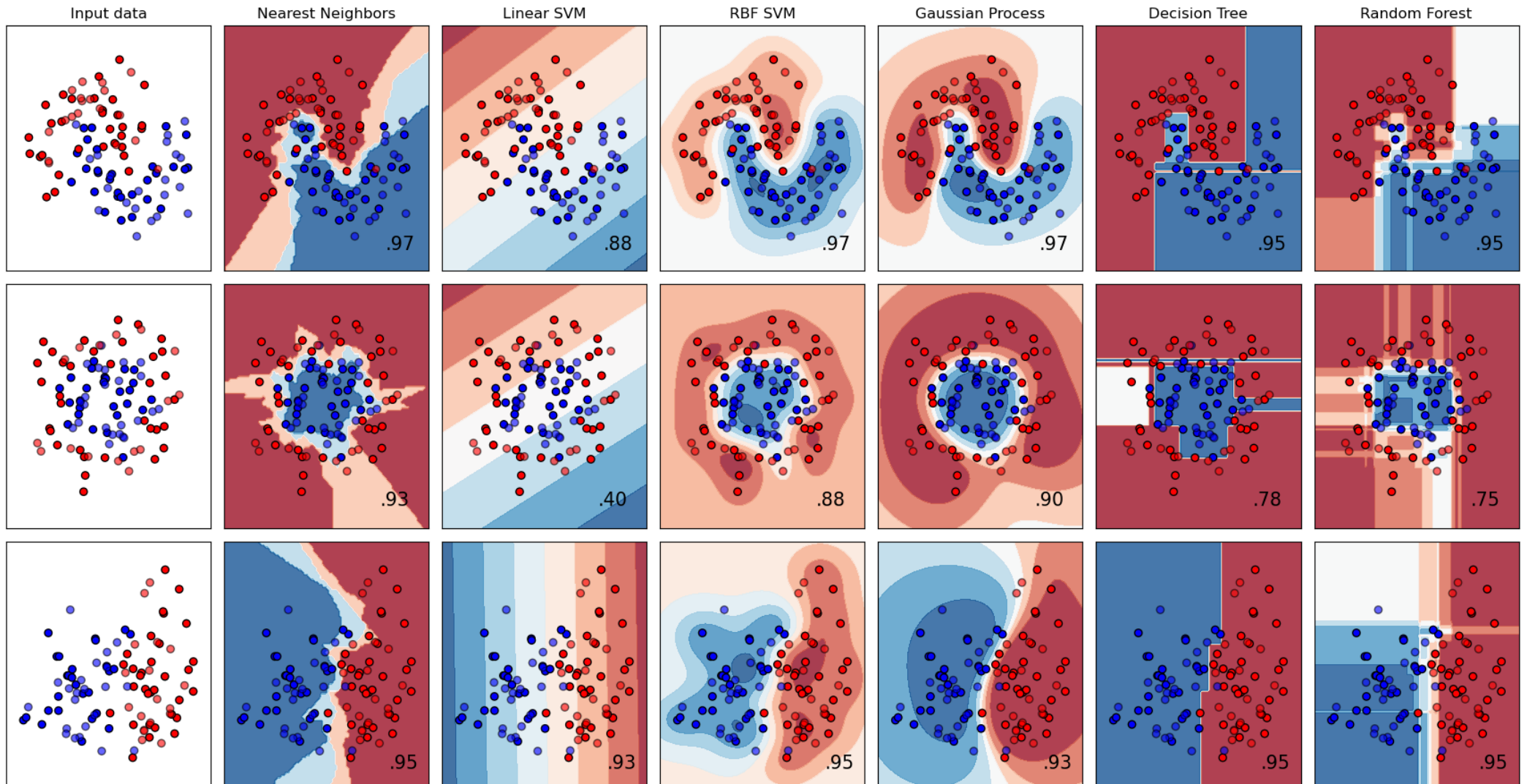
- on définit un intervalle pour chaque hyperparamètres,
 - Pour chaque hyperparamètre, on définit une liste de valeurs à tester ou une loi de probabilité pour générer des valeurs aléatoires.
 - On entraîne alors n modèle à partir d'une liste de n combinaisons d'hyperparamètre générée aléatoirement.
-
- Recherche moins exhaustive, peut être mené en parallèle
 - Moins couteux (n modèles à entrainer).

Optimisation Bayésienne

- On calcule une fonction d'acquisition, estimant une probabilité d'amélioration du modèle.
- On entraîne un modèle successivement pour les maximum de la fonction d'acquisition.



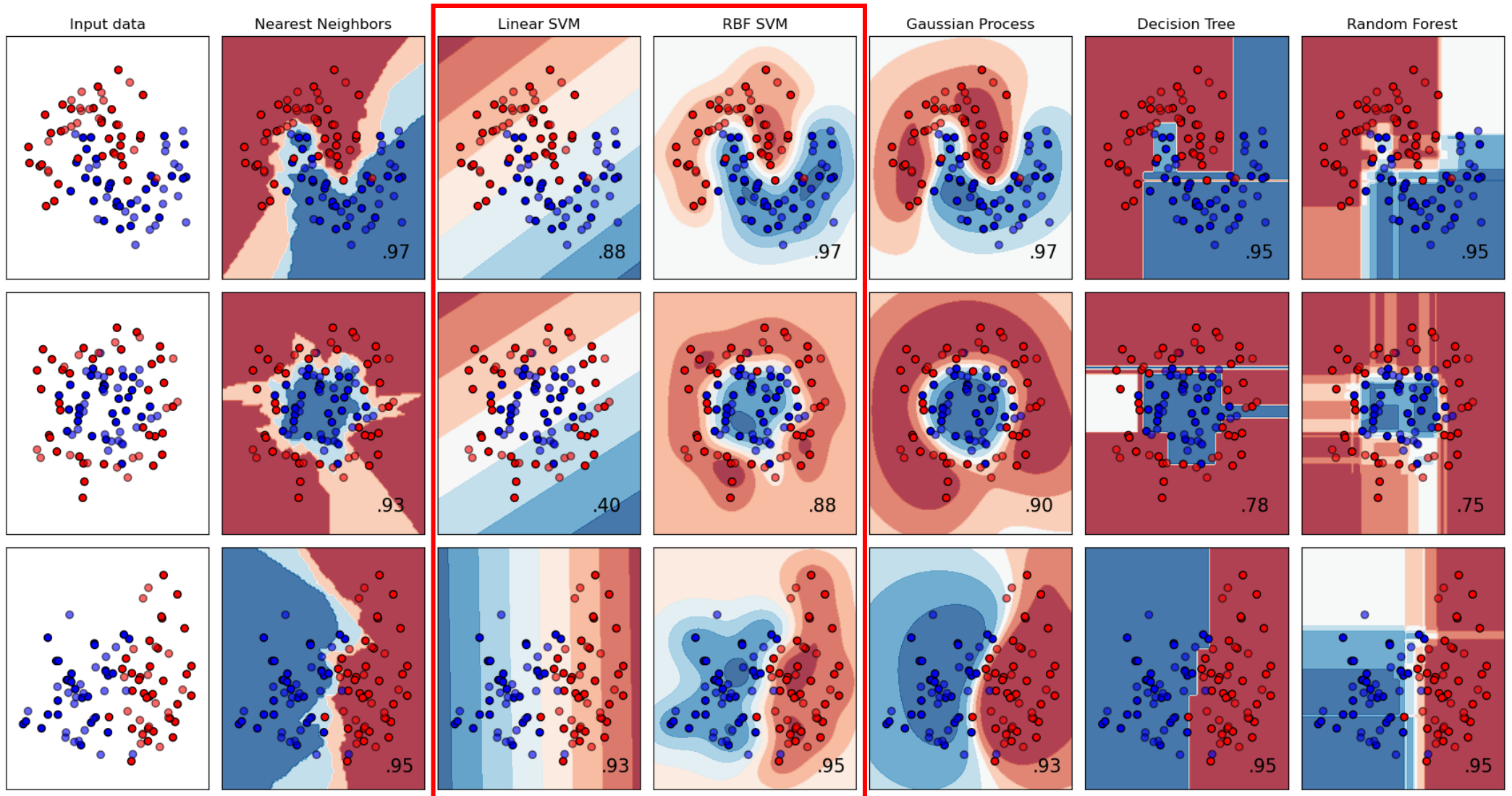
Autres algorithmes



From scikitlearn documentation (<https://scikit-learn.org>)

Autres algorithmes

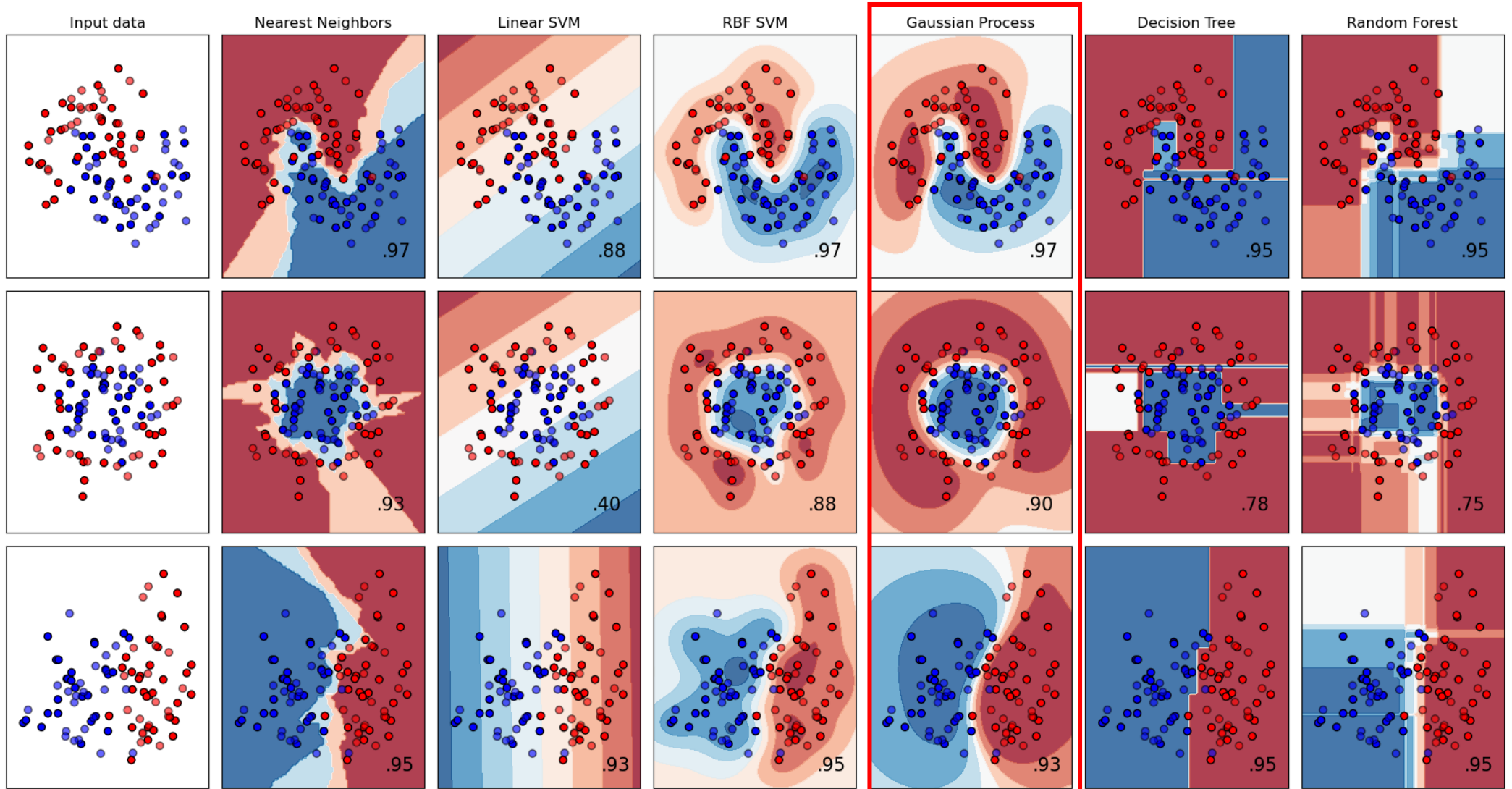
Machine à vecteur de support (support vector machine)



From scikitlearn documentation (<https://scikit-learn.org>)

Autres algorithmes

Processus gaussiens (Gaussian Processes)



From scikitlearn documentation (<https://scikit-learn.org>)

Travaux pratiques séance 2

- Deux grands types de banquise:

Banquise saisonnière (0-2m)



Banquise pluriannuelle (2 à 4m)



- Observation AMSR :
Radiomètre micro-onde aux fréquences 6.9, 10.65, 18.7, 23.8, 36.5 et 89.0GHz. Deux observations par jour à une résolution de 5 à 56 km

