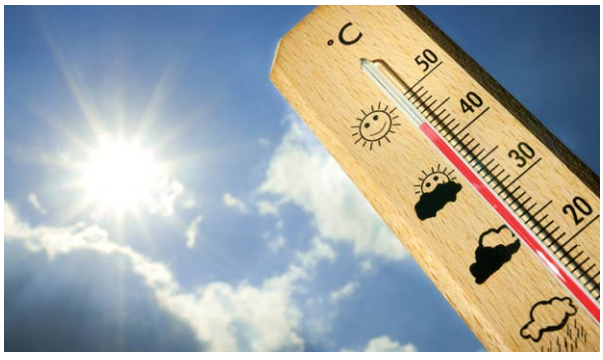
UNIVERSITÉ RENNES 2

PROJET DATA SCIENCE - MASTER 2 MAS PARCOURS SCIENCES DES DONNEES

Rapport de l'étude

Prédictions des températures de 7 stations différentes à un horizon de 36 heures

Défi Grosses Data 2018



Team : Freak-unit
Membres :

- Benjamin ALLEAU
- Abdessamad AZNAGUE
- Guillaume LE FLOCH

Organisateur : INSA de Toulouse
Encadrant : M. Romain TAVENARD

05 Octobre 2017 — 08 Janvier 2018

Plan

1 Présentation du projet **2**

2 Remerciements **3**

3 Différentes stratégies envisagées **4**

3.1 Structuration des données 4

3.2 Notions d'overfitting/underfitting 5

3.3 Validation croisée 5

3.3.1 Contexte théorique 5

3.3.2 La validation croisée Hold-Out 6

3.3.3 La validation croisée K-Fold 7

4 Analyse des corrélations entre variables et visualisation **7**

5 Gestion et remplacement des valeurs manquantes **8**

6 Choix et application d'algorithmes d'apprentissage supervisé **8**

6.1 Premiers algorithmes : la régression linéaire et ses variantes 8

6.2 Forêt aléatoire : l'algorithme Random Forest 8

6.3 Gradient Boosting : l'algorithme XGBoost 8

6.4 Deep Learning : l'utilisation des réseaux de neurones 8

7 Bilan **8**

8 Bibliographie et sources **8**

1 Présentation du projet

La compétition 2017-2018 est basée sur une stratégie d'adaptation statistique pour améliorer la prévision de températures à un horizon de 36 heures.

Les services du **CNRM** (Centre National de Recherches Météorologiques) construisent des grands modèles déterministes de l'atmosphère par résolution des équations de Navier et Stockes sur un maillage : 10km pour **ARPEGE**, kilométrique pour **AROME** mais limité à l'Europe. Il apparaît que ces modèles sont généralement biaisés, notamment parce qu'ils ne peuvent prendre en compte des phénomènes à petite échelle. Par exemple, un vent important (e.g. vent d'autan à Toulouse) provoque des turbulences qui, en mélangeant les couches d'air, entrainent une baisse de la température par rapport à celle prévue.

Un modèle statistique intégrant les prévisions du modèle déterministe peut contribuer à réduire significativement ces biais. En revanche, un modèle statistique seul est incapable de prévoir l'arrivée d'une perturbation océanique à partir de données locales. L'adaptation statistique est donc une mise en collaboration « optimale » des deux approches de modélisation, déterministe et statistique.

Le « Défi Grosses Data » est donc un challenge mettant en confrontation une multitude de groupes d'étudiants provenant des **INSA** de Toulouse et Rennes, ainsi que des Universités de **Bordeaux**, **Rennes 1**, **Rennes 2**, **Paris Descartes**, **Paul Sabatier (Toulouse 3)** ou encore la **Toulouse School of Economics**. Ce rapport détaillera le travail fourni par l'équipe **Freak-unit**, une des 8 équipes représentant l'**Université de Rennes 2**.

L'objectif de ce projet est simple : prédire au mieux les températures de 7 stations sur un horizon de 36 heures. Pour ce faire, 36 fichiers $train\_H.csv$ $(H = 1, .., 36)$, ont été mis à notre disposition. Ces fichiers contiennent les variables issues des modèles physiques décrits dans le préambule à un horizon de +H heures. Leur contenu est le suivant :

- les lignes correspondent à chaque jour sur une période allant du 1^{er} janvier 2014 au 30 mai 2016.
- les colonnes contiennent 30 variables (température, nébulosité, vent, humidité ..)

Nous allons détailler ces variables afin de bien comprendre quelles sont les mesures à notre disposition :

- **insee** : Facteur à 7 niveaux représentant le numéro insee des stations pour lesquelles nous cherchons à prédire les températures (Nice, Toulouse Blagnac, Bordeaux-Mérignac, Rennes, Lille Lesquin, Strasbourg Entzheim, Paris-Montsouris)
- **tH2_obs** : Observation de la température à 2 mètres *in situ* au point station (prédictant). **C'est la variable réponse que l'on cherche à prédire**
- **ech** : Facteur à 36 niveaux correspondant à l'échéance de validité (H)
- **capeinsSOL0** : Energie potentielle convective
- **ciwcH20** : Fraction de glace nuageuse à 20 mètres
- **clwcH20** : Fraction d'eau nuageuse à 20 mètres
- **nH20** : Fraction nuageuse à 20 mètres
- **pMER0** : Pression au niveau de la mer
- **rr1SOL0** : Précipitation horaire au niveau du sol
- **rrH20** : Précipitation horaire à 20 mètres
- **tpwHPA850** : Température potentielle au niveau 850 hPa
- **ux1H10** : Rafale 1 minute du vent à 10 mètres composante zonale
- **vapcSOL0** : Colonne de vapeur d'eau
- **vx1H10** : Rafale 1 minute du vent à 10 mètres composante verticale
- **ddH10_rose4** : Facteur à 4 niveaux indiquant la direction du vent à 10 mètres (en rose4)
- **ffH10** : Force du vent à 10 mètres en m/s
- **flir1SOL0** : Flux Infra-rouge en J/m^2
- **fllat1SOL0** : Flux de chaleur latente en J/m^2

- **flsen1SOL0** : Flux de chaleur sensible en J/m^2
- **flvis1SOL0** : Flux visible en J/m^2
- **hcoulimSOL0** : Hauteur de la couche limite en mètres
- **huH2** : Humidité 2mètres en %
- **iwcSOL0** : Réservoir neige kg/m^2 (équivalent en eau liquide des chutes de neige
- **nbSOL0_HMoy** : Nébulosité basse (moyenne sur les 6 points de grille autour de la station) (fraction en octat du ciel occulté)
- **ntSOL0_HMoy** : Nébulosité totale (moyenne sur les 6 points de grille autour de la station)
- **tH2** : Température à 2 mètres du modèle **AROME**
- **tH2_VGrad_2.100** : Gradient vertical de température entre 2 mètres et 100 mètres
- **tH2_XGrad** : Gradient zonal de température à 2 mètres
- **tH2_YGrad** : Gradient méridien de température à 2 mètres
- **mois** : Facteur à 12 niveaux représentant le mois

Il est à noter que ces 36 fichiers *train* contiennent des valeurs manquantes, le sujet sera approfondi un peu plus tard dans l'étude.

L'objectif de ce projet est le suivant : à partir des différents fichiers *train* nous devons réaliser un apprentissage (supervisé dans notre cas) afin de pouvoir appliquer par la suite notre modèle au fichier *test.csv* qui contient les mêmes variables, à l'exception de **tH2_obs** évidemment puisqu'il faut la prédire. Les lignes du fichier *test.csv* contiennent les 6 périodes de 15 jours suivantes :

- du 20/06/2016 au 02/07/2016
- du 01/08/2016 au 14/08/2016
- du 12/09/2016 au 25/09/2016
- du 24/10/2016 au 06/11/2016
- du 05/12/2016 au 18/12/2016
- du 08/05/2017 au 21/05/2017

Une fois les prédictions effectuées, il faut soumettre les résultats sous forme d'un fichier au format *csv* toujours, sur le site internet du challenge. Au bout d'exactement 1 heure après la soumission, nous pouvons visualiser notre score. La métrique utilisée pour évaluer le score est le **RMSE** (Root Mean Square Error). Si l'on note $\hat{y}$ notre vecteur de prédictions et y_{obs} les valeurs des températures réellement observées, le **RMSE** associé se note :

$$RMSE_{\hat{y}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_{obs_i})^2}$$

En d'autres termes, cette métrique permet de mesurer d'une certaine façon l'écart entre nos prédictions et la réalité. Il est à noter que le challenge fournit un score de 1.30755 appelé **Baseline** qui correspond à un score de référence, associé à « un modèle élémentaire de prévision sans effort particulier d'optimisation. C'est un objectif a minima à améliorer dans ce concours ». Le décor est planté, désormais il ne reste plus qu'à se lancer dans le challenge.

2 Remerciements

Avant de détailler les différentes étapes de ce projet, les 3 membres de la **Freak-unit** team souhaitent remercier tout particulièrement leur professeur encadrant, **M. Romain TAVENARD** pour son accompagnement tout au long du projet et pour l'aide apportée. Sa connaissance du *Deep Learning* aura notamment été un facteur déterminant dans la réussite de ce projet.

3 Différentes stratégies envisagées

Afin d'effectuer l'analyse la plus fine et pertinente possible, il est primordial de bien définir la stratégie d'étude à adopter. Par « stratégie », nous parlons ici de la façon dont nous allons structurer les données, des techniques que nous allons employer pour optimiser la performance des algorithmes. Cela peut être vu comme un problème d'optimisation sous contrainte : notre objectif est de réaliser la meilleure performance sous des contraintes de temps et de matériel. Il faut donc réussir à combiner au mieux la théorie mathématique avec la réalisation informatique.

3.1 Structuration des données

Un premier travail a consisté à agréger les données des 36 fichiers $train\_H.csv$ $(H = 1, .., 36)$ dans un seul jeu de données : cela aboutit à une matrice de 189281 lignes et 31 colonnes. Dans un deuxième temps, nous avons divisé ce fichier en 7 sous-fichiers : un fichier regroupant les prédictions aux 36 échéances de chaque station. Ce choix a été guidé par 3 éléments :

- L'intuition : sachant qu'il existe plusieurs types de climats (océaniques, continental, méditerranéen) et qu'on les retrouve à travers les stations présentes dans les données, il ne paraît pas anormal d'essayer de prédire chaque station indépendamment des autres. Ne perdons pas de vue que les chiffres sont là pour quantifier la réalité, l'objectif est donc de les utiliser pour représenter au mieux les phénomènes réels.
- Un indice lâché par les organisateurs du concours qui ont laissé entendre qu'il pouvait exister un biais lié à chaque station.
- L'analyse du comportement des indicateurs par station.

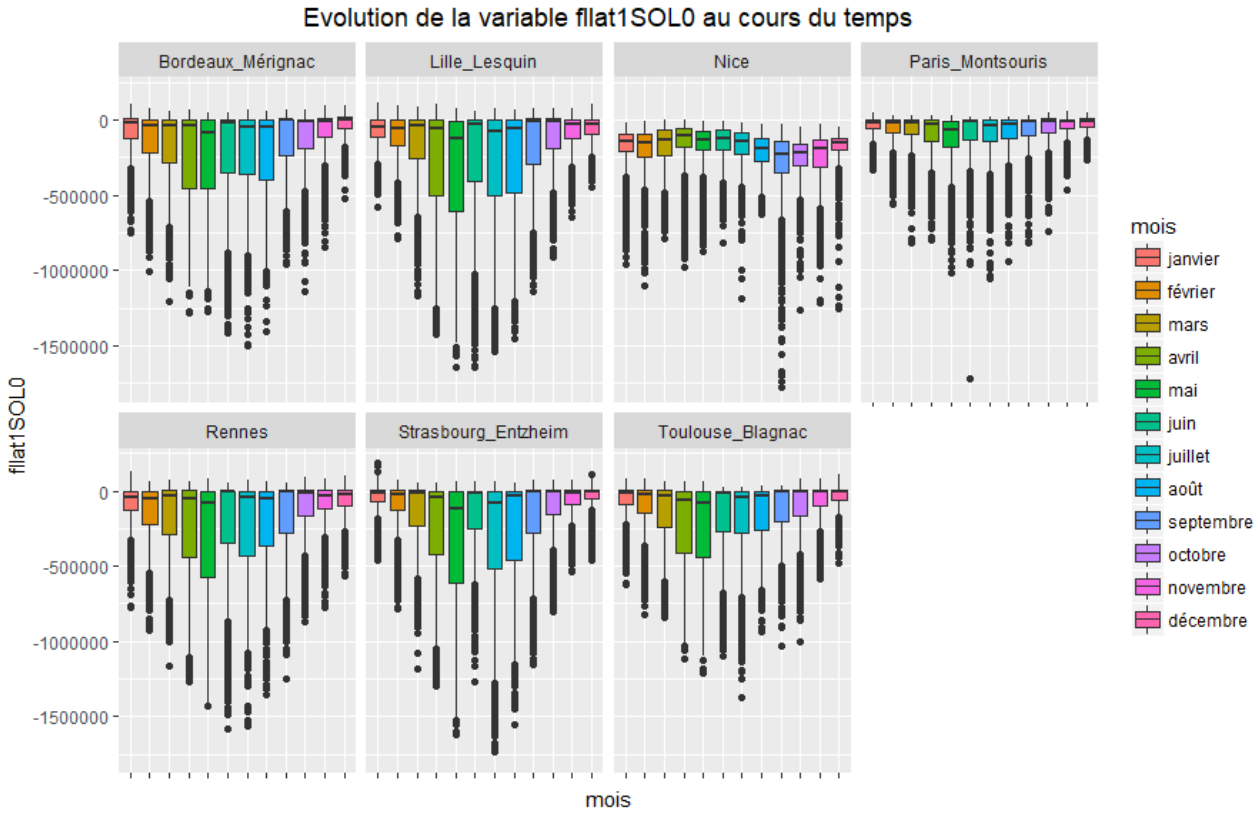


FIGURE 1 – Exemple du flux de chaleur latente dont le comportement diffère d'une station à une autre

Dans la suite, nous verrons que le gros jeu de données (189281 lignes) ou les fichiers par stations seront utilisés en fonction de l'algorithme choisi. Une autre option aurait pu être de découper les données par station et par échéance. Cependant en procédant de la sorte, on se retrouverait avec 252 modèles à estimer, et l'on s'exposerait également au risque de sur-apprentissage (*overfitting* en anglais). La dimension des données va également impacter le choix de la méthode d'estimation des paramètres et d'estimation de la performance de l'algorithme : la **validation croisée**. Nous allons donc effectuer quelques rappels sur ces différents concepts.

3.2 Notions d'overfitting/underfitting

Comme introduit brièvement un peu plus haut dans ce rapport, l'un des enjeux de ce projet dans le but d'obtenir les meilleurs prédictions possibles, selon le critère du **RMSE**, sera d'éviter de se retrouver en situation sur-apprentissage ou de sous-apprentissage (respectivement *overfitting* et *underfitting*). En ce sens, on va chercher à construire un estimateur (une fonction) optimal, robuste qui conservera un pouvoir de prédiction de qualité sur un nouvel échantillon. Prenons le dessin suivant qui sera un peu plus parlant.

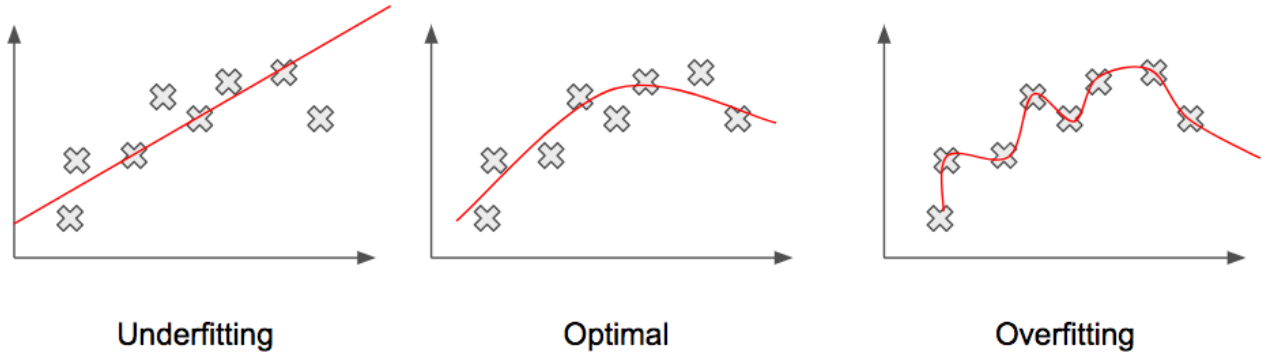


FIGURE 2 – Illustration de ces notions à partir d'un exemple simple dans $\mathbb{R}^2$

On s'aperçoit que sur le graphe de gauche, l'estimateur est trop rigide, il ne « colle » pas assez aux données. Sur le graphe de droite c'est le contraire, l'estimateur est beaucoup trop sensible aux valeurs, et lorsque les données changeront, les prédictions ne seront donc plus aussi précises. Nous souhaitons donc trouver une fonction ayant une allure similaire à celle du graphe du milieu, qui sera plus « robuste » que les deux autres.

Nous verrons dans la suite que pour remédier au problème de l'overfitting, nous pouvons utiliser des formes de **régularisation**, encore appelées pénalités et qui se traduiront par des contraintes dans le problème d'optimisation d'une certaine **fonction de perte** que l'on va choisir.

Une façon de vérifier que le modèle est robuste peut-être de procéder à de la **validation croisée**, c'est donc le prochain point que nous allons développer.

3.3 Validation croisée

Comme énoncé précédemment, nous allons utiliser la méthode de la validation croisée à la fois pour contrôler la performance et la robustesse de l'algorithme mis en œuvre, mais également pour calibrer des hyper-paramètres dans le cas de modèles complexes. Nous allons donc dans la suite effectuer quelques rappels théoriques afin de bien comprendre pourquoi nous utilisons cette méthode ainsi qu'une présentation des deux types de validation croisée que nous avons choisi d'utiliser.

3.3.1 Contexte théorique

Soit d_1^n l'observation d'un n-échantillon $D_1^n = \{(X_1, Y_1), ..., (X_n, Y_n)\}$ d'une loi conjointe P sur $\mathcal{X}$ x $\mathcal{Y}$ qui est inconnue. Le premier travail consiste à estimer cette loi P à l'aide des données disponibles, à savoir d_1^n.

En théorie, il faudrait minimiser la quantité appelée **risque** (ou encore **erreur de généralisation**) d'une règle de prédiction que l'on va noter f. Ainsi de façon mathématique, on définit le **risque** de la manière suivante :

$\mathcal{R}_P(f) = \mathbb{E}_{(X,Y)}[l(Y, f(X))]$ avec l que l'on appelle **fonction de perte** et qui a pour expression :

- $l(y, y') = |y - y'|^q$ en régression réelle (on prendra souvent la perte *quadratique*, c'est-à-dire lorsque $q = 2$)
- $l(y, y') = \mathbb{1}_{y \neq y'} = \frac{|y-y'|}{2} = \frac{|y-y'|^2}{4}$ en discrimination binaire.

Cependant en pratique, il nous est impossible de calculer le **risque**. Il faut donc l'estimer et chercher à minimiser cette quantité estimée par la suite. Une première approche naturelle serait d'utiliser le risque empirique, c'est-à-dire d'estimer l'espérance de la fonction de perte par la moyenne empirique de cette même fonction. Le **risque empirique** d'un algorithme de prédiction $f_{D_1^n}$ construit sur D_1^n se note de la façon suivante :

$$\widehat{\mathcal{R}}_n(f_{D_1^n}) = \frac{1}{n}\sum_{i=1}^{n} l(Y_i, f_{D_1^n}(X_i))$$

Le problème de cet estimateur est l'absence d'indépendance : en effet, pour effectuer nos prédictions on se sert des mêmes données qui ont servi à faire l'algorithme d'apprentissage. En ce sens, notre algorithme risque de trop « coller » aux données d'apprentissage, puisqu'il aura acquis l'information à partir de ces données il sera capable de bien prédire la variable réponse, mais risque de ne pas être aussi performant à l'avenir sur un nouveau jeu de données. C'est ce qu'on nous avons défini comme étant l'*overfitting*. C'est donc pour corriger ce problème que nous allons avoir recours à une autre méthode d'estimation du **risque**, qui est meilleure : la **validation croisée**.

3.3.2 La validation croisée Hold-Out

Le concept de base pour la validation croisée **Hold-Out** consiste à séparer nos données d_1^n en deux sous-échantillons :

- $\mathcal{A}$: l'échantillon d'apprentissage
- $\mathcal{V}$: l'échantillon de validation

Cela implique d'avoir à la base un jeu de données suffisamment « grand » pour pouvoir appliquer une telle méthode, ce qui n'est pas un problème dans notre cas. Comment séparer les données ? En pratique on va diviser notre échantillon en deux parties égales, si l'on dispose d'un échantillon de taille n, $\mathcal{A}$ et $\mathcal{V}$ seront donc de taille $\frac{n}{2}$. Si les données sont classées, on procèdera à un tirage aléatoire sans remise de $\frac{n}{2}$ éléments dans $\{1, ..., n\}$. Ensuite on « nourrit » notre algorithme à partir des données contenues dans $\mathcal{A}$ et on teste sa performance sur les données de $\mathcal{V}$: ainsi le problème d'indépendance est réglé. Illustrons cette méthode avec un schéma.

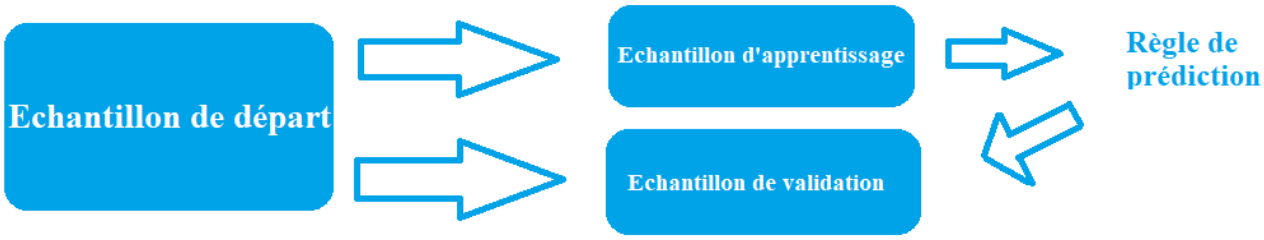


FIGURE 3 – Fonctionnement de la validation croisée Hold-Out

De façon mathématique, cet estimateur du risque se définit de la façon suivante :

$\widehat{\mathcal{R}}(f_{D_1^n}) = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} l(y_i, f_{d_\mathcal{A}}(x_i))$ avec $|\mathcal{V}|$ le cardinal de $\mathcal{V}$ (autrement dit, le nombre de lignes de l'échantillon de validation).

Dans le cadre de nos travaux, l'utilisation de cette méthode va nous permettre de gagner du temps si l'on utilise la matrice de travail contenant toutes les stations. La méthode suivante que nous allons présenter prendra plus de temps mais peut apporter plus de stabilité dans les résultats.

3.3.3 La validation croisée K-Fold

La validation croisée **K-Fold** est une variante de la méthode vue précédemment, qui on le rappelle était adaptée aux jeux de données de taille suffisamment grande. Cependant sur des jeux de données plus petits, comme c'est le cas de nos fichiers par station qui font environ 27000 lignes chacun,ce type de validation croisée peut également s'avérer efficace. Le fonctionnement de cette méthode itérative est le suivant :

- On procède à une partition de notre échantillon en K blocs équilibrés : $B_1, ..., B_K$ auxquels sont associés des tailles $n_1, ..., n_K$. (En pratique on prend $K = 10$ quand la taille des données le permet)
- On note $d_{B_i} = \{(x_j, y_j), j \in B_i\}$ le i^{eme} bloc de l'échantillon d_1^n. Pour chaque itération $i = 1, ..., K$ on laisse de côté les données de B_i pour faire notre apprentissage sur les K-1 blocs restants.
- On teste la performance de notre règle de prédiction sur le bloc B_i pour obtenir une estimation du risque à l'itération i que l'on note :
  $\widehat{\mathcal{R}}_i = \frac{1}{n_i} \sum_{j \in B_i} l(y_j, f_{d_1^n \backslash d_{B_i}}(x_j))$
- Enfin, après avoir procédé à toutes les itérations, on calcule la véritable estimation du risque en faisant la moyenne des risques calculés à chaque itération :
  $\widehat{\mathcal{R}}(f_{D_1^n}) = \frac{1}{K} \sum_{i=1}^{K} \widehat{\mathcal{R}}_i$

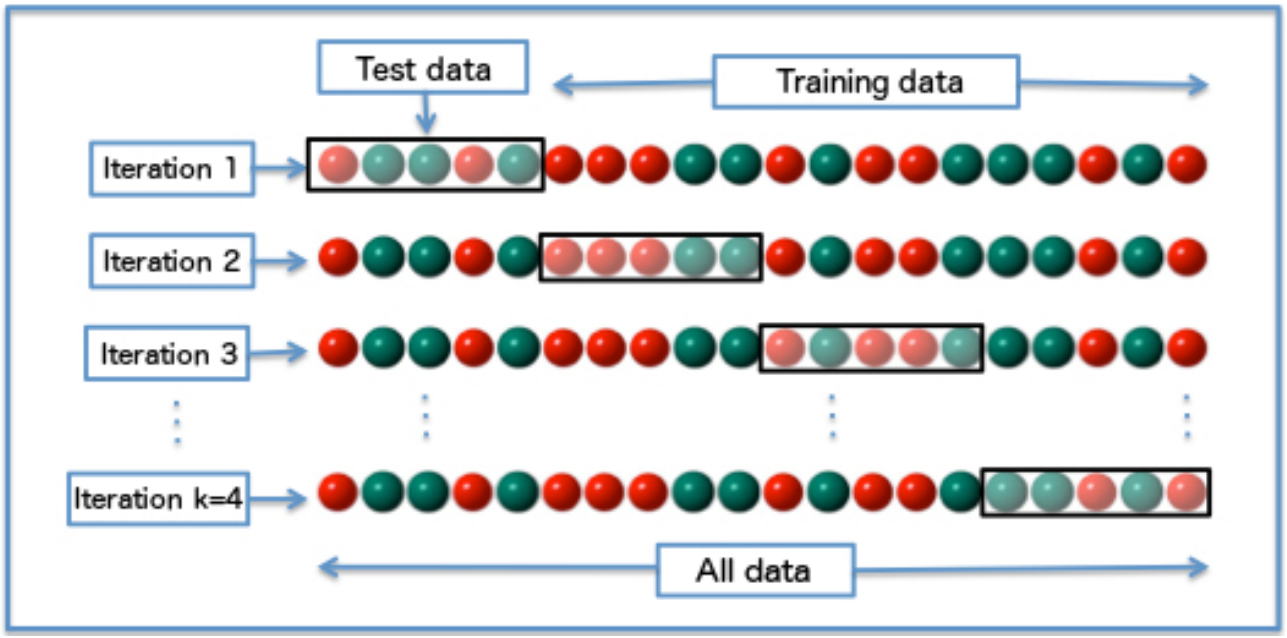


FIGURE 4 – Exemple de fonctionnement pour K=4

Il s'agit désormais d'utiliser la méthode de notre choix dans le but de trouver le meilleur compromis entre précision de l'algorithme et temps de calcul. Le domaine de la *Data Science* fait interagir la théorie mathématique avec la réalisation informatique, il est donc important de prendre en compte ces deux aspects et d'optimiser leur utilisation conjointe au mieux.

4 Analyse des corrélations entre variables et visualisation

5 Gestion et remplacement des valeurs manquantes

Une partie clé de cette étude aura été la gestion des valeurs manquantes. En effet, tous les fichiers $train\_H.csv$ $(H = 1, .., 36)$, en contiennent, ce qui nous a donc amené à nous poser les questions suivantes : faut-il éliminer les lignes contenant des *NA* ? Faut-il remplacer ces valeurs manquantes ? Si oui, pour quelles variables, de quelle façon et à quel coût ? Nous allons détailler l'analyse qui nous a permis de répondre à ces différentes interrogations et de prendre nos décisions.

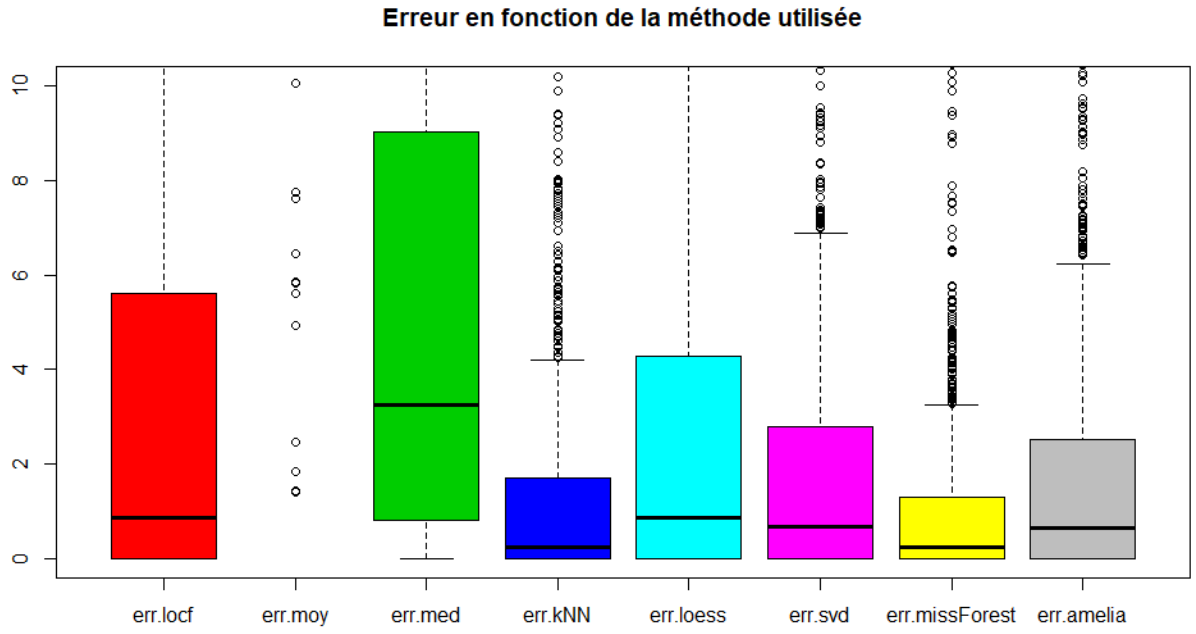


FIGURE 5 – Taux d'erreur des méthodes d'imputation testées

6 Choix et application d'algorithmes d'apprentissage supervisé

6.1 Premiers algorithmes : la régression linéaire et ses variantes

6.2 Forêt aléatoire : l'algorithme Random Forest

6.3 Gradient Boosting : l'algorithme XGBoost

6.4 Deep Learning : l'utilisation des réseaux de neurones

7 Bilan

8 Bibliographie et sources

- **WikiStat**, *Scénario: Imputation de données manquantes*