

# Rapport du projet CNAM-UASB03

Rémy LETIENT <[rletient@gmail.com](mailto:rletient@gmail.com)> & Fabrice DUNAN  
<[fabrice.dunan@laposte.net](mailto:fabrice.dunan@laposte.net)>

# Table des matières

1. Introduction au projet .....	2
1.1. Objectif "métier" .....	2
1.2. Objectifs techniques imposés .....	2
1.2.1. Visualisation .....	2
1.2.2. Analyse de données .....	2
1.2.3. Système de gestion de bases de données "scalable" .....	3
1.2.4. Architecture et "streaming" Spark .....	3
2. Présentation des "données brutes" .....	4
2.1. Les données météo .....	4
2.1.1. OMM .....	4
2.1.2. Récupération des données météo sur le site de MeteoFrance .....	4
2.1.3. Tableau synthétique des données météo "SYNOP" .....	4
2.1.4. Des détails sur les données SYNOP .....	5
Quelques mots sur les variations régionales des données SYNOP .....	6
De "numer_sta" à une commune hébergeant une station SYNOP .....	6
2.2. Les données "Arrêtés de catastrophes naturelles" .....	6
2.2.1. Tableau synthétique des données catastrophes naturelles .....	7
2.2.2. Dimensions .....	7
3. Validation du jeu de données pour la résolution du problème de données : Visualisation de l'agrégation données / catastrophes .....	8
4. Constitution et analyse de la "donnée utile" .....	13
4.1. Les données catastrophes .....	13
4.2. Les données météo .....	14
4.3. Constitution du jeu de données "utile" .....	15
4.3.1. Pré traitement des données météo .....	15
Le filtrage des variables par choix .....	15
Le filtrage des individus mesures par choix .....	17
La gestion des valeurs manquantes .....	17
Analyse unidimensionnelle .....	18
4.3.2. Analyses bidimensionnelles .....	18
4.3.3. Pré-traitement des données catastrophes .....	19
4.3.4. Les données agrégées : météo et catastrophes .....	19
Choix d'un stratégie et conséquence sur la temporalité du phénomène .....	19
Agrégation .....	20
Filtrage .....	20
Création de la variable à prédire .....	20
Distribution de la variable à prédire dans le jeu d'apprentissage .....	20
Rééchantillonnage .....	21

5. Les modèles . . . . .	22
5.1. Méthode . . . . .	22
5.2. Résultats . . . . .	22
5.2.1. Les évaluations des modèles . . . . .	22
5.2.2. Les hyperparamètres du meilleur algorithme . . . . .	24
6. Architecture de la solution . . . . .	26
6.1. InfluxDB . . . . .	26
6.1.1. Mise en oeuvre : InfluxDB, Grafana et Telegraf . . . . .	27
6.1.2. Injection des données . . . . .	27
6.1.3. Passage à l'échelle . . . . .	29
la notion de <i>shard</i> InfluxDB . . . . .	29
6.2. Spark Streaming . . . . .	30
6.2.1. Création du modèle de prédiction . . . . .	30
6.2.2. L'application <i>influxdb_streaming</i> . . . . .	31
A la rechercher des données . . . . .	31
Le streaming . . . . .	31
6.2.3. Mise en oeuvre . . . . .	32
6.3. Passage à l'échelle . . . . .	32
7. CONCLUSION . . . . .	34
8. Pour aller plus loin ... . . . . .	35
9. ANNEXES . . . . .	36
9.1. Quelques règles de conversion . . . . .	36
9.2. Description des variables des données météo . . . . .	36
9.3. SYNOP . . . . .	38
9.4. Correspondances numer_sta et communes des stations de mesure . . . . .	39
9.5. Script de récupération des données Météofrance . . . . .	42
9.6. Liste des stations SYNOP de France métropolitaine concernées par des catastrophes naturelles . . . . .	43
9.7. Types des catastrophes et effectifs . . . . .	43
9.8. Types des catastrophes, effectifs et étude de la durée . . . . .	45
9.9. "Notebooks" pour l'analyse de donnée . . . . .	49
9.10. Pourcentage de valeurs manquantes dans les données météo . . . . .	50
9.11. InfluxDB . . . . .	52
9.11.1. base de tests . . . . .	52
9.12. Telegraf . . . . .	54
Références . . . . .	55



<https://gitlab.com/logrus-fr/CNAM-UASB03>

Rémy LETIENT <[rletient@gmail.com](mailto:rletient@gmail.com)> & Fabrice DUNAN <[fabrice.dunan@laposte.net](mailto:fabrice.dunan@laposte.net)>

v1, 2018-09-29

tuteurs:

- N.DIENG <[n-deye.niang\\_keita@cnam.fr](mailto:n-deye.niang_keita@cnam.fr)>
  - M.CRUCIANU<[michel.crucianu@cnam.fr](mailto:michel.crucianu@cnam.fr)>
  - P.RIGAUX<[philippe.rigaux@cnam.fr](mailto:philippe.rigaux@cnam.fr)>

relecture:

- V.PEIGNEY <[virginie.peigney@laposte.net](mailto:virginie.peigney@laposte.net)>

# 1. Introduction au projet

Ce rapport de projet est élaboré dans le cadre de la [formation d'analyste de données massives du CNAM](#).

Les attendus de l'unité d'enseignement **UASB03**, objet de ce document, sont décrits en détail [ici](#).

## 1.1. Objectif "métier"

L'objectif de ce projet est de fournir un système permettant de lever des alertes en cas de prévision d'une catastrophe naturelle sur une commune de France métropolitaine.

Le projet sera réalisé à partir des données relatives à la météorologie de Météofrance et aux arrêtés de catastrophes naturelles disponibles sur le site des données ouvertes du gouvernement français.

Dans un cas réel, le système serait alimenté en continu par les données de l'ensemble des stations de mesure météo en France métropolitaine. Les mesures sont prises périodiquement et émises à un rythme non précisé mais supposé bref. L'objectif serait de les stocker et de les traiter pour prédire la catastrophe au plus vite.

Le passage du problème de données de l'échelle nationale à l'échelle mondiale sera également étudié.

## 1.2. Objectifs techniques imposés

### 1.2.1. Visualisation

Ce rapport présentera une [première partie](#) permettant de répondre à la question : "Les deux jeux de données 'source' nous permettent-ils de résoudre le problème de données ?"

Cette validation sera effectuée par des outils de visualisation **javascript et son framework Angular** dont le code est joint à l'archive contenant ce rapport. Le code est également présent dans un des dépôts du projet.

Cette dernière remarque vaut pour l'ensemble des livrables code du projet. A chaque fois, l'URL de la ressource sera fournie.

**Spark** a également été utilisé pour le prétraitement des données d'arrêté de catastrophe naturelle.

### 1.2.2. Analyse de données

La présentation des données a lieu dans un [chapitre introductif](#) à l'aide d'outils bureautiques standards.

Les "Analyses exploratoires, prétraitement, études préalables (normalisation, nettoyage des données, gestion de données manquantes, agrégation...)" seront effectuées en **python et ses frameworks d'analyse de données**. Les détails sont fournis dans le premier des deux notebooks joints au rapport et disponibles également dans [un des dépôts du projet](#).

On synthétisera et justifiera la méthode dans ce [chapitre](#).

On trouvera, dans le [deuxième notebook](#), la comparaison des performances de plusieurs algorithmes, la sélection du meilleur modèle prévisionnel généré et l'étude des meilleurs paramètres de ce candidat.

Un [chapitre](#) sera dédié à l'exposition des résultats et leur interprétation.

### **1.2.3. Système de gestion de bases de données "scalable"**

InfluxDB sera choisi comme système de stockage passant à l'échelle par distribution. On étudiera et expérimentera l'alimentation, la "scrutation" et l'extraction des données dans ce SGBD.

Une [partie de chapitre](#) éclairera ce sujet.

### **1.2.4. Architecture et "streaming" Spark**

L'[architecture de traitement](#) pour permettre le passage à l'échelle se fondera sur la solution **Spark Streaming**.

Elle a fait l'objet du portage de la méthode sélectionnée lors de la phase d'analyse des modèles. Une plateforme de simulation d'acquisition des données a été bâtie autour et sera détaillée dans la partie [streaming](#).

## 2. Présentation des "données brutes"

Pour réaliser notre modèle, on dispose de deux jeux de données issus du site de [Météofrance](#) et de la [plateforme ouverte des données publiques françaises](#).

### 2.1. Les données météo

(NdR : ce paragraphe est une amélioration de son équivalent dans l'étude InfluxDB NFE204)

Le site de Météofrance fournit des données publiques, gratuites et librement utilisables pour peu, d'après la licence d'utilisation que l'on mentionne sa [source et paternité](#) ... ce qui est chose faite conformément aux instructions du lien précédent dans la rubrique "Condition d'accès".

Ces données contiennent des informations intelligibles au profane de la météorologie: température, pression, lieu de mesure, vitesse du vent, enneigement...mais aussi plus absconses (par exemple le géopotentiel !)

*Descriptif des données choisies issu du site de meteoFrance*

Données d'observations issues des messages internationaux d'observation en surface (SYNOP) circulant sur le système mondial de télécommunication (SMT) de l'Organisation Météorologique Mondiale (OMM). Paramètres atmosphériques mesurés (température, humidité, direction et force du vent, pression atmosphérique, hauteur de précipitations) ou observés (temps sensible, description des nuages, visibilité) depuis la surface terrestre. Selon instrumentation et spécificités locales, d'autres paramètres peuvent être disponibles (hauteur de neige, état du sol, etc.)

#### 2.1.1. OMM

L'**OMM** est une [émancipation des Nations Unies](#) impliquée dans les problématiques écologiques actuelles et ses moyens techniques de partage de mesures météorologiques.

Le système d'information (**SIO**) et la politique d'échange de mesures météo sont brièvement décrits [ici](#).

#### 2.1.2. Récupération des données météo sur le site de MeteoFrance

Les données sont disponibles sur le site internet de MétéoFrance via un formulaire.

Après une rapide analyse du fonctionnement de ce formulaire, on développe un script pour récupérer l'ensemble des données en une passe pour constituer un jeu conséquent.

Ce code du script [de récupération des données MeteoFrance](#) est fourni en annexe.

#### 2.1.3. Tableau synthétique des données météo "SYNOP"

<b>Description</b>	<b>Données météo collectées par station météorologiques au format international SYNOP</b>
Fréquence des mesures (individus)	<p>Une mesure :</p> <ul style="list-style-type: none"> <li>• par station (61 stations en France métropolitaine et outre-mer)</li> <li>• toutes les 3 heures</li> </ul>
Période et durée	<ul style="list-style-type: none"> <li>• Janvier 1996 à décembre 2017</li> <li>• donc <b>22 ans</b> de données</li> </ul>
Nombre de dimensions	<b>41</b>
Nombre de lignes	près de <b>4M de lignes</b> (1 mois = 14000 lignes en moyenne)
Nombre de fichiers	264 (un par mois)
Format	CSV, signification des champs fournie et normalisée
Source	<a href="#">Lien</a>

#### 2.1.4. Des détails sur les données SYNOP

Quelques caractéristiques des données SYNOP mises à disposition [dans les diverses rubriques de la page leur étant dédiée](#).

Ci-après, un échantillon de lignes et colonnes :

<b>numer_sta</b>	<b>date</b>	<b>pmer</b>	<b>tend</b>	<b>cod_tend</b>	<b>dd</b>	<b>ff</b>	<b>t</b>	<b>td</b>	<b>u</b>
7005	2017050 1000000	100160	60	0	150	2.700000 00	282.4500 00	280.8500 00	90
7015	2017050 1000000	100270	190	2	200	4.600000 00	283.4500 00	281.8500 00	90
61996	2017050 1030000	100890	-80	5	300	12.30000 0	290.3500 00	286.0500 00	76
61997	2017050 1030000	101850	-120	8	240	15.40000 0	276.6500 00	275.1500 00	90
7181	2017050 1060000	101190	210	3	190	5.200000 00	279.6500 00	278.4500 00	92

<b>numer_sta</b>	<b>date</b>	<b>pmer</b>	<b>tend</b>	<b>cod_tend</b>	<b>dd</b>	<b>ff</b>	<b>t</b>	<b>td</b>	<b>u</b>
7190	2017050 1060000	101210	390	3	270	3.000000	280.9500 00	277.8500 00	81
7335	2017050 6120000	101210	160	3	240	9.600000	287.8500 00	282.8500 00	72
7434	2017050 6120000	101350	190	1	250	6.100000	285.7500 00	281.4500 00	75

En [annexe des variables météo SYNOP](#), le tableau décrivant les mnémoniques des colonnes du fichier CSV, leur types et unités.

On verra plus tard dans ce document, dans ce [chapitre](#), une carte mentale décrivant plus visuellement le sous ensemble des variables sélectionnées pour ce projet.

### Quelques mots sur les variations régionales des données SYNOP

Les données SYNOP suivent une norme internationale. La norme prévoit des variations selon les continents voire les pays. La France ne déroge pas à ce point. Ce sujet, qui permet de comprendre les variables fournies par MeteoFrance, est secondaire dans cette étude. Il est traité en [annexe](#).

### De "numer\_sta" à une commune hébergeant une station SYNOP

On trouvera en [annexe](#) ou dans [la rubrique information sur les stations du site dédié aux données](#) les diverses correspondances entre codes indicatif d'une station (*numer\_sta*) d'une part et noms de villes, paramètres géodésiques et altitude d'autre part.

## 2.2. Les données "Arrêtés de catastrophes naturelles"

Les données d'arrêtés de catastrophe naturelle ont été récupérées depuis le site <http://www.data.gouv.fr>. Elle répertorient les arrêtés de catastrophes naturelles depuis 1982. Vous trouverez [ici](#) la définition wikipédia.

En substance, il faut noter qu'il s'agit d'un acte juridique émanant de l'état permettant le déclenchement d'indemnisation systématique des victimes.

Il n'y a pas de caractère automatique (ni à proprement parler météorologique !) à la déclaration d'un arrêté, et on peut imaginer une dimension politique à le faire. Pour être clair, **toutes les catastrophes météorologiques n'ont pas donné lieu à un arrêté de catastrophe naturelle**.

Il s'agit ici donc de nos premières limitations identifiées. En particulier, nous ne disposons pas de toutes les catastrophes météorologiques survenues en France depuis 1982.

Pour autant, en avons nous assez pour permettre la réalisation d'un modèle permettant la prévention d'une catastrophe naturelle ?

Dans la suite du document nous utiliserons indifféremment les termes "arrêtés de catastrophe naturelle" et "catastrophe naturelle".

## 2.2.1. Tableau synthétique des données catastrophes naturelles

Description	Liste des arrêtés de catastrophes naturelles par commune
Fréquence	Quotidien
Période	Juillet 1982 à Avril 2015
Nombre de dimensions	6
Nombre de lignes	149707
Nombre de fichiers	1
Format	Excel
Source	<a href="#">Lien</a>

## 2.2.2. Dimensions

- Code INSEE : Code INSEE de la commune au moment de la promulgation de l'arrêté de catastrophe naturelle, *nous verrons plus tard qu'il a été nécessaire de le mettre à jour.*
- Département : Département de la commune au moment de la promulgation de l'arrêté de catastrophe naturelle.
- Commune : Nom de la commune.
- Périls : Libellé de la ou des catastrophes, exemple : Inondations et coulées de boue.
- Date de début : Date de début de la catastrophe.
- Date de fin : Date de fin de la catastrophe.

### 3. Validation du jeu de données pour la résolution du problème de données : Visualisation de l'agrégation données / catastrophes

Les premières questions que nous nous sommes posées, étaient relatives au positionnement de nos 61 stations météorologiques :

- Est ce qu'avec nos 61 stations météorologiques nous arrivons à couvrir la France ?
- Avons nous les moyens en terme de données de prédire les catastrophes naturelles pour l'ensemble des communes Françaises ?

Pour répondre à ces premières questions nous avons créé le projet [UASB03-catastrophe-visu](#), permettant de traiter ces problèmes d'un point de vue géographique. *Programme développé en Angular/Javascript.*

Dans un premier temps, nous avons dû constituer la carte de France des communes, en nous basant sur les données suivantes au format Geojson : <https://github.com/gregoiredavid/france-geojson> , il s'agit des données à jour au 1 Janvier 2018.

*Le classe [app.component.ts](#) est responsable de l'affichage ci-dessous.*

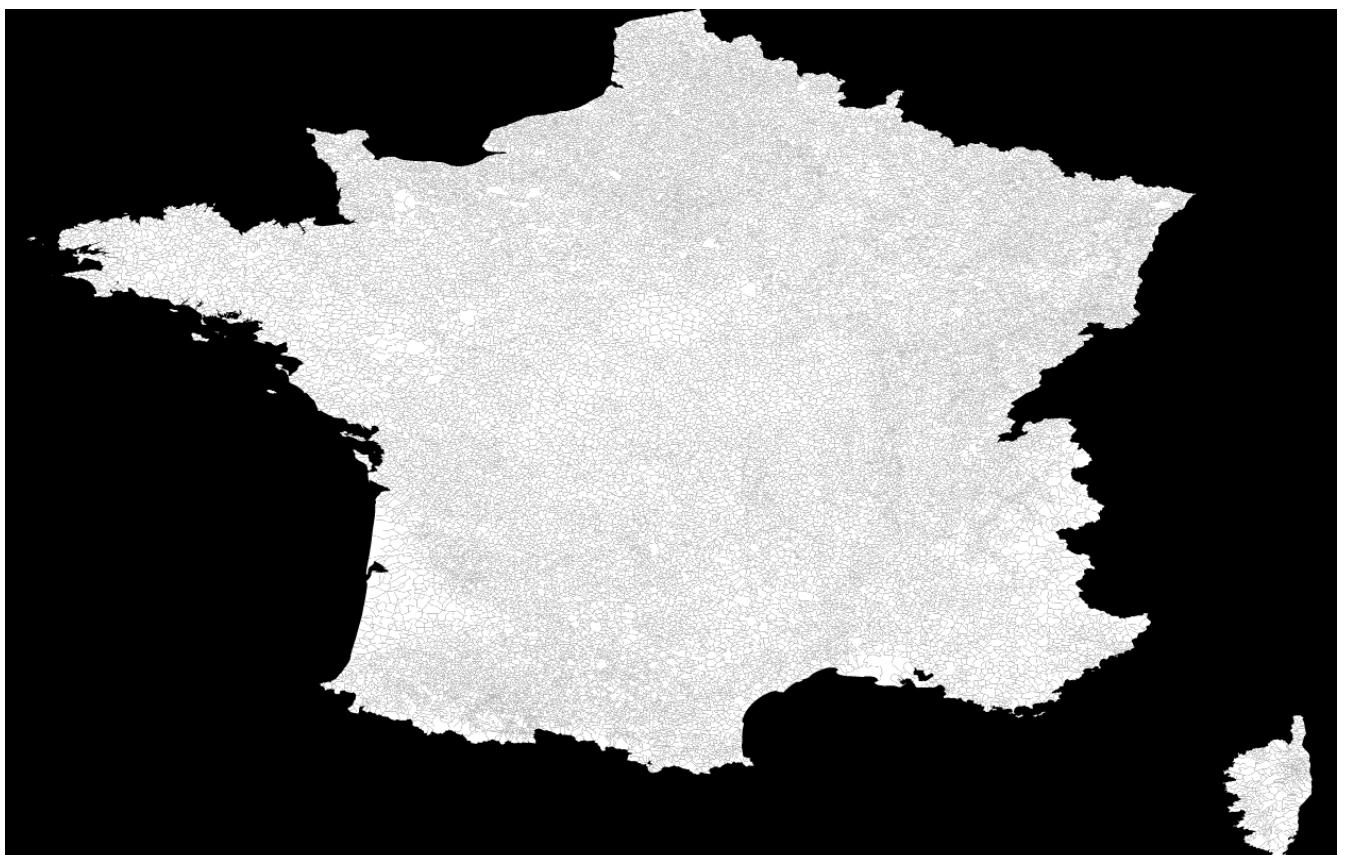


Figure 1. Carte de France des communes

Nous avons ensuite récupéré les coordonnées de nos stations météorologiques au format Geojson

(lien) pour pouvoir les projeter sur la carte de France.

La classe `station.ts` est en partie responsable de l'affichage ci-dessous.

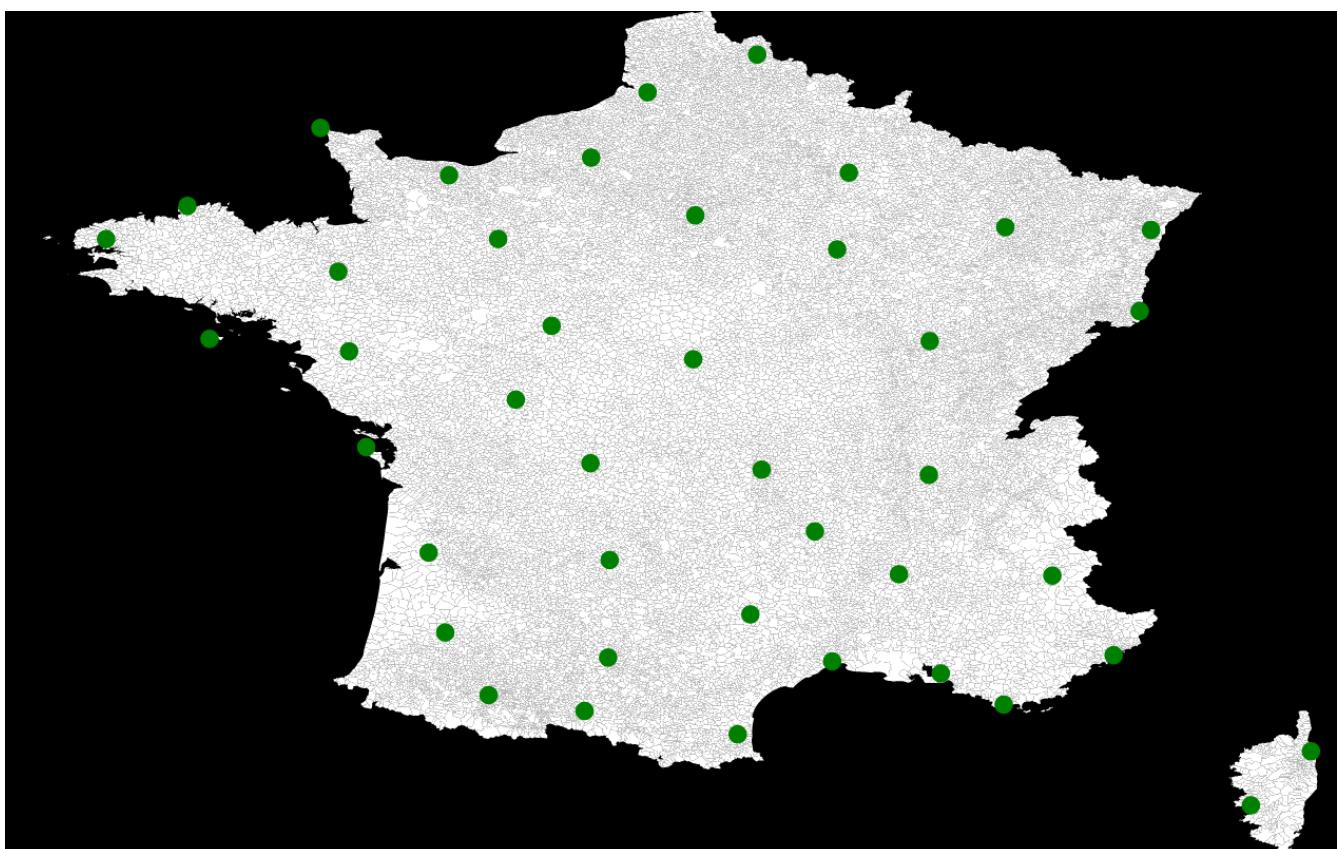


Figure 2. Carte de France des communes avec les stations météorologiques

Nous pouvons constater que 61 stations ne permettent pas de couvrir toute la France. Pour être exact, seulement 43 stations ont été projetées, les autres appartenant au domaine de l'outre-mer. Concernant ces 43 stations, il va nous falloir filtrer les communes pour ne garder que les communes "à proximité" d'une station.

Cependant, nous avons tout de même souhaité projeter l'ensemble des catastrophes sur la carte pour pouvoir faire notre première interprétation de nos données de catastrophes, sous la forme d'une *Heat map*.

Plus il y a eu de catastrophes sur une commune, plus la couleur de sa représentation gagnera en intensité vers la couleur rouge.

Ce travail n'a pu se faire sans un prétraitement des données (*réalisé en Spark*) :

- Des noms de communes n'étaient pas bien orthographiés, exemple : *Costes* au lieu de *Les Costes*
- Normalisation des codes INSEE sur 5 caractères
- 1341 codes INSEE doivent être mis à jour car ils ne trouvent pas de correspondances avec nos données de géolocalisation des communes qui pour rappel correspondent aux données 2018.

Après étude, il apparaît que les codes INSEE des communes ne sont pas immuables. Nos premières données datant de 1982, on constate que les codes communes sont amenés à évoluer dans le temps. Des rattachements sont fait, des communes nouvelles sont créées.

Le site de l'INSEE nous permet de suivre cette évolution, voir figure ci-dessous. Pour automatiser ce traitement nous avons élaboré le parseur [GetINSEEInformation](#) s'appuyant sur

le site de l'INSEE et intégré à Spark sous la forme d'une *User Define Function*.

### *Evolution des codes INSEE dans le temps*

lien

The screenshot shows a web page titled "Commune de Notre-Dame-d'Allençon (49227) commune déléguée". At the top right, it says "GÉOGRAPHIE" and "Date de référence : 01/01/2018". A message box states: "Le code officiel géographique de la commune de Notre-Dame-d'Allençon était 49227.". Below this is a dropdown menu labeled "Modifications (depuis 1943)" which contains the entry "01/01/2017 : Notre-Dame-d'Allençon devient commune déléguée au sein de Terranjou (commune nouvelle).".

Ces tâches de prétraitement ont été réalisées par les programmes *Spark* :

- [Part1CatastropheMajCodeCommunePart1](#)
- [Part2CatastropheMajCodeCommunePart2](#)

Le programme `commune.ts` est responsable de l'affichage ci-dessous.

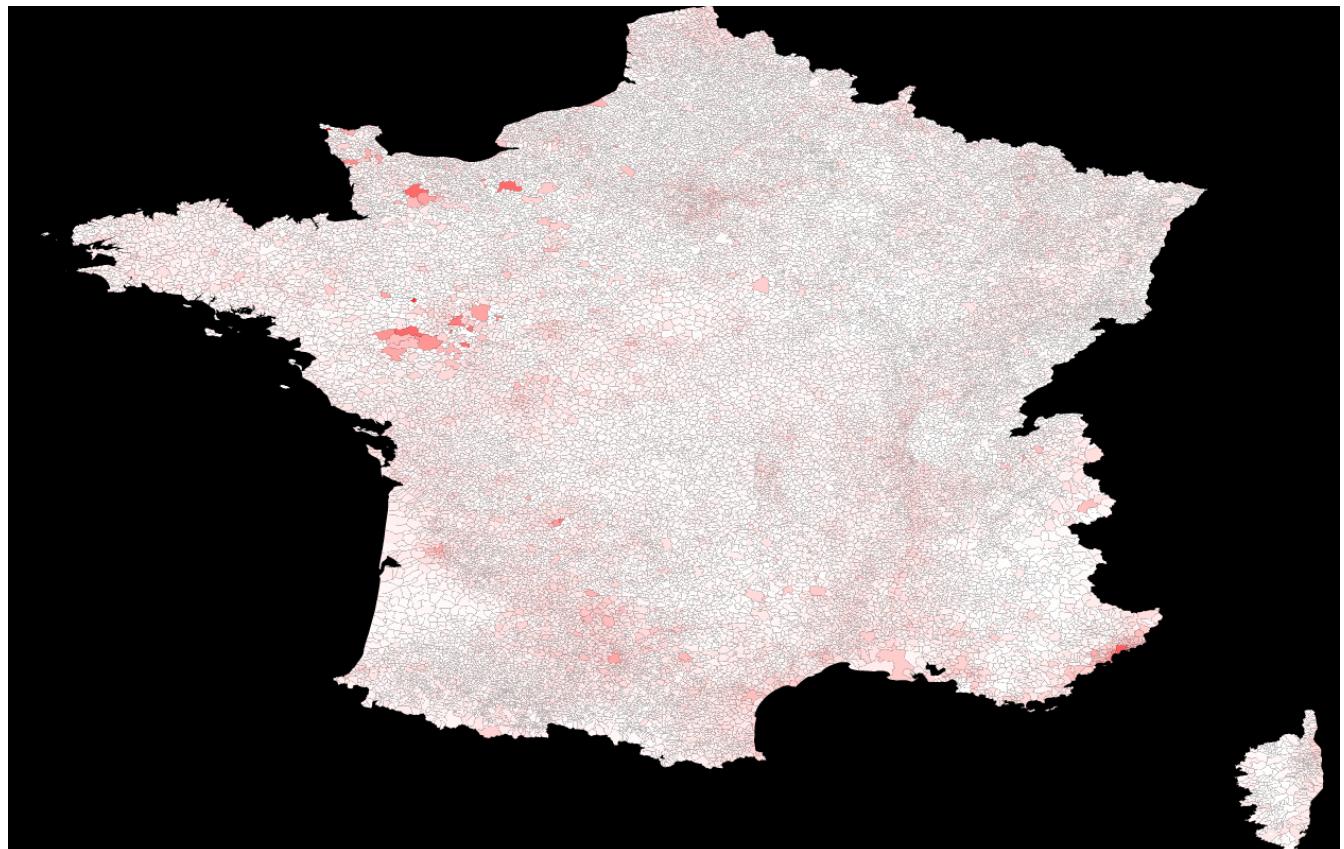


Figure 3. Heat map des arrêtés de catastrophe naturelle depuis 1982

Cette figure nous apprend que les catastrophes naturelles ne sont pas uniformément réparties sur l'ensemble de la France, des régions sont plus impactés que d'autres comme la commune de Nice avec 63 déclarations (le maximum), ou encore Mauges-sur-Loire avec 59 déclarations, contre 2 pour

la commune de *Fontainesbleau*.

Nos stations météorologiques ne couvrant pas toute la France nous avons procédé au filtrage de nos arrêtés de catastrophe naturelle en fonction de la proximité géographique des communes affectées aux stations.

Si nous projetons nos stations, sans filtre nous obtenons la figure ci-dessous.

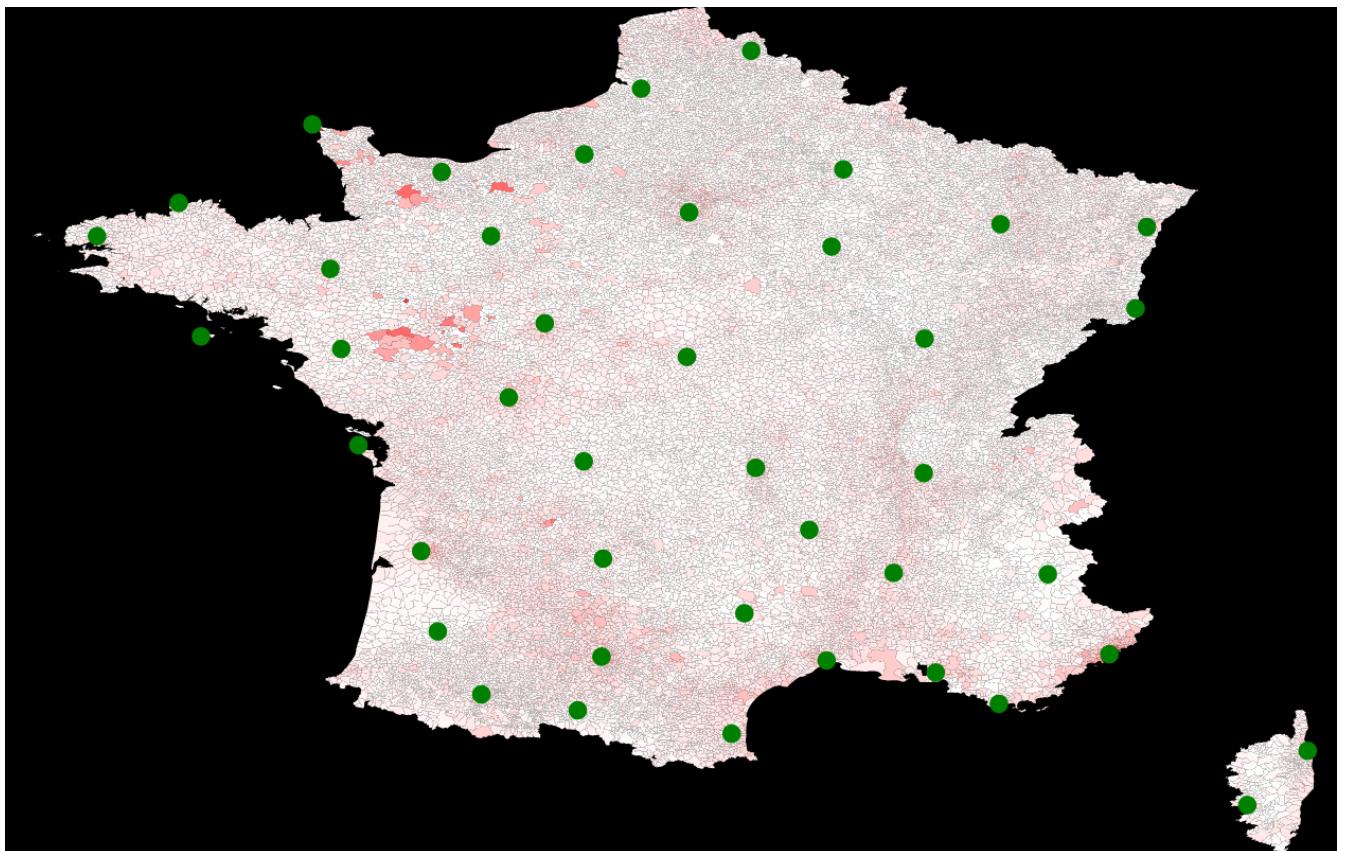


Figure 4. Heat map des arrêtés de catastrophe naturelle depuis 1982 avec le positionnement des stations météorologiques

Pour filtrer les communes nous avons créé le programme [Part3SelectCommunesAroundStations](#) et choisi **un rayon de 30km autour des stations**.

Nous avons jugé que les communes dans ce périmètre étaient des communes fortement influencées par la météo relevée par la station. Bien entendu il s'agit d'une valeur globale qu'il s'agirait de faire varier au travers de différentes itérations, voir de faire varier indépendamment d'une station à l'autre. Il s'agit des axes d'amélioration répertoriés.

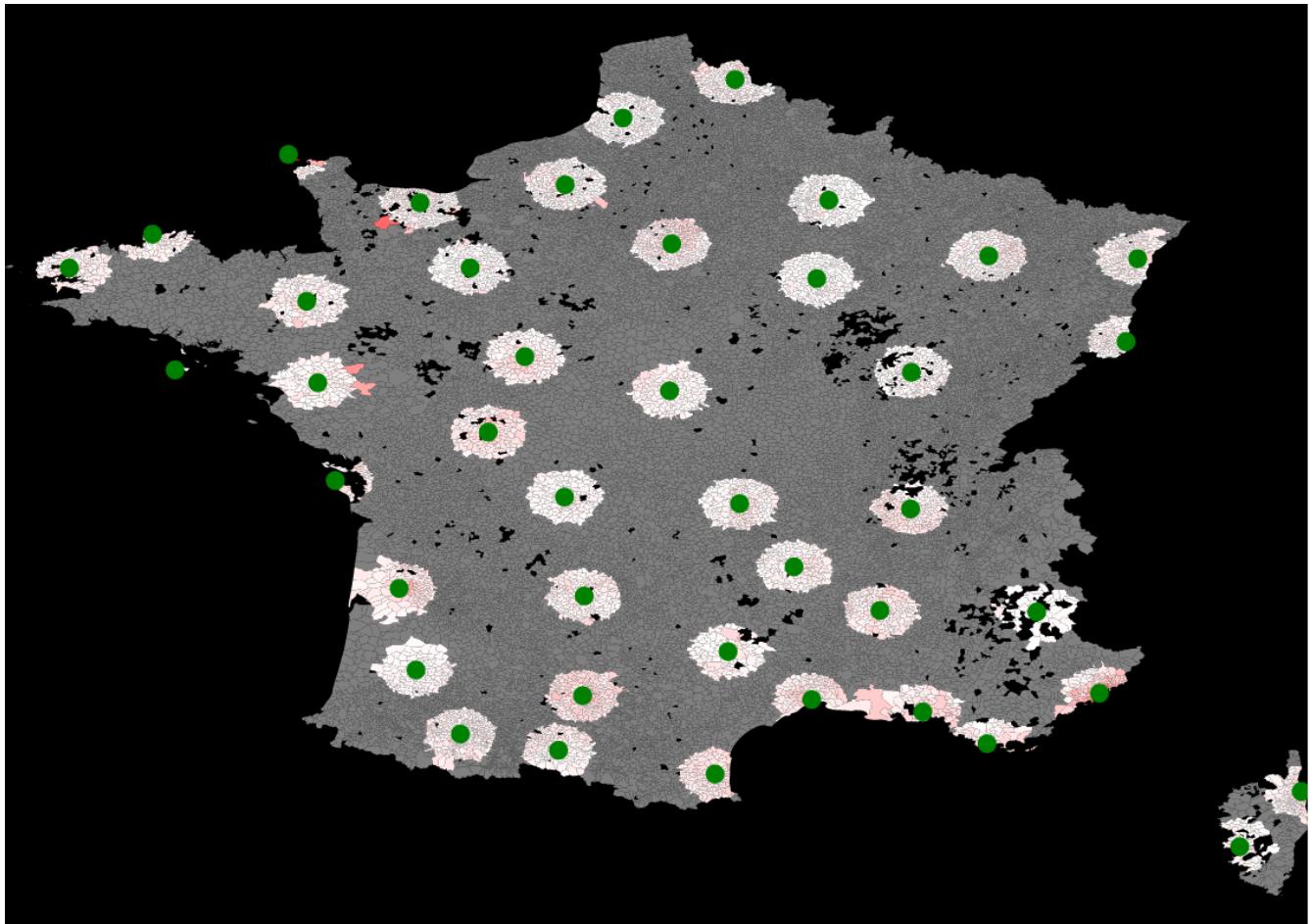


Figure 5. Communes après filtrage (en noir, absent, les communes n'ayant jamais eu d'arrêté de catastrophe naturelle)

# 4. Constitution et analyse de la "donnée utile"

Les données récoltées sont valides d'après le chapitre précédent pour la résolution du problème de données. L'objet de ce chapitre sera donc la constitution d'un jeu de données agrégées pour servir de base à l'élaboration de modèles prédictifs.

Comme demandé, on trouvera la synthèse des analyses exploratoires, prétraitement, études préalables (normalisation, nettoyage des données, gestion de données manquantes, agrégation...)

Cette partie de l'étude est couverte par le notebook [analyse de données](#). En cas de besoin de plus de détails, justifications ou code, se reporter à ce document.

## 4.1. Les données catastrophes

Pour l'analyse suivante nous n'avons pris en compte que les arrêtés de catastrophe naturelle émis sur les communes que nous avons sélectionnées. Bien que le but de notre projet soit la prédiction d'une catastrophe sans tenir compte de son type, nous avons tout de même mené une étude détaillée sur celles-ci.

Pour débuter l'analyse des périls nous avons décomposé les énoncés administratifs qui comporter plusieurs typologies de catastrophe, par exemple : "Inondations et coulées de boue" en 2 catastrophes : "inondations" et "coulées de boue".

Vous trouverez ci-dessous les 10 principaux types de catastrophes.

Le programme [Part5CatastropheSplitPeril](#) permet d'appliquer ce traitement. On constate que les 2 premières typologies sont liées à l'eau et elles sont en terme d'effectif largement plus fréquentes que les autres. La liste de toutes les typologies est disponible en [annexe](#).

*Table 1. Tableau représentant les 10 plus fortes typologies de catastrophe, sur un total de 33*

	<b>Catastrophe</b>	<b>Effectif</b>
1	inondations	106911
2	coulée de boue	103621
3	mouvements de terrain	28512
4	mouvements de terrain differentiels consecutifs a la secheresse et a la rehydratation des sols	15747
5	tempete	15473
6	chocs mecaniques lies a l'action des vagues	6624
7	mouvements de terrain consecutifs a la secheresse	4852
8	glissement	2269

	<b>Catastrophe</b>	<b>Effectif</b>
9	glissement de terrain	2100
10	poids de la neige - chutes de neige	1275

Si on regarde la durée, on est surpris par la durée moyenne de 45 jours. Cependant l'axe médian de 5 jours démontre qu'il existe sans doute une grande disparité dans la durée des catastrophes.

*Table 2. Durée en jour, vue globale*

Min.	1st Qu.	Median	Mean	3rd Qu.	90.	95.	99.	Max.
1.00	2.00	5.00	45.11	5.00	24.0	92.0	1310.0	4029.00

Nous avons, par la suite, poursuivi l'analyse de la durée par type de catastrophe. Le tableau de synthèse est disponible en [annexe](#). Sur le tableau en question on constate la présence d'*outlier* dans notre top 5. Une durée longue n'est pas liée à une typologie, mais peut se trouver sur n'importe quelle typologie.

Le fichier brut récolté sur le site de [des données publiques du gouvernement Français](#), auquel on a ajouté la station SYNOP la plus proche s'il y en a une, et supprimé les catastrophes trop éloignées, est un fichier produit par l'étude du chapitre précédent.

Ce fichier est une donnée d'entrée de cette partie. A ce stade, il n'y a pas de filtrage.

Néanmoins, on remarque que sur l'ensemble des lignes décrites dans la partie description des données, seulement ~70000 lignes sont à une distance inférieure à 30km d'une station.

On trouvera en [annexe](#) la liste des 43 stations concernées par des arrêtés de catastrophes naturelles.

## 4.2. Les données météo

Lors de l'import des nombreux fichiers de données météo décrites plus haut, un filtre sélectionnant les données est appliqué, pour ne sélectionner que :

- la période couverte par les données catastrophe qui est inférieure à celle des données
- les données météo concernées par le sous ensemble des stations proches (30km) des communes dans lesquelles des catastrophes naturelles se sont produites.

Naturellement, les données produites par les stations d'outre mer présentes sont exclues.

L'objectif est d'amoindrir l'impact des 4 millions de lignes sur les ressources, principalement la mémoire vive et le "swap", et d'augmenter la performance de l'import.

A noter, si nous avions disposé d'une infrastructure plus performante, nous aurions aussi considéré ces données filtrées pour améliorer la qualité de nos données et augmenter l'effectif de la classe "non catastrophe". Ce point est inventorié dans le paragraphe : "Pour aller plus loin".

Très rapidement on remarque dans les 2 millions de mesures restantes, les valeurs manquantes sont légion, plus pour certaines variables que pour d'autres. Le tableau descriptif est disponible en [annexe](#).

## 4.3. Constitution du jeu de données "utile"

Eu égard à la qualité des données, du temps imparti et de l'objet de l'étude, on décide de procéder à un certain nombre de simplifications lors de la constitution du jeu de données qui nous permettra d'élaborer nos modèles.

En effet, l'objet du document n'est pas seulement l'analyse de données. Néanmoins, chaque point qui peut être amélioré a posteriori sera référencé dans le chapitre "Pour aller plus loin...".

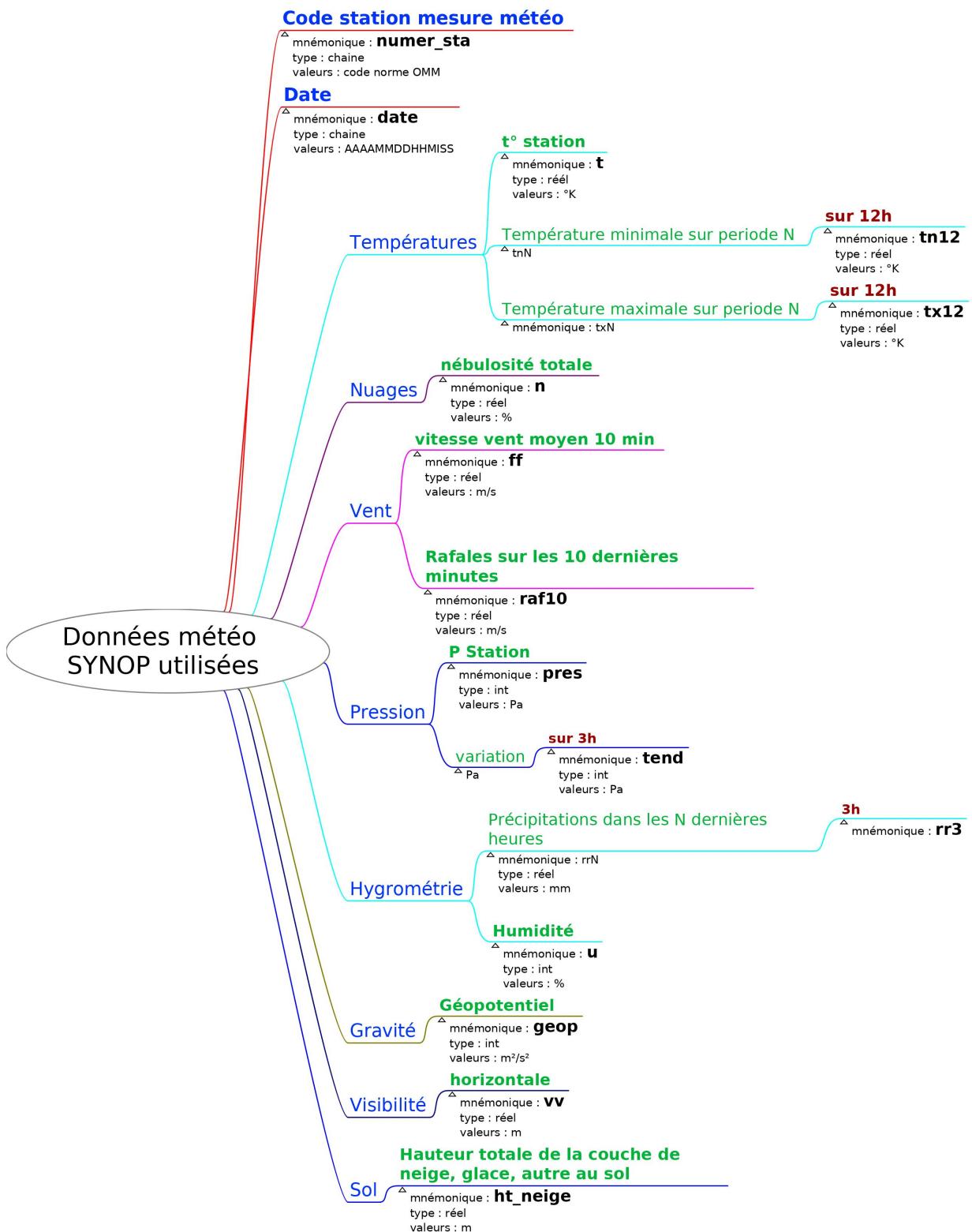
### 4.3.1. Pré traitement des données météo

#### Le filtrage des variables par choix

On choisit de sélectionner un sous ensemble de 13 variables prédictives ('t','tn12','tx12','n','raf10','ff','pres','tend','rr3','u','geop','vv','ht\_neige') et un index temporel( 'date').

On trouvera ci-après les explications des différentes mnémoniques.

*Arbre des données SYNOP sélectionnées pour l'élaboration de nos modèles*



Les critères qui ont présidés à ce choix sont les suivants :

- sélection des indicateurs principaux dans les différentes familles de mesures physique
- Des mesures simples à traiter et à comprendre
  - exclusivement quantitatives
  - unité intelligible

- des valeurs qui tiennent compte de la **temporalité**. En effet, les données SYNOP nous offrent des valeurs sur des périodes distinctes de la période de mesure. Exemples : tn12, tx12, raf10....

Dans une optique d'amélioration, on pourra accroître le nombre de variables considérées, et appliquer des traitements factoriels sur celles-ci. Ce point est mentionné dans le chapitre "Pour aller plus loin...".

### **Le filtrage des individus mesures par choix**

Dès l'import, on ne considère que les mesures issues de stations proches de communes où ont eu lieu des évènements catastrophes. Le choix de distance entre station et commune atteinte d'une catastrophe est un rayon de 30 kilomètres autour de la station.

Sur ce choix, et pour l'anecdote, un échange avec un ingénieur meteoFrance à la cité de l'espace à Toulouse a permis de la valider.

A nouveau, il aurait fallu réinjecter les mesures des autres stations, celles négatives de France métropolitaine.

Quant aux données outre mer, il est bien dommage que nous n'ayons pas disposé de données catastrophes naturelles : Un certain nombre de phénomènes climatiques extrêmes peuvent s'y dérouler, qui vont au delà du simple aspect administratif de l'arrêté en métropole.

### **La gestion des valeurs manquantes**

Une fois filtré, le profil des valeurs manquantes dans les données météo est le suivant :

*Table 3. tableau pourcentage des valeurs manquantes*

<b>mnémonique</b>	<b>% de valeurs manquantes</b>
t	<b>0.08</b>
tn12	<b>75.15</b>
tx12	<b>75.15</b>
n	<b>24.57</b>
raf10	<b>96.72</b>
ff	<b>0.21</b>
pres	<b>0.41</b>
tend	<b>0.49</b>
rr3	<b>4.15</b>
u	<b>0.21</b>
geop	<b>95.87</b>
vv	<b>16.71</b>
ht_neige	<b>85.25</b>

Après avoir essuyé deux échecs en tentant une interpolation et constaté l'évidence que le nettoyage des lignes à valeur manquante vident le dataset de son contenu, on a choisi une stratégie différenciée selon des groupes de variables de nature et quantité de valeurs manquantes similaires.

- Le groupe des valeurs manquantes très faibles (t,ff,pres,tend,u)

La conséquence de leur suppression étant a priori faible pour le dataset, on a choisi de les écarter. En toute rigueur, dans le cas de la recherche de phénomènes exceptionnels, on est en droit de contredire cette stratégie. Néanmoins dans le cadre et les moyens de cette étude, on simplifiera ainsi.

- Le groupe des valeurs manquantes faibles (n,rr3,vv)

On remplace la valeur par la moyenne des valeurs observées. Là encore on introduit un biais.

1. Le groupe des phénomènes météo exceptionnels et valeurs manquantes fortes (raf10, geop, ht\_neige)

On remplace les valeurs manquantes par la valeur 0 supposant que l'absence de valeur découle de l'absence de phénomène.

2. Le groupe des extrêmes de températures sur une période et valeurs manquantes fortes

Dans ce cas on choisit de se reporter à des données présentes de mesures précédentes. On introduit un biais "lissant" les valeurs extrêmes mais c'est un lissage cohérent avec les valeurs météo proches temporellement.

On aboutit, *in fine*, à une perte de 0,8% des individus, que l'on juge négligeable dans le cadre de cette étude.

### Analyse unidimensionnelle

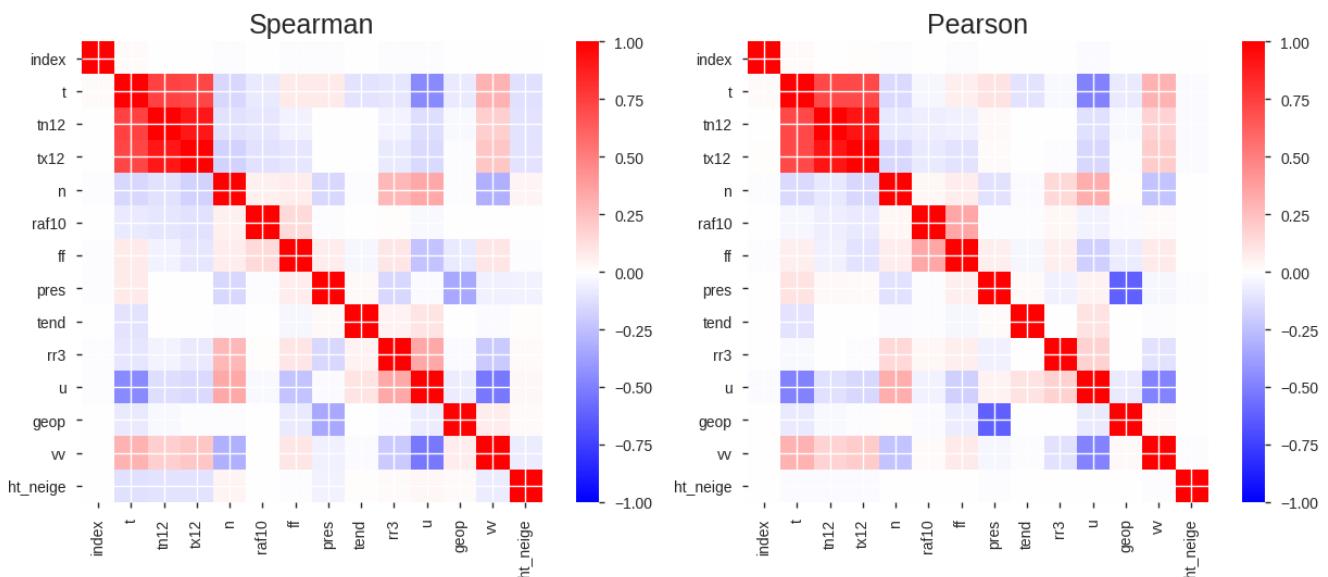
On extrait quelques éléments chiffrés (nombres, moyennes, écarts types, extremums, et quartiles) sur les individus et variables :

On pourra se reporter en [annexe](#) pour quelques règles de conversions des mesures pour que les valeurs de ce tableau soient parlantes.

#### 4.3.2. Analyses bidimensionnelles

Dans cette section on étudie les relations entre les variables q pour compléter l'analyse unidimensionnelle précédente.

Pour ce faire on a recours aux indices de Spearman et Pearson représentés ci dessous :



Ils sont quasi similaires à la corrélation t° / ht\_neige près où le Spearman est plus précis que le Pearson.

Sauf t, tx12 et Tn12, qui sont très corrélées à cause de notre méthode de gestion des valeurs manquantes, les autres variables n'ont pas de corrélation qui justifierait une suppression.

On note les anti-corrélation "modérées" entre :

- humidité et température
- pression et géopotentiel

Tx12 et tn12 représentant des valeurs extrêmes utiles pour les phénomènes extrêmes, nous ne les supprimerons pas.

### 4.3.3. Pré-traitement des données catastrophes

Un premier pré-traitement a été réalisé à l'import des données de l'étape de visualisation : l'ajout de la station la plus proche. Il est décrit dans le chapitre "[Données catastrophes](#)".

L'action de pré-traitement suivante consiste à supprimer :

- les catastrophes survenant sur les communes pour lesquelles nous ne disposons pas de stations SYNOP suffisamment proches.

En effet la carence de données météo nous empêcherait de les modéliser. C'est une limite supplémentaire de notre étude liée d'une part à la nature des données collectées mais aussi à notre critère de choix sur la valeur du "rayon de pertinence météorologique de la station de mesure", fixé à 30km dans le chapitre précédent.

### 4.3.4. Les données agrégées : météo et catastrophes

Le choix d'agrégation ci-dessous sont éminemment discutables. Leur direction est le pragmatisme et la simplicité.

#### Choix d'un stratégie et conséquence sur la temporalité du phénomène

Au cours de l'étude une question épineuse s'est posée : l'arrêté de catastrophe naturelle est administratif ! Sa fréquence est quotidienne là où les données météo sont plus fréquentes.

Considérer la période spécifiée par l'arrêté peut nous mener à des durées aberrantes météorologiquement parlant. Ce qui est vrai pour des phénomènes à inertie comme la sécheresse ne l'est sans doute pas pour une tempête ou une inondation.

D'un autre coté, supprimer les jours de mesures de catastrophes peut empêcher de déterminer ces mêmes phénomènes.

Qui plus est les données météo avant déclaration de l'arrêté peuvent être cruciales car préparant à la survenance du phénomène !

Devant ce dilemme, on choisit donc qu'**une catastrophe sera définie par les variables du jour du début de l'arrêté de catastrophe naturelle**.

Autrement dit, les 8 (1 / 3h pendant 24h...) séries de mesures sur l'ensemble des stations se verront marquées de l'information : catastrophe.

On assure ainsi l'aspect temporel de l'établissement de la catastrophe puisque certaines variables SYNOP contiennent des informations sur l'avant catastrophe (ex: tend, ff, raf10...). La quantité de séries de mesures sur la journée assurent les informations post survenance du phénomène.

On limite également la contradiction des différentes données caractéristiques de différents phénomènes.

## Agrégation

On distribuera l'information catastrophe sur les mesures SYNOP de la journée de début de l'arrêté de catastrophe naturelle.

## Filtrage

Suite au choix simplificateur du paragraphe sur la temporalité du phénomène, se pose une question : "Que faire des données météo couvertes par le reste de la durée de la catastrophe ?"

Il y a trois possibilités :

1. Les considérer comme catastrophes naturelles
2. Les considérer comme non catastrophes naturelles
3. Ne pas les considérer

Nous choisissons d'éliminer les mesures concernées par les autres jours des catastrophes naturelles car nous pensons que selon le type de catastrophe (ou l'objectif de l'arrêté...) les valeurs des variables météo post déclaration de catastrophes peuvent relever de la catastrophe météo ou pas... On peut se ramener à l'analyse de la donnée catastrophe plus haut pour s'apercevoir des disparités en matière de durée d'arrêtés de catastrophes naturelles.

Aussi se pose la question de l'éviction de mesures qui auraient pu être témoin d'un "régime permanent" de la catastrophe... autre point à améliorer.

## Création de la variable à prédire

L'objectif est une classification binaire : les mesures collectées nous annoncent-elles une catastrophe ?

On choisit :

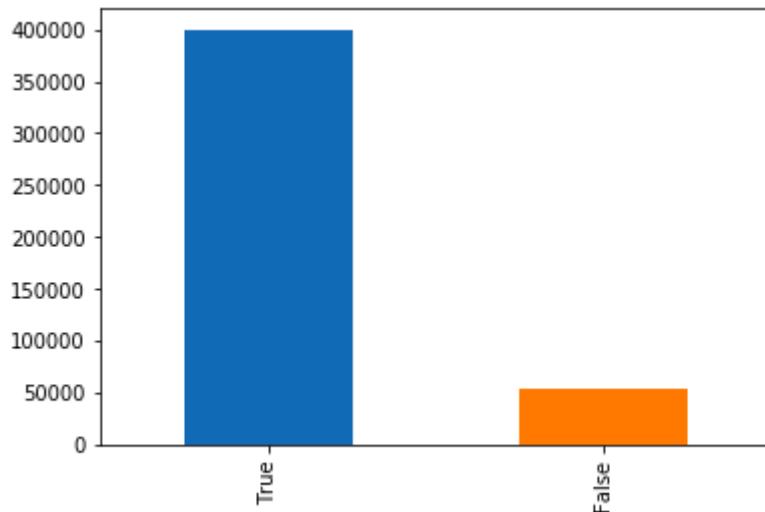
- variables prédictives : les variables seront telles que décrites dans le chapitre "[Le filtrage des variables par choix](#)".
- variable à prédire : la mesure correspond oui ou non à un événement susceptible de faire émettre oui ou non un arrêt de catastrophe naturelle.

## Distribution de la variable à prédire dans le jeu d'apprentissage

## WARNING

l'ensemble de données catastrophe est rare si l'on considère les résultats du chapitre de validation du problème de données... qui considère le nombre de catastrophes par commune. Si l'on suit les choix précédents et que l'on s'intéresse aux dates événements... on se retrouve à traiter un jeu de données déséquilibré.

*Graphe de distribution de la variable à prédire*



## Rééchantillonnage

On choisit de rééchantillonner à la baisse la classe positive (une catastrophe survient) pour Ce choix est guidé par l'impossibilité d'appliquer, avec les ressources disponibles, la méthode de choix décrite ci après.

On aboutit à un jeu de données d'apprentissage / test de 453643 lignes pour 13 variables prédictives.

# 5. Les modèles

Dans cette partie, l'objectif est de fournir "le meilleur" modèle issu de l'application de l'algorithme le plus adapté. On tentera d'optimiser le modèle en le générant par cross validation.

Une fois le meilleur modèle obtenu selon les critères, on tentera d'ajuster finement les hyperparamètres du modèle.

## 5.1. Méthode

Ci-dessous le pseudo algorithme appliqué pour obtenir le meilleur modèle :

1. Génération du jeu d'apprentissage et du jeu de tests

Pour chaque algorithme :

- . apprentissage du modèle

1. prédiction du modèle sur le jeu de tests
2. évaluation de la performance du modèle
  - a. matrice de confusion
  - b. précision, rappel, F1-score
  - c. score aire sous la courbe ROC
  - d. affichage courbe ROC

3. amélioration de la performance du modèle par validation croisée sur l'ensemble des critères

Pour le meilleur algorithme :

- . donner la liste des paramètres à évaluer
- . quadrillage des paramètres avec validation croisée sur l'ensemble des critères
- . estimation de la performance du modèle

## 5.2. Résultats

### 5.2.1. Les évaluations des modèles

Le tableau ci-dessous expose les résultats :

*Table 4. Scores par algorithmes du résultat de la prévision de catastrophe par les modèles générés*

méthode & algorithme/fonction de score	Précision	Rappel	F1-SCORE	ROC_AUC
Random Forest	0.928	0.964	0.946	0.945
Random Forest_CV	0.887	0.962	0.913	0.943

méthode & algorithme/fonction de score	Précision	Rappel	F1-SCORE	ROC_AUC
Random Forest_GS_CV	<b>0.950</b>	<b>0.962</b>	<b>0.956</b>	<b>0.956</b>
Random Forest_GS_CVx2	0.874	0.962	0.904	0.929
<i>Adaptative boosting</i>	0.863	0.939	0.900	0.897
Adaptative boosting_CV	0.877	0.933	0.894	0.931
<i>Gradient boosting</i>	0.873	0.945	0,795	0.908
Gradient boosting_CV	0.866	0.940	0.900	0.936
<i>Naïf bayesien</i>	0.849	0.583	0.691	0.740
Naïf bayesien_CV	0.866	0.573	0.685	0.845
<i>Regression logistique</i>	0.797	0.872	0,833	0.827
Regression logistique_CV	0.814	0.867	0.834	0.882
<i>K-plus proches voisins</i>	0.690	0.757	0.722	0.712
K-plus proches voisins_CV	0.687	0.758	0.719	0.768
<i>Support Vector Machine</i>	1	0,10	0,18	0.550
Support Vector Machine_CV	<b>0,997</b>	0,08	0,15	0,589

légende

1. en italique : les scores des modèles obtenus *sans cross validation*

- suffixé par \_GS : les scores du modèle auquel on a appliqué des hyperparamètres obtenus par "gridsearch"

L'algorithme duquel est issu le modèle qui conduit aux meilleures évaluations est l'algorithme de forêt aléatoire. En effet, le modèle arrive assez nettement en avance sur les autres algorithmes en ce qui concerne les différents critères sélectionnés pour un problème de classification binaire.

A ce sujet, l'indicateur ROC nous paraît plus important vu que nous avons équilibré, même artificiellement, notre jeu de données et que notre attention va plus aux faux négatifs qu'aux faux positifs.

Ces résultats ne nous étonnent pas outre mesure :

- Les méthodes ensembles ont les meilleurs résultats
- Les random forests restent très efficaces en cas de jeu de données déséquilibré

**TIP** On note qu'une deuxième passe de validation croisée sur le modèle (déjà élaboré avec une cross validation) donne des résultats sous optimaux.

### 5.2.2. Les hyperparamètres du meilleur algorithme

On expose ci-dessous les résultats de l'étude des hyperparamètres du modèle "random forest" obtenus par "quadrillage en validation croisée" :

- 500** arbres (paramètre n\_estimators)  
Sur ce point, chaque évaluation effectuée a été proportionnelle à la valeur de ce paramètre. Seule la puissance de notre infrastructure donc le temps de calcul nous a limité à cette valeur.
- fonction d'évaluation de la qualité de la "coupure" de l'arbre fixée à "**entropie**" (paramètre "criterion").
- Les autres paramètres ont été validés à leurs valeurs par défaut dont le paramètre de "split" qui considère **toutes les variables**.

Les résultats de l'évaluation de ce modèle sont fournis dans le tableau précédent.

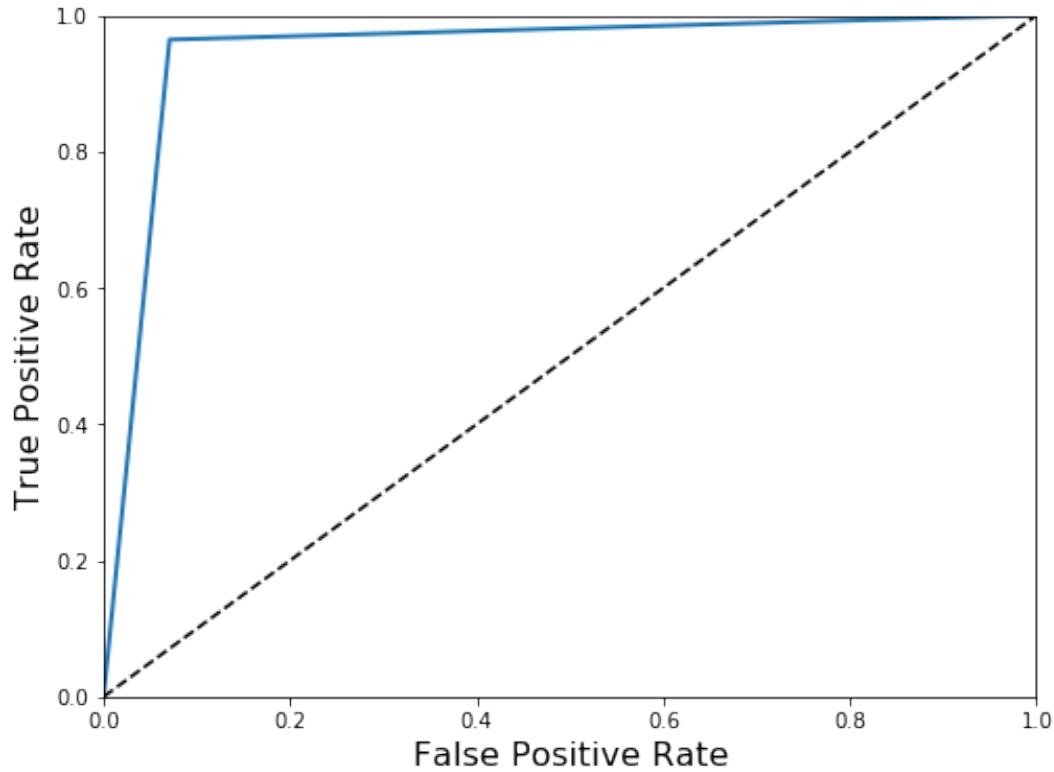


Figure 6. Graphe de la courbe ROC du "meilleur" modèle

Plus de détails sur les hyperparamètres (notamment par défaut) du "random forest" sont disponibles dans la [documentation de scikit-learn](#).

On trouvera également plus de détails sur la méthode de quadrillage avec validation croisée [ici](#).

# 6. Architecture de la solution

Suite à l'exploration des données et la sélection de la meilleure méthode de prédiction pour nos catastrophes naturelles, nous allons à présent mettre en place une architecture permettant sa mise en oeuvre.

Nous savons que nous recevons périodiquement les données météorologiques, indépendamment par station. Dès réception nous souhaitons prédire si une catastrophe naturelle va survenir.

La figure ci-dessous donne un aperçu de celle-ci. L'architecture se base sur InfluxDB pour le stockage des mesures, Grafana pour exploration de la base et enfin de Spark Streaming pour les aspects traitements.



Figure 7. Vue de l'architecture

Nous n'avons pas approfondi les aspects : réception des mesures et émission des alertes, pour nous concentrer sur l'essentiel : le stockage et la prédiction.

## 6.1. InfluxDB

Nos données étant des séries temporelles, nous nous sommes orientés vers InfluxDB pour leur stockage. Cette base s'entoure d'un écosystème appelé *TICK* :

- Telegraf : Il s'agit de la solution pour injecter des données au sein d'InfluxDB, nous l'aborderons plus en détail dans la section dédiée aux connecteurs
- InfluxDB : La base de données orientée *time series*
- Chronograf : Outil de supervision
- Kapacitor : Moteur de traitement de flux de données

Après avoir étudié cette stack en détail, nous n'avons pu sélectionner Kapacitor pour notre architecture. Nous souhaitions le positionner pour le traitement du flux de données mais il ne propose pas de connecteurs vers Spark nativement, et, bien que le projet soit OpenSource, il existe encore peu de documentation.

De plus, Kapacitor n'offre pas de bibliothèque pour permettre la réalisation d'algorithme de prédiction.

Autre point, il semble qu'InfluxDB ait la possibilité de pousser de la donnée dans Kapacitor afin de traiter la donnée sous la forme d'un flux.

Sans l'utilisation de Kapacitor et en gardant notre architecture de départ, nous allons devoir aller chercher la donnée périodiquement.

### 6.1.1. Mise en oeuvre : InfluxDB, Grafana et Telegraf

On rappelle quelques commandes pour créer un environnement technique :

- Création d'un réseau docker *influxdb* :

```
docker network create influxdb
```

- Lancement d'une instance InfluxDB :

```
docker run -d --name influxdb --net=influxdb -p 8086:8086 -v /home/developper/admin/influxdb:/var/lib/influxdb influxdb
```

- Lancement de Telegraf :

```
docker run -d --name telegraf --net=influxdb -v /home/developper/dev/CNAM-UASB03/code/InfluxSparkJava/src/main/resources/telegraf.conf:/etc/telegraf/telegraf.conf:ro telegraf
```

- Lancement de Grafana :

```
docker run -d --name grafana -p 3000:3000 grafana/grafana
```

### 6.1.2. Injection des données

Pour l'injection des données, l'utilisation de *Telegraf* est un pilier du succès d'InfluxDB grâce aux grands nombres de connecteurs qu'il propose, le [repository git](#) sur le sujet est impressionnant.

Si nous devions spéculer sur les composants permettant de collecter les données au fil de l'eau pour les insérer dans InfluxDB, nous pourrions imaginer deux approches :

- L'approche simple, la station insère directement en IQL
- Une approche utilisant un protocole supplémentaire dédié aux objets connectés : les stations météorologiques utilisent le protocole **MQTT** pour faire parvenir les données, via un **MOM** type **RabbitMQ**.

Alors nous utiliserions la configuration *Telegraf* ci-dessous pour les insérer en base.

```
[[outputs.influxdb]]
urls = ["http://influxdb:8086"]
database = "ua-telegraf-rle"
[[inputs.mqtt_consumer]]
servers = ["tcp://meteofrance:1883"]
qos = 0
connection_timeout = "30s"
topics = [
    "meteo/measure"
    persistent_session = false
    client_id = ""
    data_format = "influx"
]
```

En [annexe](#), un exemple de configuration Telegraf simple dont nous nous sommes inspirés.

Hormis *Telegraf*, qui insère les données, InfluxDB propose une API REST pour la manipulation des données ou une interface en ligne de commande de type shell.

Via les 2 canaux, un langage d'interrogation : l'[InfluxQL](#).

Dans le cadre de notre projet nous avons décidé de créer notre propre injecteur en Java qui s'appuie sur le driver [influxdb-java](#) qui est une surcouche de l'API REST native. Il s'agit du programme [influxdb-injector](#). Nous avons créé notre programme non pas parce qu'il fallait insérer en masse les données dans InfluxDB, pour cela nous disposons de notre script python fourni en paramètre, mais parce que nous souhaitions simuler un flux de données. Il s'agit donc d'insérer périodiquement les données.

*influxdb-injector* prend en paramètre un fichier, comme celui-ci [meteo\\_stations\\_catastrophes.csv](#), puis insérera les données d'une même date par lot toutes les X secondes/minutes selon le paramétrage. A la cible, nous sommes sur un insertion toutes les 3 heures des données, il s'agit de la fréquence des données SYNOP, cependant pour les besoins de développement et de tests nous avons volontairement inséré les lots de données SYNOP, tous les 1/4 d'heure voir toutes les 10 secondes.

La tâche [CreateDatabase](#) est responsable de la création de la base de données *meteo* avec une *retention policy aRetentionPolicy* que nous détaillerons dans la section *Passage à l'échelle*. Le programme [Injector](#) est responsable de la simulation du flux de données.

Pour contrôler son bon fonctionnement, nous avons mis en oeuvre l'outil [Grafana](#) dont vous trouverez la copie d'écran du tableau de bord que nous avons créé. Il nous permet de comptabiliser le nombre d'insertion que nous effectuons toutes les 1/4 d'heure.

Requête InfluxQL présente dans la configuration Grafana :

```
SELECT count("station") FROM "meteo_measure" WHERE time >= now() - 15m GROUP BY time(15m)
```

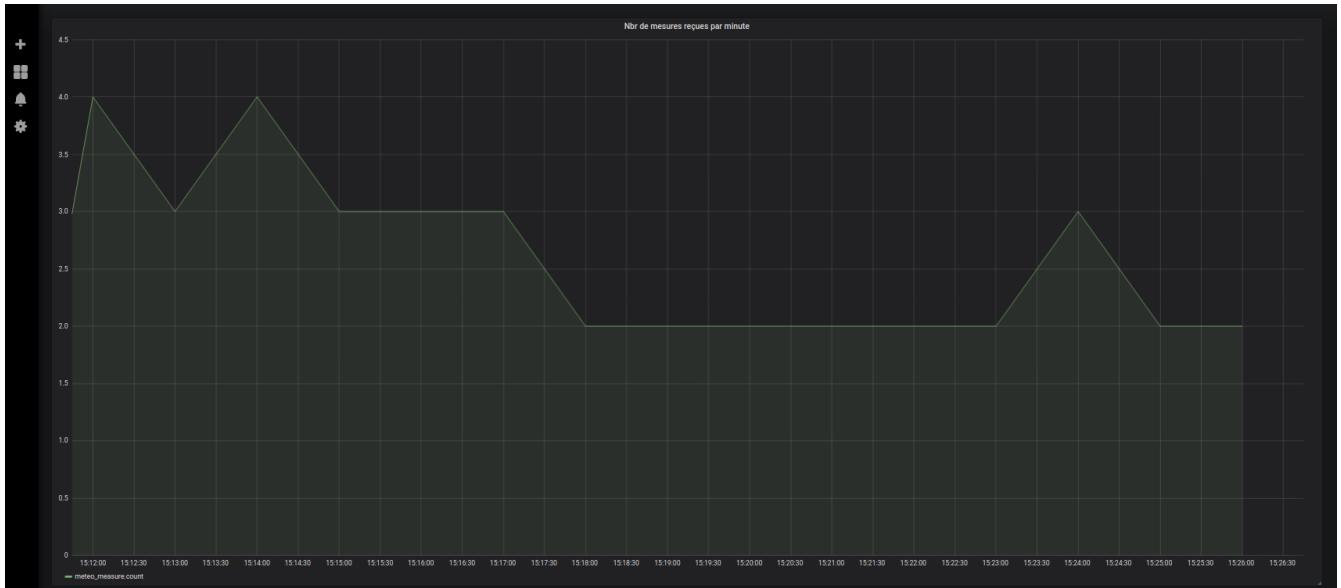


Figure 8. Monitoring de notre injecteur de données

### 6.1.3. Passage à l'échelle

InfluxDB au travers de son offre *Entreprise* permet la création d'un cluster, sur lequel il convient de distinguer les noeuds qui hébergent les métadonnées (définition des *users*, *retention policies* ...) et les noeuds hébergeant les données.

#### la notion de *shard* InfluxDB

InfluxDB, *time series database*, n'offre qu'un champ pour permettre le partitionnement des données, c'est le champ *time* qui est LE champ obligatoire au sein d'une table InfluxDB.

Comme son nom l'indique c'est le champ qui définit l'instant au sein des données, dans notre cas il sera peuplé avec la datation contenue dans nos données SYNOP.

On comprend mieux pourquoi la stratégie de partitionnement se définit au travers de la notion de *retention policies* et de *shard duration*.

.Ci dessous une définition créée pour nos données météos :

```
// DURATION 30 jours
// SHARD DURATION : 3 heures
// Nombre de replica : 1 (au delà il faut un cluster InfluxDB Enterprise)
// Retention policy par défaut de la base de données : oui
influxDB.createRetentionPolicy("aRetentionPolicy", DBNAME, "30d", "3h", 1, true);
```

Cette définition nous permet de dire que nous souhaitons garder nos données 30 jours, elles seront regroupées physiquement en sous groupe de 3 heures. C'est l'ensemble du sous groupe qui est répliqué au travers du cluster, et qui est amené à être supprimé au bout de 30 jours.

Comme dans *elasticsearch*, les *data node* jouent aussi le rôle de coordinateurs/proxy: Si le *data node* interrogé ne possède pas la partition de données il se chargera d'aiguiller la demande vers le "bon" noeud, si une opération porte sur différents *shard* il fera l'agrégation du résultat avant de faire la réponse. On renvoie à la [documentation InfluxDB](#) pour plus de détails.

Dans notre cas, prenons l'exemple d'une requête sur un intervalle de temps de plus 5 heures,

comme nos shards ne représentent que 3 heures de données il y a donc 2 shards groups impliqués dans le résultat de cette recherche. Dans un cluster, les shards sont susceptibles d'être sur 2 machines différentes.

A noter, qu'InfluxDB ne fournit pas de *load balancer* en frontal de ses *data node*, il faut mettre en place sa propre solution. Pour une application cliente comme la notre il conviendra de pointer sur un *load balancer* en amont pour la gestion du *failover* au sein de notre architecture.

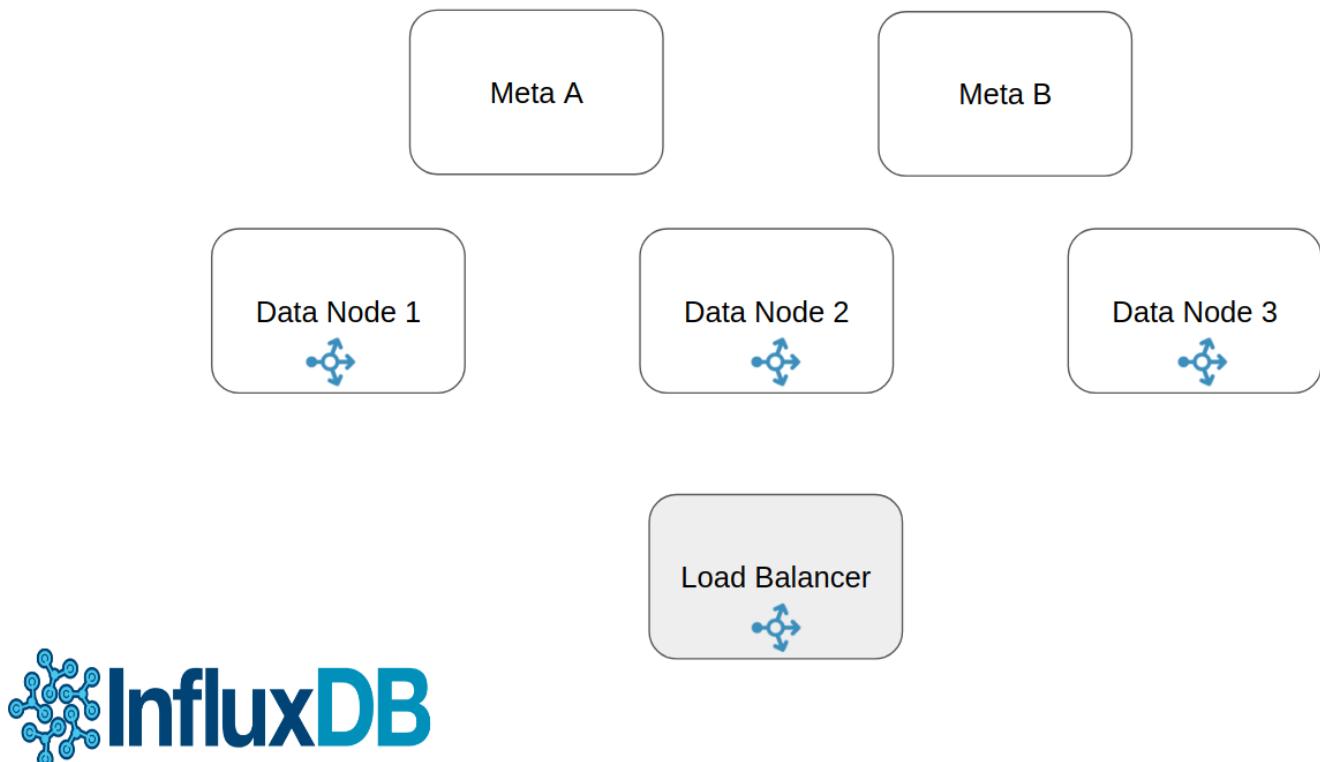


Figure 9. Cluster InfluxDB

## 6.2. Spark Streaming

Pour la réalisation de la partie prédition nous avons opté pour Spark Streaming. Notre besoin est de pouvoir constituer des datasets toutes les 3 heures pour y appliquer un algorithme de prédition. Pour se faire nous avons créé le programme [influxdb-streaming](#).

### 6.2.1. Crédation du modèle de prédition

Avant de rentrer dans le détail, l'une de nos premières étapes a été de créer un modèle exploitable depuis notre programme [influxdb\\_streaming](#).

Nous avons donc réalisé le programme [spark-model-trainer](#) pour créer notre modèle de prédition basée sur l'approche *Random Forest*.

Sur ce point nous avons rencontré des difficultés : Lors du travail en Python nous avons été amenés à manipuler des hyperparamètres comme *min\_split*. **En Spark on ne trouve pas d'équivalent !** Il faut donc re-paramétrer son modèle malgré une analyse précédente plus fine en Python. Une solution est envisagée dans un [chapitre ultérieur](#).

## 6.2.2. L'application *influxdb\_streaming*

### A la rechercher des données

La première difficulté que nous avons rencontrée a été de rechercher comment notre flux de données pouvait être alimenté depuis InfluxDB.

Après avoir recherché une solution en vain, nous avons codé notre propre solution :

Un [Receiver Spark Streaming](#), via la classe [InfluxDBReceiver](#).

Cette classe a pour rôle d'aller périodiquement rechercher des données, dans notre cas toutes les 10 secondes, en exécutant la requête ci-dessous.

```
Query query = new Query("SELECT * FROM meteo_measure where time > now() - 10s",  
"meteo");  
...  
store(data.iterator());
```

Une fois les données acquises elles sont stockées au sein du cluster Spark. Notre configuration ci-dessous permet :

- Le stockage dans la mémoire de la JVM (La Heap) des données constituant le RDD. Si les données ne peuvent tenir en mémoire, la partition concernée sera stockée sur disque. Les données stockées sur disque seront lues à la demande (si besoin).
- Répliquer les partitions sur 2 noeuds du cluster pour prévenir un crash de notre programme

### Configuration du stockage

```
public InfluxDBReceiver() {  
    super(StorageLevel.MEMORY_AND_DISK_2());  
}
```

### Le streaming

Une fois la problématique de l'obtention des données et de leurs sécurisation terminée, reste à appliquer le modèle de prédiction toutes les 3 heures.

Spark Streaming est une solution permettant en continu d'effectuer périodiquement des batchs. Dans notre code (ci-dessous), l'intervalle est de 3 heures.

```
final JavaStreamingContext jssc = new JavaStreamingContext(conf, Durations.hours(3));  
JavaReceiverInputDStream<MeteoMeasure> mesuresStreaming = jssc.receiverStream(new  
InfluxDBReceiver());
```

Faisons le lien avec la précédente section concernant notre receiver:

Nous avons un receiver qui toutes les 10 secondes va aller chercher en base de données les données insérées depuis les 10 dernières secondes pour les stocker au sein du cluster Spark.

Puis nous avons défini un batch récurrent qui toutes les 3 heures va constituer depuis les données collectées par le receiver, un DStream, semblable à un RDD, sur lequel nous pourrons appliquer notre modèle de prédiction.

C'est la classe [CatastrophePredictor](#) qui est responsable de l'ensemble de ce traitement.

### 6.2.3. Mise en oeuvre

Lancement du noeud manager : `sh $SPARK_HOME/sbin/start-master.sh &`

Lancement de 2 Workers :

*Configuration : Lancement de 2 Workers par noeud*

```
export SPARK_WORKER_INSTANCES=2
```

*Configuration : Lancement de 2 Workers esclave chacun utilisant 2 core et 1 Go de mémoire*

```
./sbin/start-slave.sh spark://developper-VirtualBox:7077 --cores 2 --memory 1
```

## 6.3. Passage à l'échelle

Le passage à l'échelle est assuré pour la partie stockage par le cluster InfluxDB et pour la partie traitement par le cluster Spark.

Toutes les 10 secondes, une requête émanera du cluster Spark pour collecter les données météorologiques. Cette requête sera globale sur la période des 10 secondes passées, elle sera donc redirigée vers un des membres du cluster InfluxDB responsable du shard quelque soit le nombre de station météo conservé. Rappelons que c'est la donnée de temps qui est l'unique indice de shard pour InfluxDB via la définition de *retention policies*.

Si ce projet devait passer à une échelle mondiale :

la prédiction de l'ensemble des catastrophes se ferait à partir de la collecte de toutes les stations météorologiques mondiales. Les ressources allouées au cluster Spark devront être augmentées.

Dans le cas contraire, constaterons un dégradation des performances dûe à 2 facteurs :

- Les données étant de plus en plus volumineuses l'espace en mémoire est saturé par notre receiver, Spark commence à se décharger sur le disque dur. Lors du déclenchement de la phase de prédiction, Spark doit accéder à la donnée sur le disque dur, les temps de lecture sont lents, le temps d'exécution de la tâche s'en trouve alors dégradé.
- Le nombre de core/thread n'ayant pas évolué, le traitement deviendrait plus long suite à l'empilement des tâches pourtant parallélisable.

On veillera donc à avoir suffisamment de mémoire vive, et à allouer de nouveaux threads à notre batch.

Si l'on cherche à atteindre une parallélisation des tâches maximum de la part de Spark, on lui allouerait un nombre de thread au moins égal au nombre de stations pour avoir les meilleures performances possibles en terme de temps d'exécution. Pour cela il faudrait admettre l'hypothèse d'indépendance des mesures météo d'une station à une autre.

Concernant InfluxDB, les contraintes sont moins fortes, on cherche uniquement de très bonnes performances pour l'écriture et l'interrogation des données uniquement sur les 10 dernières secondes. InfluxDB, comme beaucoup d'autres bases de données gère un mécanisme de *write ahead log* sur lequel s'applique un cache mémoire qu'il conviendra de paramétrer convenablement pour assurer de très bonne performances.

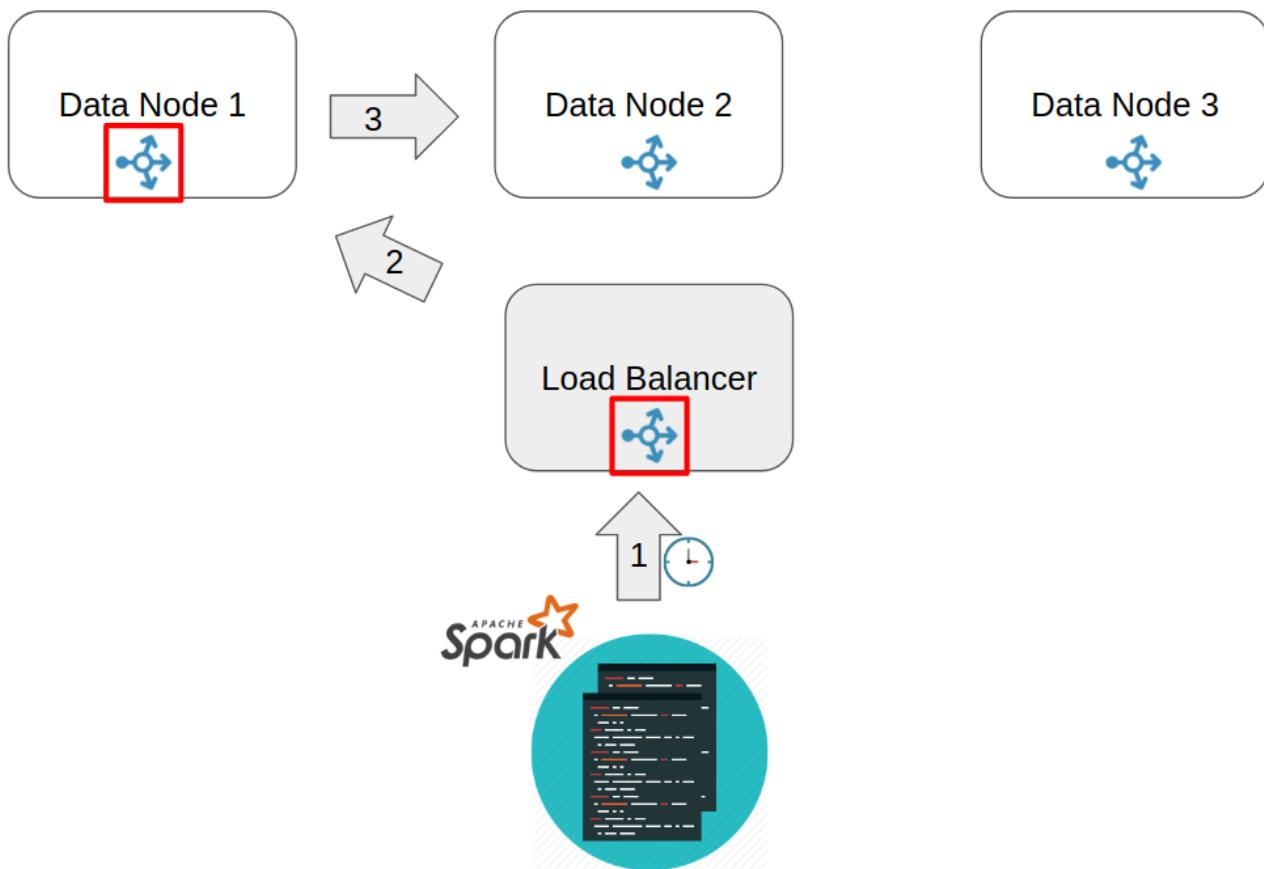


Figure 10. Spark Streaming

Ce qui n'est pas optimal dans notre architecture, c'est le polling de 10 secondes pour aller chercher la donnée.

Idéalement la donnée doit être poussée au sein de notre cluster Spark.

Une première solution consisterait à étudier la méthode de surveillance de Kapacitor des tables InfluxDB pour l'insérer dans notre code Spark Streaming. On peut imaginer un système d'enregistrement du consommateur Spark Streaming et d'émission d'évènement lors de la modification de la table scrutée. Cet évènement déclencherait l'exécution du modèle et la prédiction.

A défaut, on pourrait se fonder sur Kafka par exemple. Spark est déjà outillé avec au sein du framework un [receiver pour Kafka](#).

## 7. CONCLUSION

Cette étude nous a permis, outre répondre aux imposés de l'exercice, d'expérimenter presque tous les volets d'un projet de data science... auquel normalement participent plusieurs équipes.

La visualisation nous a permis de valider efficacement notre capacité à résoudre notre problème de données et d'envisager le passage à l'échelle.

L'analyse des données fut ardue par les biais initiaux (agrégation administratif et physique), liés aux données (valeurs manquantes, filtrages d'individus et de variables...), liés aux algorithmes (pas d'algorithmes "times series", limitations liées aux ressources machine...) et n'a sans doute pas donné l'information optimale à l'étape architecture.

Le fait de ne pas utiliser le modèle généré et le recalculer en Spark java peut remettre en cause l'efficacité du modèle en question, n'ayant pas les mêmes leviers de paramétrage.

L'architecture, si elle s'est révélée efficace et "scalable", est limitée dans l'appréhension de l'écosystème, des capacité de répartition et d'interactions de et à influxDB.

C'est d'ailleurs pour cela qu'un certain nombre de chantiers, de l'ordre de l'optimisation, de l'architecture ou purement fonctionnel seront, sinon traités, évoqués dans le paragraphe suivant.

En effet, on remarquera que l'agrégation de données physiques avec des données plus administratives ou juridiques en tout cas découlés de la météorologie n'est pas forcément fiable. Cela crée un biais très en amont de notre solution.

De plus, notre approche globale du problème de bétouin en matière de météo, c'est à dire de ne pas considérer les spécificités des catastrophes, des lieux où elles se produisent combinée avec un sous ensemble de variables météo, est forcément questionable.

Néanmoins, nous avons réussi à atteindre les objectifs que nous sommes fixés et un résultat...ce qui reste encourageant !

# 8. Pour aller plus loin ...

Ci-dessous des pistes d'améliorations qui n'ont pu être traitées.

## *Améliorations*

- Considérer toutes les mesures de l'ensemble des stations métropolitaines.
- Utiliser le modèle généré en python en PySpark.
- Passer d'une approche globale de la problématique à une approche par station :
  - Variation du rayon d'influence des stations par station
  - Un modèle de prédiction par station
- Passer d'une approche globale de la problématique à une approche par catastrophe :
  - Un modèle de prédiction par catastrophe
- Réalisation d'une classification de l'ensemble de nos données météorologiques quotidiennes:
  - **K-Means / CAH éventuellement complémentées par de l'analyse factorielle**
  - recherche du "bon k".
- Pour l'algorithme random forest, fournir :
  - Le nombre de classes
  - l'importance des variables
- Utiliser pleinement la stack TICK :
  - Telegraf pour l'injection
  - Kapacitor pour l'alerting
  - Chronograf à la place de Graphana pour la visualisation
- Améliorer la méthode de "scrutation" d'InfluxDB (polling)

## *Extensions*

- Récupérer les catastrophes naturelles outre mer
- Etudier voire utiliser les "continuous queries" InfluxDB
- Envisager le coût du sinistre, le nombre de victimes...
- Prendre en compte la nature catastrophe, la prédire...
- Création d'un modèle de prédiction à l'aide d'une méthode "TimeSeries"

## **9. ANNEXES**

### **9.1. Quelques règles de conversion**

- Températures

$$t(^{\circ}\text{C}) = t(\text{K}) - 273.15$$

- Vitesse du vent

$$\text{ff en km/h} = 3,6 * \text{ff en m/s}$$

- Pression

$$1\ 013,25 \text{ hPa} = 101\ 325 \text{ Pa} = 1,013\ 25 \text{ bar} = 1 \text{ atm}$$

### **9.2. Description des variables des données météo**

**Paramètres inclus dans les fichiers de données SYNOP**

Descriptif	Mnémonique	type	unité
Indicatif OMM station	numer_sta	car	
Date (UTC)	date	car	AAAAMMDDHHMISS
Pression au niveau mer	pmer	int	Pa
Variation de pression en 3 heures	tend	int	Pa
Type de tendance barométrique	cod_tend	int	<a href="#">code</a> (0200)
Direction du vent moyen 10 mn	dd	int	degré
Vitesse du vent moyen 10 mn	ff	réel	m/s
Température	t	réel	K
Point de rosée	td	réel	K
Humidité	u	int	%
Visibilité horizontale	vv	réel	m
Temps présent	ww	int	<a href="#">code</a> (4677)
Temps passé 1	w1	int	<a href="#">code</a> (4561)
Temps passé 2	w2	int	<a href="#">code</a> (4561)
Nebulosité totale	n	réel	%
Nébulosité des nuages de l'étage inférieur	nbas	int	octa
Hauteur de la base des nuages de l'étage inférieur	hbas	int	mètre
Type des nuages de l'étage inférieur	cl	int	<a href="#">code</a> (0513)
Type des nuages de l'étage moyen	cm	int	<a href="#">code</a> (0515)
Type des nuages de l'étage supérieur	ch	int	<a href="#">code</a> (0509)
Pression station	pres	int	Pa
Niveau barométrique	niv_bar	int	Pa
Géopotentiel	geop	int	m2/s2
Variation de pression en 24 heures	tend24	int	Pa
Température minimale sur N heures	tnN	réel	K
Température maximale sur N heures	txN	réel	K
Température minimale du sol sur 12 heures	tminsol	réel	K
Méthode mesure tw	sw	int	<a href="#">code</a> (3855)
Température du thermomètre mouillé	tw	réel	K
Rafales sur les 10 dernières minutes	rafl0	réel	m/s
Rafales sur une période	rafper	réel	m/s
Période de mesure de la rafale	per	réel	min
Etat du sol	etat_sol	int	<a href="#">code</a> (0901)
Hauteur totale de la couche de neige, glace, autre au sol	ht_neige	réel	m
Hauteur de la neige fraîche	ssfrai	réel	m
Periode de mesure de la neige fraîche	perssfrai	réel	1/10 heure
Précipitations dans les N dernières heures	rrN	réel	mm
Phénomène spécial	phenspeN	réel	<a href="#">code</a> (3778)
Nébulosité cche nuageuse N	nnuageN	int	octa
Type nuage N	ctypeN	int	<a href="#">code</a> (0500)
Hauteur de base N	hnuageN	int	m

car : caractère ASCII, int : nombre entier, réel : nombre réel (avec décimale).

Les nombres entre parenthèses après le mot "code" sont les numéros de table de code de l'OMM (Organisation Mondiale de la Météorologie).

Vous pouvez cliquer sur le mot code pour accéder directement à ces tables sur le site de l'OMM

Figure 11. Description des 59 colonnes du fichier

## 9.3. SYNOP

Un exemple de variable (température maximale) dont la définition varie selon la région de mesure.  
La variation étant sur la durée de la période de mesure.

*manuel code WMO international :*

### 12.4.4 Groups (1snTxTxTx), (2snTnTnTn)

The period of time covered by the maximum and the minimum temperature and the synoptic hour at which these temperatures are reported shall be determined by regional decision.

### 12.4.5 Group (3Ejjj)

*version régionale*

306\_II-2011-2017\_fr.pdf

### 6/12.4 Groupe (1snTxTxTx)

Ce groupe est inclus dans les messages d'observation de 1800 UTC et peut être inclus dans ceux de 0600 UTC pour indiquer la température maximale pour la période de 12 heures qui précède.

### 6/12.5 Groupe (2snTnTnTn)

Ce groupe est inclus dans les messages d'observation de 0600 UTC et peut être inclus dans ceux de 180

Ici, la température maximale relevée en France le sera sur une période de 12 h.

## 9.4. Correspondances numer\_sta et communes des stations de mesure

Table 5. Correspondance entre codes et valeurs de la colonne numer\_sta

ID	Nom	Latitude	Longitude	Altitude
07005	ABBEVILLE	50.136000	1.834000	69
07015	LILLE-LESQUIN	50.570000	3.097500	47
07020	PTE DE LA HAGUE	49.725167	-1.939833	6
07027	CAEN-CARPIQUET	49.180000	-0.456167	67
07037	ROUEN-BOOS	49.383000	1.181667	151
07072	REIMS-PRUNAY	49.209667	4.155333	95
07110	BREST-GUIPAVAS	48.444167	-4.412000	94
07117	PLOUMANAC'H	48.825833	-3.473167	55
07130	RENNES-ST JACQUES	48.068833	-1.734000	36
07139	ALENCON	48.445500	0.110167	143
07149	ORLY	48.716833	2.384333	89
07168	TROYES-BARBEREY	48.324667	4.020000	112
07181	NANCY-OCHEY	48.581000	5.959833	336
07190	STRASBOURG-ENTZHEIM	48.549500	7.640333	150
07207	BELLE ILE-LE TALUT	47.294333	-3.218333	34
07222	NANTES-BOUGUENAIS	47.150000	-1.608833	26
07240	TOURS	47.444500	0.727333	108
07255	BOURGES	47.059167	2.359833	161
07280	DIJON-LONGVIC	47.267833	5.088333	219
07299	BALE-MULHOUSE	47.614333	7.510000	263
07314	PTE DE CHASSIRON	46.046833	-1.411500	11
07335	POITIERS-BIARD	46.593833	0.314333	123
07434	LIMOGES-BELLEGARDE	45.861167	1.175000	402
07460	CLERMONT-FD	45.786833	3.149333	331
07471	LE PUY-LOUDES	45.074500	3.764000	833
07481	LYON-ST EXUPERY	45.726500	5.077833	235

ID	Nom	Latitude	Longitude	Altitude
07510	BORDEAUX-MERIGNAC	44.830667	-0.691333	47
07535	GOURDON	44.745000	1.396667	260
07558	MILLAU	44.118500	3.019500	712
07577	MONTELIMAR	44.581167	4.733000	73
07591	EMBRUN	44.565667	6.502333	871
07607	MONT-DE-MARSAN	43.909833	-0.500167	59
07621	TARBES-OSSUN	43.188000	0.000000	360
07627	ST GIROS	43.005333	1.106833	414
07630	TOULOUSE-BLAGNAC	43.621000	1.378833	151
07643	MONTPELLIER	43.577000	3.963167	2
07650	MARIGNANE	43.437667	5.216000	9
07661	CAP CEPET	43.079333	5.940833	115
07690	NICE	43.648833	7.209000	2
07747	PERPIGNAN	42.737167	2.872833	42
07761	AJACCIO	41.918000	8.792667	5
07790	BASTIA	42.540667	9.485167	10
61968	GLORIEUSES	-11.582667	47.289667	3
61970	JUAN DE NOVA	-17.054667	42.712000	9
61972	EUROPA	-22.344167	40.340667	6
61976	TROMELIN	-15.887667	54.520667	7
61980	GILLOT-AEROPORT	-20.892500	55.528667	8
61996	NOUVELLE AMSTERDAM	-37.795167	77.569167	27
61997	CROZET	-46.432500	51.856667	146
61998	KERGUELEN	-49.352333	70.243333	29
67005	PAMANDZI	-12.805500	45.282833	7
71805	ST-PIERRE	46.766333	-56.179167	21
78890	LA DESIRADE METEO	16.335000	-61.004000	27
78894	ST-BARTHELEMY METEO	17.901500	-62.852167	44
78897	LE RAIZET AERO	16.264000	-61.516333	11
78922	TRINITE-CARAVEL	14.774500	-60.875333	26
78925	LAMENTIN-AERO	14.595333	-60.995667	3

ID	Nom	Latitude	Longitude	Altitude
81401	SAINT LAURENT	5.485500	-54.031667	5
81405	CAYENNE-MATOURY	4.822333	-52.365333	4
81408	SAINT GEORGES	3.890667	-51.804667	6
81415	MARIPASOULA	3.640167	-54.028333	106
89642	DUMONT D'URVILLE	-66.663167	140.001000	43

## 9.5. Script de récupération des données Météofrance

```
import requests
import time
import gzip
import shutil

#
https://donneespubliques.meteofrance.fr/donnees_libres/Txt/Synop/Archive/synop.201801.
csv.gz
URL_base =
"https://donneespubliques.meteofrance.fr/donnees_libres/Txt/Synop/Archive/synop."
URL_suffixe = ".csv.gz"
OUTPUT_DIR="extractions"
OUTPUT_FILENAME_PREFIX="meteo-"
OUTPUT_FILENAME_EXTENSION=".csv"

for year in range(1996, 2018):
    for month in [%.2d % i for i in range(1, 13)]:
        URL_fichier = URL_base + str(year) + str(month) + URL_suffixe
        rep = requests.get(URL_fichier)
        status = rep.status_code
        print("URL <" + rep.url + "> code retour <" + str(status) + ">\n")
        if (status == 200):
            downloaded_filename = OUTPUT_DIR + "/" + OUTPUT_FILENAME_PREFIX + str(year)
            + "_" + str(month) + OUTPUT_FILENAME_EXTENSION
            #     downloaded_filename_gz = downloaded_filename + ".csv.gz"
            with open(downloaded_filename, "wb") as outfile:
                outfile.write(rep.content)
            # with gzip.open(downloaded_filename_gz, 'rb') as f_in,
            open(downloaded_filename, 'wb') as f_out:
                #     shutil.copyfileobj(f_in, f_out)
            # http://www.meteofrance.com/robots.txt
            time.sleep(1)
#TODO multithread & temporisation
#TODO temps passe
```

## 9.6. Liste des stations SYNOP de France métropolitaine concernées par des catastrophes naturelles

'MONTELIMAR', 'TOULOUSE-BLAGNAC', 'NICE', 'ALENCON',  
'MARIGNANE', 'ABBEVILLE', 'LILLE-LESQUIN', 'CAEN-CARPIQUET',  
'RENNES-ST JACQUES', 'STRASBOURG-ENTZHEIM', 'ORLY', 'MILLAU',  
'DIJON-LONGVIC', 'CAP CEPET', 'ST GIRONS', "PLOUMANAC'H",  
'BREST-GUIPAVAS', 'POITIERS-BIARD', 'LE PUY-LOUDES', 'MONTPELLIER',  
'BALE-MULHOUSE', 'REIMS-PRUNAY', 'GOURDON', 'MONT-DE-MARSAN',  
'NANTES-BOUGUENAIS', 'PTE DE CHASSIRON', 'TOURS',  
'PTE DE LA HAGUE', 'BOURGES', 'TARBES-OSSUN', 'BORDEAUX-MERIGNAC',  
'BASTIA', 'LIMOGES-BELLEGARDE', 'CLERMONT-FD', 'LYON-ST EXUPERY',  
'PERPIGNAN', 'NANCY-OCHEY', 'ROUEN-BOOS', 'TROYES-BARBEREY',  
'EMBRUN', 'AJACCIO', 'BELLE ILE-LE TALUT'

## 9.7. Types des catastrophes et effectifs

type catastrophe	effectif
inondations	106911
coulees de boue	103612
mouvements de terrain	28512
mouvements de terrain differentiels consecutifs a la secheresse et a la rehydratation des sols	15747
tempete	15473
chocs mecaniques lies a l'action des vagues	6624
mouvements de terrain consecutifs a la secheresse	4852
glissement	2269
glissement de terrain	2100
poids de la neige - chutes de neige	1275
effets exceptionnels dus aux precipitations	987
inondations par remontees de nappe phreatique	956
seisme	629
effondrement de terrain	200
affaissement de terrain	167
eboulement	146
avalanche	103
eboulements rocheux	77
effondrements / eboulements	26
tornade	20

<b>type catastrophe</b>	<b>effectif</b>
grele	20
coulee de boue	9
eboulement de falaise	8
eboulement de terrain	7
affondrement / eboulement de coteaux	6
tassemement de terrain	5
crues torrentielles	5
affaissement de falaise	3
chutes de rochers / de blocs rocheux	3
raz-de-maree	3
lave torrentielle	2
inondations par remontee de la nappe phreatique	2
eboulement ou effondrement de carriere	2
effondrement de falaise	1

## 9.8. Types des catastrophes, effectifs et étude de la durée

Table 6. Statistique sur la durée par ordre décroissant d'effectif

Catastr ophe	Effectif	Min.	1st Qu.	Median	Mean	3rd Qu.	90.	95.	99.	Max.
inondat ions	106911	1	2.0	4.0	5.85	5.0	15.0	24.0	31.0	1888
coulee de boue	103621	1	2.0	4.0	5.92	5.0	15.0	24.0	31.0	1888
mouve ments de terrain	28512	1	5.0	5.0	6.36	5.0	5.0	5.0	11.0	3683
mouve ments de terrain differe ntiels consecu tifs a la sechere sse et a la rehydr atation des sols	15747	2	92.0	92.0	500.27	456.0	1826.0	2373.0	3136.0	4029
tempet e	15473	1	5.0	5.0	4.39	5.0	5.0	5.0	5.0	5
chocs mecan iques lies a l'action des vagues	6624	1	4.0	4.0	4.04	5.0	5.0	5.0	5.0	47
mouve ments de terrain consecu tifs a la sechere sse	4852	1	457.0	610.0	730.14	975.0	1341.0	1584.0	1706.0	2191
glissem ent	2269	1	5.0	5.0	6.25	5.0	5.0	5.0	39.0	366

Catastr ophe	Effectif	Min.	1st Qu.	Median	Mean	3rd Qu.	90.	95.	99.	Max.
glissement de terrain	2100	1	2.0	4.0	19.25	30.0	31.0	31.0	151.0	2435
poids de la neige - chutes de neige	1275	2	3.0	3.0	2.82	3.0	3.0	4.0	4.0	6
effets exceptionnels dus aux precipitations	987	4	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4
inondations par remontee de nappe phreatique	956	1	17.0	56.0	78.41	121.0	152.0	182.0	419.0	510
seisme	629	1	1.0	1.0	3.19	9.0	9.0	9.0	9.0	9
effondrement de terrain	200	1	1.0	31.0	108.905	214.0	214.0	214.0	274.0	1096
affaissement de terrain	167	1	1.0	3.0	47.58	29.0	77.0	120.0	1236.0	2192
eboulement	146	1	1.0	3.0	24.48	28.0	62.0	92.0	365.0	366
avalanche	103	1	1.0	1.0	3.48	4.0	7.0	9.0	22.0	59
eboulements rocheux	77	1	1.0	1.0	17.51	1.0	6.0	17.0	1113.0	1113
effondrements / eboulements	26	1	1.0	1.0	97.15	1.0	2.0	3.0	2497.0	2497

Catastr ophe	Effectif	Min.	1st Qu.	Median	Mean	3rd Qu.	90.	95.	99.	Max.
tornade	20	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1
grele	20	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1
eboule ment de falaise	8	1	1.0	2.0	12.37	2.0	59.0	59.0	59.0	59
eboule ment de terrain	7	1	1.0	1.0	2.42	2.0	10.0	10.0	10.0	10
affondre ment / eboule ment de coteaux	6	1	1.0	1.0	3.66	1.0	17.0	17.0	17.0	17
crues torrentielles	5	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1
tassemem nt de terrain	5	1	1.0	123.0	283.4	488.0	804.0	804.0	804.0	804
raz-de- marez	3	2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2
chutes de rochers / de blocs rocheux	3	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1
affaissement de falaise	3	1	1.0	1.0	2.0	4.0	4.0	4.0	4.0	4
inondat ions par remont ee de la nappe phreati que	2	28	28.0	28.0	28.0	28.0	28.0	28.0	28.0	28

Catastr ophe	Effectif	Min.	1st Qu.	Median	Mean	3rd Qu.	90.	95.	99.	Max.
lave torrentielle*	2	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1
eboulement ou effondrement de carrier e	2	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1
effondrement de falaise	1	2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2

\*Lave torrentielle : voir définition

## 9.9. "Notebooks" pour l'analyse de donnée

*Notebook pour l'analyse du dataset agrégeant les données météo et catastrophes*

- cf document joint
- [lien github](#)

*Notebook pour faire le choix d'un modèle "optimal"*

- cf document joint
- [lien github](#)

## 9.10. Pourcentage de valeurs manquantes dans les données météo

mnémonique de la variable	pourcentage de valeur manquante
numer_sta	0.00
date	0.00
pmer	5.36
tend	0.49
cod_tend	0.48
dd	0.23
ff	0.22
t	0.09
td	0.21
u	0.22
vv	16.71
ww	17.23
w1	31.71
w2	31.98
n	24.58
nbas	37.19
hbas	33.31
cl	37.14
cm	37.42
ch	37.48
pres	0.41
niv_bar	95.98
geop	95.88
tend24	97.84
tn12	75.15
tn24	100.00
tx12	75.16
tx24	100.00
tminsol	88.06
sw	100.00
tw	100.00
raf10	96.72
rafper	42.08

per	42.07
etat_sol	88.12
ht_neige	85.26
ssfrai	79.40
perssfrai	79.52
rr1	44.61
rr3	4.15
rr6	73.06
rr12	73.08
rr24	86.98
phenspe1	32.66
phenspe2	41.13
phenspe3	73.83
phenspe4	99.22
nnuage1	43.89
ctype1	44.48
hnuage1	43.92
nnuage2	71.16
ctype2	71.47
hnuage2	71.16
nnuage3	93.41
ctype3	93.55
hnuage3	93.41
nnuage4	99.82
ctype4	99.80
hnuage4	99.82

## 9.11. InfluxDB

### 9.11.1. base de tests

*Recuperation de la dernière version d'InfluxDB*

```
root@M40474:~# docker pull influxdb
Using default tag: latest
latest: Pulling from library/influxdb
55cbf04beb70: Pull complete
1607093a898c: Pull complete
9a8ea045c926: Pull complete
4c8b66fe6495: Pull complete
40585d6fd267: Pull complete
1d919b518544: Pull complete
9da0d73e805a: Pull complete
32d08014d804: Pull complete
Digest: sha256:b0f774d3162e36894d9fd7f39f7b1c4d6b91319d156f6e04b97ea3bed0b39c1f
Status: Downloaded newer image for influxdb:latest
```

*Création du container (1.6.0)*

```
sudo docker run -d -p 8086:8086 \
-v /home/user/Bureau/CNAM/git/CNAM-
projets/NFE204/etc/influxdb.conf:/etc/influxdb/influxdb.conf:ro \
-v /var/opt/influxdb:/var/lib/influxdb \
-e INFLUXDB_REPORTING_DISABLED=true \
--name fdu-uasd03-influxdb \
influxdb -config /etc/influxdb/influxdb.conf
```

*CLI*

```
sudo docker run --rm --name influx --link=fdu-uasd03-influxdb -it influxdb influx
-precision rfc3339 -host `sudo docker inspect -f '{{.NetworkSettings.IPAddress}}' fdu-
uasd03-influxdb`
```

## Enumération des colonnes

```
> SHOW FIELD KEYS
name: synop
fieldKey fieldType
-----
dd      float
ff      float
hbas    float
pmer    float
pres    float
rafper   float
t       float
td      float
tend   float
u       float
vv      float
```

## Import d'un sous-ensemble des colonnes (13) sur l'intégralité des individus

```
21:57:11-user@M40474:~/Bureau/CNAM/git/CNAM-projets/NFE204(master)$ for i in
donnees/extractions/*.csv;do python3 ../../csv-to-influxdb/csv-to-influxdb.py -d;' -s
'172.17.0.2:8086' -m 'synop' -tc 'date' -tf '%Y%m%d%H%M%S' --fieldcolumns
pmer,tend,dd,ff,td,u,vv,hbas,tnN,txN,rafper,pres,t --tagcolumns numer_sta --dbname
meteo --input $i;done
Read 5000 lines
Inserting 5000 datapoints...
Wrote 5000, response: True
Read 10000 lines
...
```

## Récupération de quelques lignes

```
> SELECT * FROM synop LIMIT 5
name: synop
time              dd  ff  hbas numer_sta pmer   pres  rafper t      td      tend u
vv

1996-01-01T00:00:00Z 170 2      07005      100030 99090      276.04 275.95 100 99
1996-01-01T00:00:00Z 100 2     2250 89642      98820 98290      271.15 261.65 10 44
40000
1996-01-01T00:00:00Z 0   0      07015      100020 99380      275.95 275.35 100 96
300
1996-01-01T00:00:00Z 120 4.1 250  07027      99940 99130      279.04 279.04 0   100
1200
1996-01-01T00:00:00Z 110 1     70   07130      99900 99380      281.25 280.65 -10 96
800
```

## Comptage des lignes

```
> SELECT COUNT(*) FROM synop
name: synop
time           count_dd count_ff count_hbas count_pmer count_pres count_rafper
count_t count_td count_tend count_u count_vv
-----
-----
1970-01-01T00:00:00Z 3323727 3324681 1998697 3252545 2891372 1776188
3380581 3366987 2840858 3365351 2479722
```

## 9.12. Telegraf

A titre d'exemple, le fichier de configuration ci-dessous permet d'injecter dans la base *ua-telegraf\_rle* la collecte des données *cpu* de la machine courante.

```
[[outputs.influxdb]]
  urls = ["http://influxdb:8086"]
  database = "ua-telegraf-rle"
[[inputs.cpu]]
  ## Whether to report per-cpu stats or not
  percpu = true
  ## Whether to report total system cpu stats or not
  totalcpu = true
  ## If true, collect raw CPU time metrics.
  collect_cpu_time = false
```

# Références

## *Données*

- le site de l'organisation mondiale de la météorologie : <https://public.wmo.int/fr> et son extranet : [https://www.wmo.int/pages/index\\_fr.html](https://www.wmo.int/pages/index_fr.html)
- Source des données météoFrance : [https://donneespubliques.meteofrance.fr/?fond=produit&id\\_produit=90&id\\_rubrique=32](https://donneespubliques.meteofrance.fr/?fond=produit&id_produit=90&id_rubrique=32)
- Source des données d'arrêté de catastrophes naturelles sur la plateforme ouverte des données publiques françaises:  
[https://www.data.gouv.fr/fr/datasets/arretes-de-catastrophe-naturelle-en-france-metropolitaine-2/#\\_](https://www.data.gouv.fr/fr/datasets/arretes-de-catastrophe-naturelle-en-france-metropolitaine-2/#_)  
( le lien sur le site des catastrophe cite une source dont le lien est mort !)