RAPPORT DU PROJET « DÉFI-IA 2022 » ÉLABORÉ PAR :

Thomas FRAMERY

Yoann MAAREK

Dorian VOYDIE

Dans le cadre du Mastère Spécialisé VALDOM

Table des matières

[I. Introduction 3](#_Toc92470989)

[II. Analyse exploratoire des données 3](#_Toc92470990)

[Les jeux de données 3](#_Toc92470991)

[Statistiques du jeu X\_train 3](#_Toc92470992)

[Statistiques du jeu Y\_train (target feature = « Ground\_truth ») 4](#_Toc92470993)

[III. Transformation des datasets 4](#_Toc92470994)

[Preprocessing 4](#_Toc92470995)

[IV. Modélisation 6](#_Toc92470996)

[V. Introduction des datasets AROME et ARPEGE (site MeteoNet) 6](#_Toc92470997)

[VI. Le Deep Learning 8](#_Toc92470998)

[Modèle 8](#_Toc92470999)

[Résultats 9](#_Toc92471000)

[VII. Problèmes rencontrés 10](#_Toc92471001)

[NaNs : 10](#_Toc92471002)

[Passer en dessous 29.48242 de MAPE 10](#_Toc92471003)

[Soumission Kaggle : 10](#_Toc92471004)

# Introduction

Dans le cadre de l’UE « IA Frameworks » du master VALDOM, nous avons pu travailler sur un projet en Data Science en collaboration avec METEO FRANCE. En effet, grâce aux données fournies par l’organisme météorologique, notre objectif était de prédire la quantité cumulée de pluie tombée sur une certaine station au sol. Par conséquent, nous avons élaboré une stratégie de conception pour obtenir la meilleure précision dans nos prédictions. Vous pouvez trouver les données sur le site MeteoNet : <https://meteonet.umr-cnrm.fr/>. L’évaluation de cette compétition était effectuée à travers la plateforme Kaggle : <https://www.kaggle.com/c/defi-ia-2022>

# Analyse exploratoire des données

## Les jeux de données

Dans cette première partie, nous allons analyser le contenu des données (la forme, les dimensions, la cardinalité, …) proposées par Météo France. Cette première phase est cruciale car elle permet de prendre en main les données et de savoir véritablement ce que l’on manipule.

Initialement, METEO FRANCE nous a fourni 2 datasets d’entrainement à travers la plateforme Kaggle :

* X\_station\_train (= X\_train)
* Y\_train

*Figure 1 : Les 5 premières lignes du dataset X\_station\_train*

Une image contenant table

Description générée automatiquement

Ici, ce qu’il a été important de remarquer, ce sont les informations contenues au niveau de l’identifiant du ligne. Par exemple, pour la première ligne, l’identifiant est :

Numéro de station 14066001\_0\_0 Heure

Jour

Ainsi, une ligne peut se lire comme suit : « Pour cette station, ce jour, cette heure, nous avons ces différents paramètres météorologiques ».

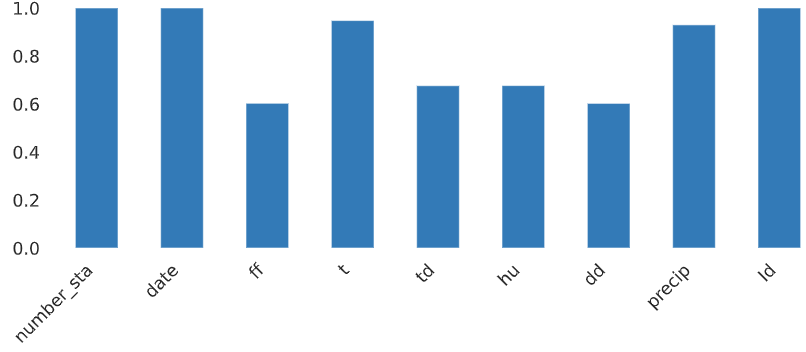
*Figure 2 : Les 5 premières lignes du dataset Y\_train*

|  |  |
| --- | --- |
| *Une image contenant table  Description générée automatiquement* | Ici, on voit que contrairement à précédemment, l’identifiant contient le numéro de station ainsi que le jour.  Par conséquent, il va falloir effectuer un travail de preprocessing sur X\_train et plus précisément au niveau de l’identifiant pour qu’il contienne seulement le numéro de station et le jour. |

## Statistiques du jeu X\_train

|  |  |
| --- | --- |
| Number of Variables | 9 |
| Number of rows | 4409474 |
| Missing cells (%) | 17.4% |
| Variable types | Numerical : 7  Categorical : 2 (date & number\_sta) |

Missing Values (in **blue** percentage of not NaN values) :



## Statistiques du jeu Y\_train (target feature = « Ground\_truth »)

|  |  |  |
| --- | --- | --- |
| Number of Variables | 4 |  |
| Number of rows | 183747 |
| Missing cells (%) | 2.9% |
| Variable types | Numerical : 2  Categorical : 2 (Id & number\_sta) |

# Transformation des datasets

Tout d’abord, cette transformation se subdivise en plusieurs étapes :

1. Écrire les fonctions de preprocessing
2. Appliquer ces fonctions aux datasets associés
3. Merge les 2 datasets résultants pour construire le trainset
4. Introduire les données AROME et ARPEGE (fournies par MeteoNet)
5. Introduire les coordonnées des stations pour plus de précision

Conformément à la stratégie mentionnée ci-avant, examinons les fonctions de preprocessing.

## Preprocessing

*Figure 3 : Fonction de preprocessing concernant X\_train*

Une image contenant texte

Description générée automatiquement

Création de colonnes « jour », « mois », « heure »

Mise en ordre des colonnes et au format int

Création d’une colonne pour merge avec Y\_train

Tri des lignes chronologique

*Figure 4 : Application du preprocessing sur X\_train*

Attribut de jointure

Une image contenant table

Description générée automatiquement

Ici, l’enjeu était de transformer le dataset X\_train pour que l’identifiant soit composé du numéro de station ainsi que du jour. Ce nouvel identifiant, nommé « Id\_merge » a servi d’attribut de jointure avec le dataset Y\_train.

L’autre enjeu concernait la colonne « precip ». Nous avons décidé de la multiplier par 24 mais cette opération prend tout son sens après la jointure des datasets et plus précisément lors du calcul de la moyenne.  
Mais avant cela, visualisons la façon dont le dataset Y\_train a été mis en forme

*Figure 5 : Fonction de preprocessing concernant Y\_train*

Une image contenant texte

Description générée automatiquement

On ne garde que le label Ground\_truth, on crée l’Id\_merge pour merge avec X\_train et on enlève les lignes NaNs 🡺 perte = 3%

*Figure 6 : Application du preprocessing sur Y\_train*

Une image contenant table

Description générée automatiquement

Attribut de jointure

Ensuite, nous devons donc effectuer la jointure des 2 datasets et créer ainsi le trainset.

*Figure 7 : Création du trainset*

Jointure des datasets

Une image contenant texte

Description générée automatiquement

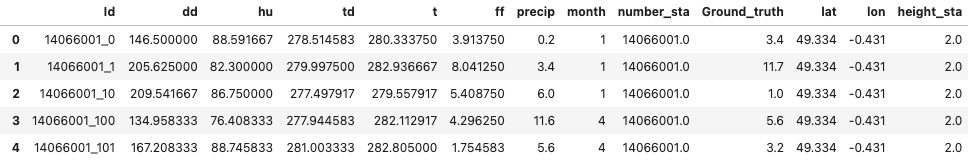
Jointure des coordonnées de stations

Moyenne des attributs de X\_train sur 24h car X\_train est décomposé en heures et Y\_train en jours.

Moyenne des features sur la journée

Somme des précipitations sur un jour

*Figure 8 : Visualisation du trainset*



Features Target Features

Si nous suivons notre stratégie mentionnée en début de partie, elle comporte l’introduction des données AROME (données supplémentaires) fournies par METEO FRANCE.

L’identifiant d’un dataset Arome étant composé du numéro de station ainsi que du jour, nous n’avons pas de modifications particulières à apporter pour l’intégrer dans notre trainset.

*Figure 9 : Imputation*

|  |  |
| --- | --- |
| Une image contenant texte  Description générée automatiquement | Nous avons essayé plusieurs imputations, d’abord dropna, puis un IterativeImputer (BayesianRidge) afin d’aller chercher les valeurs manquantes dans des lignes similaires. Le KNNImputer étant très long nous ne l’avons pas testé |

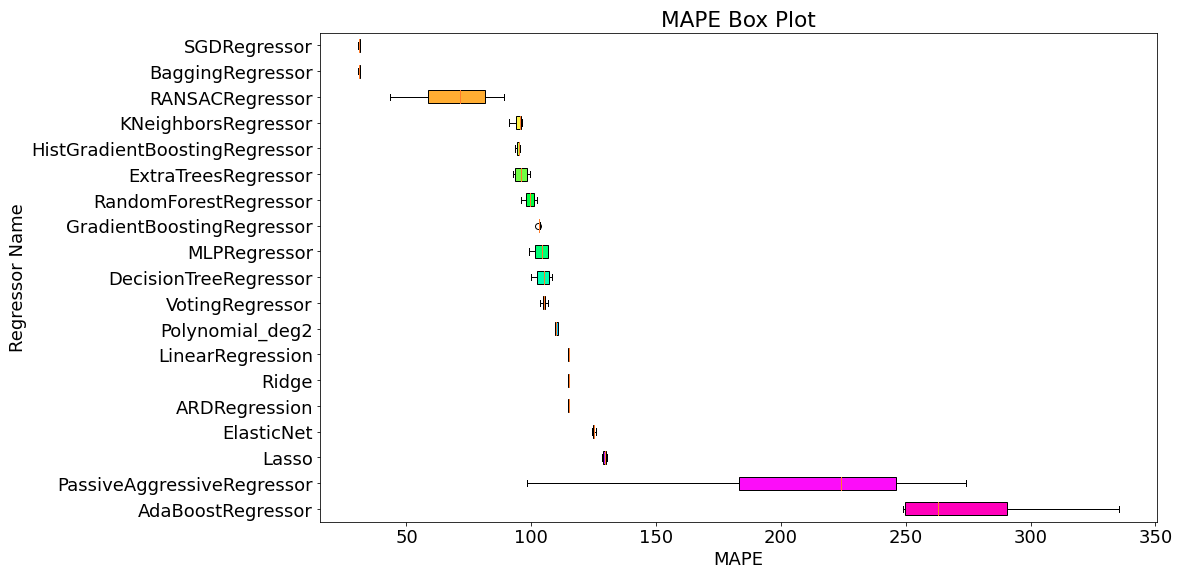
*Figure 10 : Encodage*

|  |  |
| --- | --- |
| Une image contenant texte  Description générée automatiquement | Encodage de la seule colonne qualitative : number\_sta |

# Modélisation

En l’état actuel du jeu de données, nous avons réussi à entrainer plusieurs modèles de machine learning via la bibliothèque SKLearn et voici les résultats (MAPE) :

*Figure 11 : Boxplot MAPE (sans AROME et ARPEGE) (N\_FOLDS = 10)*

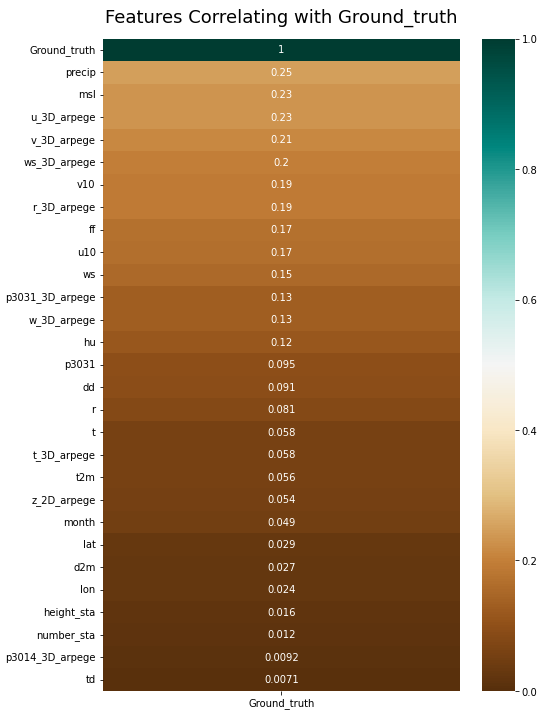


Les meilleurs résultats sont ceux du KNRegressor, RandomForestRegressor, ExtraTreesRegressor, à savoir que le SGD ainsi que le BAGGING supposés être aux alentours de 35 ne prédisent en réalité que 0 (impertinent).

# Introduction des datasets AROME et ARPEGE (site MeteoNet)

**PARTIE THOMAS**

L’introduction de ces deux datasets a joué un rôle majeur dans les performances de nos prédictions :





Une image contenant table

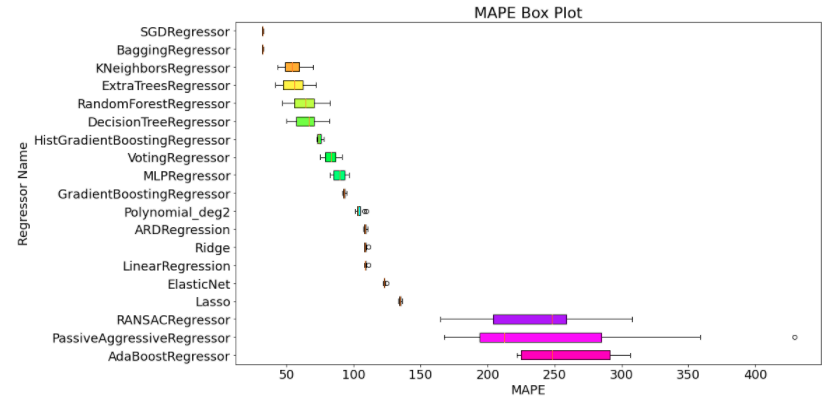
Description générée automatiquement

Une image contenant table

Description générée automatiquement

De 11 features nous sommes passés à 28. Ce qui a considérablement augmenté nos possibilités. Nous avons donc commencé par dresser une table des corrélations avec la Target. Les corrélations s’étendent de 0.0071 à 0.25, et on retrouve en majorité des éléments du jeu de données ARPEGE au sommet de la pile.

Avec les mêmes modèles de Machine Learning que sur le jeu de données initial, voici le boxplot des MAPE que nous avons retenu :



Les valeurs sont bien meilleures, nos algorithmes tournent à présent autour de 50 (contre 100 pour le dataset initial)

Nous avons ensuite effectué de la features importance selon les modèles que nous testions, en voici quelques exemples qui nous montrent la variation de la MAPE en fonction des features retenues par ordre d’importance :

|  |  |  |
| --- | --- | --- |
|  | | Distribution Prediction |
| DecisionTreeReg |  |  |
| RandomForestReg |  |  |

Globalement la MAPE se stabilise pour les 21 plus importantes features, à savoir :

['dd', 'hu', 't', 'ff', 'precip', 'month', 'ws', 'p3031', 'u10', 'v10', 't2m', 'r', 'msl', 't\_3D\_arpege', 'r\_3D\_arpege', 'ws\_3D\_arpege', 'p3031\_3D\_arpege', 'u\_3D\_arpege', 'v\_3D\_arpege', 'w\_3D\_arpege', 'z\_2D\_arpege']

Voici les prédictions que l’on peut avoir avec 3 modèles différents :

|  |  |
| --- | --- |
| RandomForestReg |  |
| SGDReg |  |
| ExtraTreesReg |  |

# Le Deep Learning

Le modèle nous ayant donné jusqu’alors les meilleurs résultats reste le réseau de neurones profond. En effet, alors que les modèles de Machine Learning classiques, même avec les hyper-paramètres optimisés peinaient à descendre en deçà de 35 de MAPE. Or grâce aux réseaux de neurone, **nous avons réussi à atteindre 25 de MAPE** pour le validation-set. Voici comment nous avons procédé :

ann = Sequential()

ann.add(BatchNormalization())

ann.add(Dense(350, activation="relu", kernel\_initializer='normal'))

ann.add(Dropout(0.3))

ann.add(Dense(512, activation="relu", kernel\_initializer='normal'))

ann.add(Dropout(0.3))

ann.add(Dense(512, activation="relu", kernel\_initializer='normal'))

ann.add(Dropout(0.3))

ann.add(Dense(128, activation="relu", kernel\_initializer='normal'))

ann.add(Dense(1))

ann.compile(loss='mean\_absolute\_percentage\_error', optimizer='adam', metrics=['mean\_absolute\_percentage\_error'])

history = ann.fit(X\_train,y\_train,epochs=2048, batch\_size=8192, verbose = 2 ,validation\_data=(X\_test,y\_test))

history\_dict = history.history

|  |  |
| --- | --- |
| **Ann train MAPE : 20.94**  **Ann val MAPE : 25.25** |  |

Un des gros avantages de ce réseau de neurones et qu’il arrive à discerner les pics de pluie parmi la masse de jours sans pluie. Voici un aperçu de 200 prédictions aléatoires effectuées sur le jeu de validation (données jamais encore vues) comparé à la labellisation :

Une image contenant texte, ciel, jour, plusieurs

Description générée automatiquement

Nous avons donc réussi à créer un modèle qui n’overfit pas nos données et qui a une performance pouvant nous propulser dans le top10 du leaderboard sur Kaggle (4 jours avant la fin sur 68 équipes) :

Une image contenant texte

Description générée automatiquement

# Problèmes rencontrés

## NaNs :

Le jeu de données comportait beaucoup de NaNs (jusqu’à 50% pour certaines features), et notamment le jeu de données CSV présent sur Kaggle (colonnes « dd », « ff », « hu », « t » en particulier). Nous avons également retrouvé des NaNs lors de nos soumissions par rapport aux Id demandés à la Baseline (pour être plus clair, on nous demandait d’effectuer des prédictions sur des lignes totalement vide : ~4.7%). Pour pallier ce problème, nous remplacions simplement les valeurs manquantes par 0 (pas de pluie le jour j sur la station S). Il y a notamment un gros manque de données (erronées ou simplement non acquises) sur les jeux ce données AROME et ARPEGE (jusqu’à 27% suivant les features). L’imputation n’a pas été évidente.

## Passer en dessous 29.48242 de MAPE

Il s’avère que si on s’amuse à soumettre une ligne de 0 (en l’occurrence de 1 car il faut ajouter +1 à notre prédiction pour le calcul de la MAPE), et bien on obtient un honnête **score de 29.48242**. En effet la distribution étant centrée très proche de 0 et les valeurs au-dessus de 2 étant peu nombreuses, on ne se trompe pas beaucoup en prédisant qu’il ne pleut jamais… En revanche, le vrai défi réside en notre capacité à réussir à identifier les pics de pluie à travers la majorité de jours sans pluie.

## Soumission Kaggle :

Nous avons rencontré un gros problème conséquent lorsque nous voulions soumettre une prédiction sur la compétition Kaggle, en effet si nous prenons l’exemple du Réseau de Neurone, **la MAPE calculée sur le validation-set était de 25.25**. Or une fois les prédictions ayant été effectuées avec le même modèle sur le test-set, la soumissions nous offrait **une MAPE autour de 40**. Après mûre réflexion, nous ne comprenons toujours pas d’où cet écart provient, éventuellement on suppose que plus le nombre de valeurs prédites est grand, plus il y a de chance que la MAPE augmente (compte tenu de la distribution de la variable target) …

# Autres pistes explorées

## La classification puis la régression personnalisée sur chaque classe

Nous avons également essayé de séparer le jeu de données en 2 classes (0 si Ground\_truth < moyenne, et 1 sinon). Nous avons testé plusieurs modèles de classification dont le meilleur (RandomForestClassifier) nous a donné une **précision de 89%**. Nous n’avons pas poursuivi cette piste (par souci de temps) mais si nous l’avions fait, nous aurions dû entrainer deux modèles de régression différents séparément sur un jeu de données séparé en deux par la classe (un jeu de classe 0 et un de classe 1), puis pour nos soumissions avoir dans notre pipeline la classification d’une observation, et selon le résultat (0 ou 1) la prédiction par le modèle entrainé sur la classe correspondante.

|  |  |
| --- | --- |
| PCA | Courbe ROC |
|  |  |