Defi-IA 2022 : prédiction du cumul de pluie pour le lendemain sur les stations au sol.

Alice LALOUE, Juan AYALA, Aldo MELLADO, Jeong Hwan KO, Nicolas PREVOT
INSA-MA-Cangrejo

January 14, 2022

1 Introduction

La prévision météorologique est une tâche très compliquée, les phénomènes météo sont très variables et dépendent fortement des conditions climatiques et atmosphériques. Les modèles physiques actuels de prévision météo de Météo-France peinent encore à prédire précisément les cumuls de pluie. C'est dans ce contexte que l'IA s'est récemment imposée comme un outil important pour la prévision météorologique. Dans le cadre du défi IA 2022, nous nous proposons d'améliorer les prédictions de cumul de pluie du modèle de Météo-France avec un modèle d'IA.

Objectif

L'objectif de ce projet est de prédire le cumul sur 24 h de pluie pour le lendemain (J) pour 267 stations données au Nord-Ouest de la France (fig. 1). Ce problème est donc un problème de régression. Nous disposons pour répondre à ce problème d'observations horaires de la veille (J-1) sur ces stations, de prévisions de plusieurs grandeurs météorologiques pour le lendemain (J), issues des modèles Arpège (2D et 3D) et Arome de Météo-France ainsi que de la prévision de cumul pour le lendemain (J) émise par le modèle de Météo-France (`baseline_forecast`) et des cumuls réellement observés le jour (J).

Figure 1: Stations au sol, pluviomètre et capteurs de station météo.

Approche

Notre approche consiste à corriger la prédiction du modèle de Météo-France pour le lendemain (J) en fonction des conditions météorologiques rencontrées avec un modèle de machine learning.

Métrique d'évaluation

Pour mesurer l'efficacité de notre modèle, nous utilisons le MAPE (*Mean Average Percentage Error*), que l'on cherche à minimiser :

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right| \tag{1}$$

avec F_t la prédiction de notre modèle et A_t le vrai cumul pour une station un jour donné de l'échantillon test.

Critère de réussite

L'objectif est considéré comme atteint si la prédiction du modèle construit obtient, sur l'échantillon de test, un score inférieur à celui du modèle de Météo-France `baseline_forecast` : 47,52.

2 Données

Dans ce projet, nous disposons d'une très grande quantité de données. Nous décidons de nous concentrer sur les observations des stations, sur quelques variables météorologiques issues des modèles Arpège 2D et Arpège 3D (isobare), ainsi que sur la prédiction de cumul du modèle `baseline_forecast` et le cumul de la veille réellement observé `baseline_obs`. Pour toutes ces données, on dispose d'un jeu d'entraînement correspondant à 2 ans (2016 et 2017) de données et d'un jeu de test.

2.1 Grandeurs physiques utilisées

Données station
Les variables observées par les stations sont données dans la table 2.1. Toutes les variables sont horaires et disponibles sur 24 h la veille (J-1) du cumul à prédire.

Table 1: Données observées par les stations.

Paramètre	Unité	Variable
vitesse du vent	m/s	ff
direction du vent	°	dd
humidité	%	hu
précipitations	?	precip
température de rosée	K	td

Arpège 2D
Les variables prédites par le modèle Arpège 2D sont quant à elles données dans la table 2. Toutes les variables sont également horaires et disponibles sur 25 h le lendemain (J). On se limite toutefois aux 24 premières heures de 00 h à 23 h comprises.

Table 2: Données du modèle Arpège 2D.

Paramètre	Unité	Variable
température à 2 m	K	t2m
direction du vent à 2 m	°	d2m
vent zonal à 10 m	m/s	u10
vent méridien à 10 m	m/s	v10
vitesse du vent	m/s	ws
humidité relative	%	r
précipitations	kg/m^2	tp
pression au niveau de la mer	Pa	msl
cumul de précipitation sur 1 h	?	tp

Arpège 3D
Pour les variables prédites par le modèle Arpège 3D, on ne sélectionne que quelques grandeurs. De plus, on sélectionne, pour chaque paramètre, un seul niveau de pression afin de limiter le nombre de variables. Les paramètres et niveaux de pression choisis sont donnés dans la table 3. Ces variables et ces niveaux d'altitude ont été choisis d'après des recommandations de Météo-France pour le concours Kaggle, et d'après leur importance dans les prévisions météo actuelles. Toutes les variables sont horaires et disponibles sur 17 h le lendemain (J).
Baseline_forecast, baseline_obs et Ground_truth
La variable `baseline_forecast` contient la prédiction de Météo-France du cumul de pluie du jour J pour chaque station par le point le plus proche. La `baseline_obs` contient le cumul de pluie mesuré le jour J-1 pour chaque station.

Table 3: Paramètres et niveaux de pression choisis dans Arpège 3D.

Paramètre	Isobare (hPa)	Variable
température potentielle	850	p3014
température	500	t
géopotentiel	500	z
humidité relative	700	r
vitesse du vent	1 000	ws
vitesse verticale	950	w

2.2 Cumul de pluie sur 24 h

Ground_truth : Cumul observé

La variable que l'on cherche à prédire, `Ground_truth`, correspond au cumul de pluie sur 24 h observé le jour J pour chaque station. La variable n'est fournie que pour le jeu d'entraînement. On trace le cumul de pluie sur 24 h moyen par mois sur les années 2016 et 2017 (cf. fig.2).

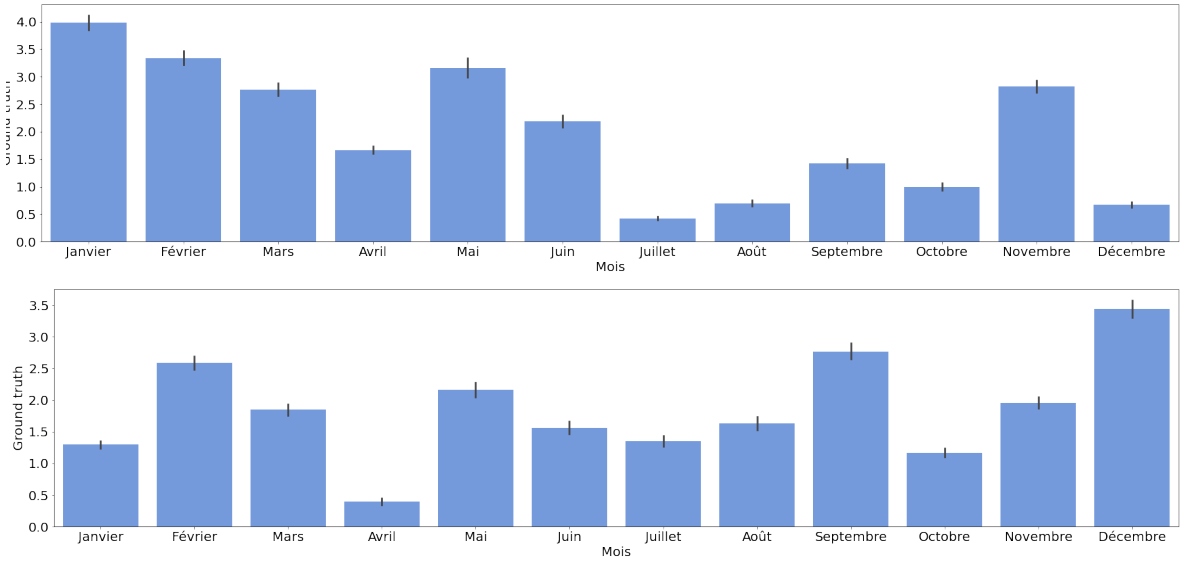

Figure 2: Cumuls moyens mensuels de pluie sur 24 h sur l'année (a) 2016 et (b) 2017.

Le cumul de pluie sur 24 h est une variable quantitative comprise en moyenne entre 0.0 et 4.0 en 2016 et entre 0.0 et 3.5 en 2017. Le cumul moyen mensuel de pluie maximal varie d'une année sur l'autre. Le cumul semble largement dépendre de la saison. En effet, le cumul est en moyenne maximisé pendant l'hiver et minimisé pendant l'été. Toutefois, on remarque que le cumul peut varier mensuellement différemment d'une année sur l'autre.

Baseline_forecast : Prédiction du modèle de Météo-France

On affiche l'erreur moyenne commise par le modèle de Météo-France sur le cumul de pluie sur 24 h par mois sur les années 2016 et 2017 (cf. fig. 3).

On observe que le modèle de Météo-France tend à surestimer le cumul de pluie sur 24 h sur l'ensemble de l'année, en dehors des mois de juillet, août et septembre. Il faut donc que notre modèle, en moyenne mensuelle, prédise des cumuls de pluie inférieurs à ceux prédits par le modèle de Météo-France en dehors de ces mois-ci.

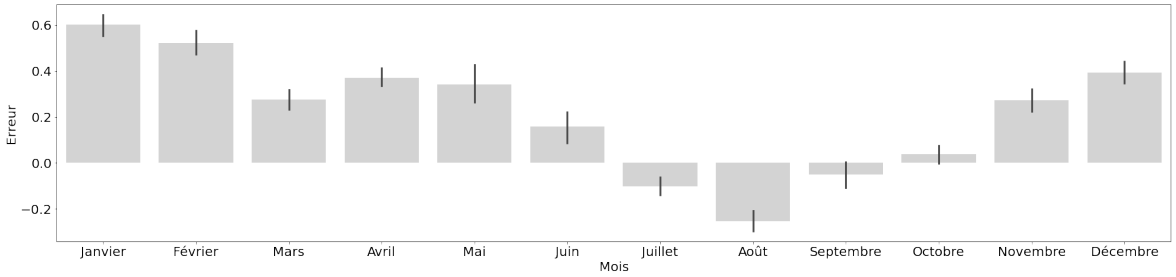

Figure 3: Erreur commise par le modèle de Météo-France sur la prédiction de cumul de pluie en fonction du mois. (`Erreur` = `baseline_forecast` - `Ground_truth`).

2.3 Corrélation

Afin de mieux comprendre ces données, nous faisons une étude de corrélation entre toutes les variables des stations et de 3D arpège, ainsi que le `Ground_truth`. Pour cela, nous utilisons la moyenne et l'écart type journalier de chaque variable pour réduire le nombre de variables.

La première étude de corrélation est fait avec un corrélogramme avec les variables des stations et les variables Arpège observées séparément. La figure 4 montre la corrélation des variables des stations. Nous observons très clairement que la seule variable qui est significativement corrélée avec `Ground_truth` est `baseline_forecast`. Cela a du sens, sachant que cette dernière est déjà censée être une très bonne approximation du cumul de pluie réel. Nous faisons la même chose avec les variables Arpège, et nous trouvons que `Ground_truth` est inversement corrélée à la pression moyenne au niveau de la mer, la vitesse verticale du vent à 950 hPa et le géopotentiel à 500 hPa.

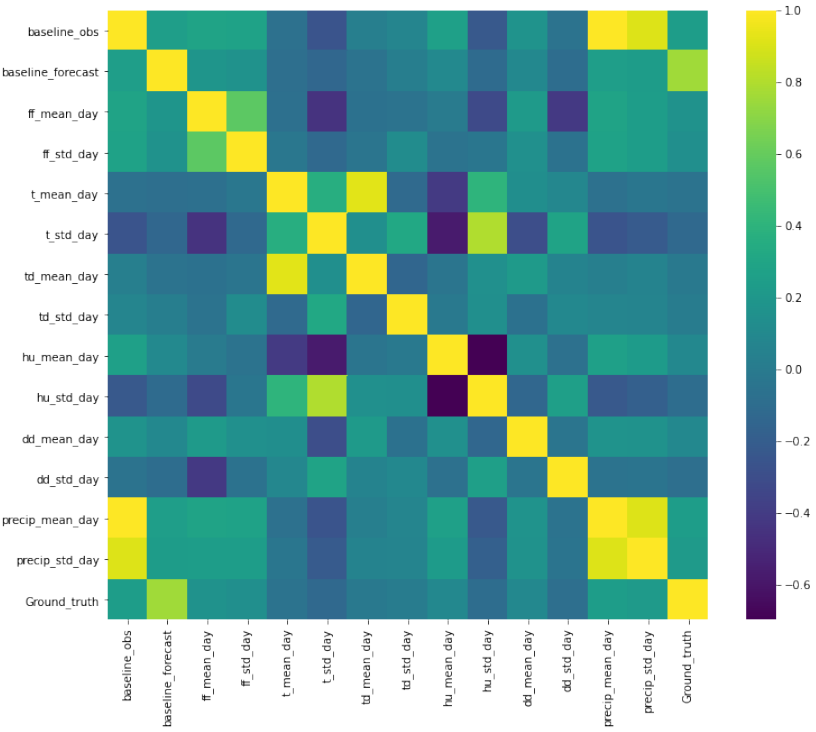

Figure 4: Corrélogramme des variables des stations

Ensuite, nous effectuons des Analyses en Composantes Principales (ACP) pour les deux groupes de variables séparément pour analyser encore une fois la corrélation de ces variables et leur importance dans la variabilité totale des données. Pour analyser ceci, nous utilisons des cercles de corrélation pour les deux premières dimensions principales. La figure 5 montre l'un des cercles correspondant aux variables des stations. Dans celui-ci nous pouvons observer que le cumul réel est plutôt corrélé aux

deux variables baseline, à la direction, et à la vitesse du vent.

Pour les variables Arpège, nous trouvons à nouveau que la variable `Ground_truth` est inversement corrélée à la pression moyenne au niveau de la mer et à la vitesse verticale du vent à 950 hPa. Également, on observe qu'elle est plus ou moins corrélée à la prédiction de précipitation totale et au vent zonal et méridional.

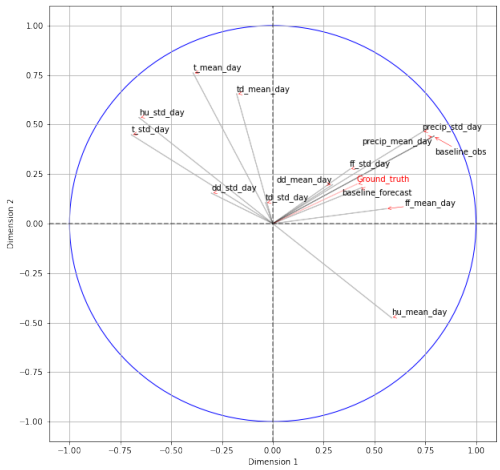

Figure 5: Cercle de corrélation (ACP) des variables des stations

Avec ces études ci-dessus nous pouvons réfléchir à quelles variables mettre en avant au moment de prédire le cumul de pluie en fonction de leur corrélation avec la variable `Ground_truth`.

2.4 Données manquantes

2.4.1 Données station

Nous observons dans la Figure 6 que les données manquantes dans les mesures des stations sont reparties aléatoirement. Cela fait que leur remplacement est relativement simple, parce qu'aux alentours des données manquantes il y aura très certainement d'autres données qui pourront nous aider à récupérer les données qui manquent. Également, le taux de données manquantes est presque le même pour les échantillons d'entraînement et de test.

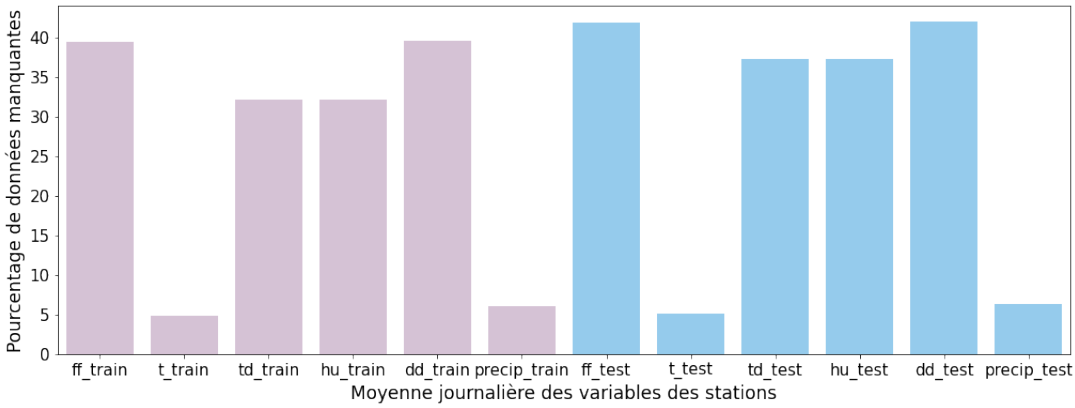

Figure 6: Pourcentage de données manquantes pour les variables station (train et test)

2.4.2 Données Arpège 2D et Arpège 3D

Dans la Figure 7 nous observons que les données manquantes sont reparties de la même manière pour les données Arpège 2D et Arpège 3D. Les données ne sont pas manquantes aléatoirement. Cela est

dû au fait qu'il y a des jours entiers où il n'y a pas de données. De même, nous voyons que les pourcentages de données manquantes sont beaucoup plus hauts pour l'échantillon test que pour celui d'entraînement. Ceci peut provoquer des problèmes au moment de tester nos modèles entraînés.

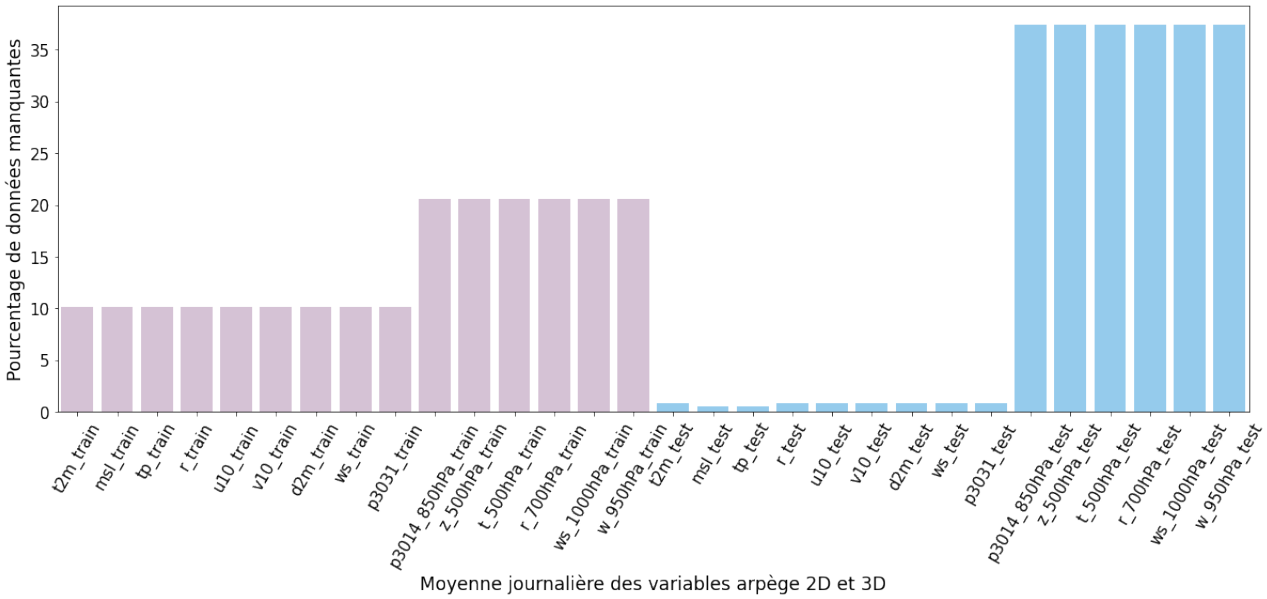

Figure 7: Pourcentage de données manquantes pour les variables Arpège 2D et 3D (train et test)

3 Pre-processing

3.1 Agrégation des données station et des données modèle

Grille 2D

Comme mentionné précédemment, notre objectif est de prédire le cumul de pluie des stations. Toutefois, les données des modèles Arpège 2D et Arpège 3D sont réparties sur une grille de latitudes et de longitudes qui ne contient pas directement les coordonnées des stations. Nous décidons alors, pour chaque station, de récupérer les variables du point de grille le plus proche (cf. fig. 10).

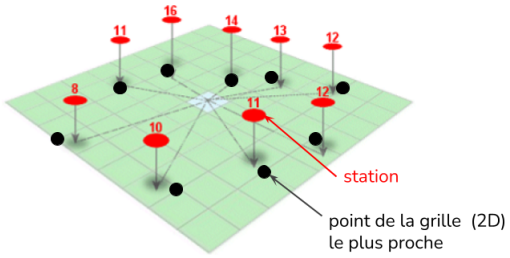

Figure 8: Récupération des données associées aux stations sur la grille 2D du modèle Arpège.

La distance entre une station de coordonnées (ϕ_s,λ_s) et un point de grille (ϕ_g, λ_g) est calculée avec la distance de Haversine :

$$d(s,g) = 2r\,arcsin\left(\sqrt{sin^2\left(\frac{\phi_s-\phi_v}{2}\right)+cos(\phi_s)cos(\phi_v)sin^2\left(\frac{\lambda_s-\lambda_v}{2}\right)}\right). \quad (2)$$

On se contente de récupérer le point de grille le plus proche pour des raisons de temps de calcul. En effet, nous disposons de 255 stations et la grille 2D est de taille 58x80.

3.2 Imputation des données manquantes

Données station : Interpolation spatiale
Les données stations sont manquantes aléatoirement, c'est-à-dire que les valeurs manquantes sont liées aux stations n'ayant pas de données pour un jour / une heure donnée, puisque ces stations peuvent parfois avoir des problèmes, notamment de défaillance des capteurs. On peut donc majoritairement les combler par interpolation spatiale. On impute les données manquantes, pour chaque station s à un temps t, par la moyenne des valeurs de ses cinq plus proches voisins géographiquement pondérée par leur distance à s:

$$\bar{X}_{s,t} = \frac{1}{5} \sum_{v=1}^{5} \frac{1}{d(s,v)} X_{v,t} \tag{3}$$

avec $\bar{X}_{s,t}$, la variable à combler de la station s au temps t, $X_{v,t}$, la variable pour la station voisine v au temps t et $d(s,v)$, la distance entre les stations s et v.

Données Arpège 2D et 3D : moyennes saisonnières
Les données des modèles Arpège 2D et 3D ne sont pas manquantes aléatoirement, des jours complets sont manquants, certains fichiers ne contiennent pas tous les horaires non plus. Ces dates représentent une grande proportion de données manquantes que l'on ne peut pas remplir par interpolation comme précédemment. On remplit donc les valeurs manquantes de chaque paramètre par ses moyennes saisonnières. Les moyennes saisonnières sont calculées pour chaque paramètre sur trois mois : décembre-janvier-février, mars-avril-mai, juin-juillet-août, septembre-octobre-novembre. Les moyennes saisonnières permettent de garder une certaine temporalité dans les données comblées.

baseline_obs, baseline_forecast
Les valeurs manquantes des variables `baseline_obs` et `baseline_forecast` sont imputées par leurs moyennes saisonnières. En effet, on a vu précédemment sur la figure 2 que le cumul de pluie sur 24 h dépend du mois et de la saison.

Ground_truth
Quant à la variable que l'on cherche à prédire, `Ground_truth`, on abandonne les lignes du jeu de données d'entraînement pour lesquelles la variable est manquante. On décide de ne pas altérer la variable à prédire.

4 Feature engineering

Moyennes et écart-types journaliers
Les données station et les donnés Arpège 2D sont des données horaires, disponibles sur 24 h. Tandis que les données Arpège 3D sont disponibles pour 17 h réparties sur 24 h. Ces trois jeux de données correspondent à un ensemble de ... variables. On remplace les paramètres horaires par leur moyenne. Cela nous permet de réduire le nombre de paramètres du jeu de données.
Toutefois, au sein d'une même journée, le même paramètre peut largement varier. La température à 2m varie entre ... et ... K pour une station la journée du 10/02/16 et entre ... et ... K pour cette dernière la journée du 10/08/16 (voir fig.9). On décide donc également de récupérer les écart-types journaliers.
Baseline_forecast et Ground_truth
Le score que nous utilisons pour évaluer nos modèles est la MAPE, calculée entre notre prédiction F et le vrai cumul A (voir éq. 1). Il est nécessaire d'après les recommandations de la compétition sur Kaggle, de rajouter +1 à la prédiction du modèle et à la vérité pour calculer ce score. Nous décidons donc de rajouter directement +1 à la variable `Ground_truth` que l'on cherche à prédire. Nos modèles sont donc directement entraînés à prédire `Ground_truth` + 1, on ne rajoutera donc pas +1 sur la prédiction du modèle.
Afin de rester consistants, on décide de transformer également la variable `baseline_forecast` en lui rajoutant +1.

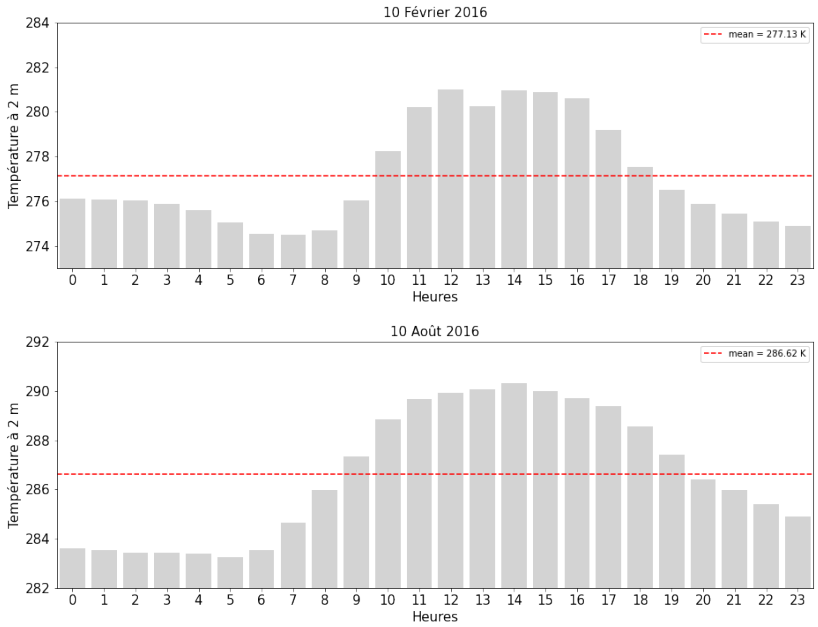

Figure 9: Température à 2 m pour une station le 10/02/2016 et le 10/08/2016. La ligne rouge représente la moyenne de la distribution.

5 Méthodes d'apprentissage

5.1 Fonctions de perte

L'un des aspects les plus importants auquel nous avons dû faire face pendant ce projet était le choix de la *loss function* (fonction de perte) pour nos modèles de machine learning. En effet, les résultats variaient beaucoup d'une fonction de perte à l'autre.

Fonction de perte du MLP

Pour le réseau de neurones nous utilisons le *mean absolute error* (MAE) comme fonction de perte. Cette fonction est très utile pour mesurer l'erreur de prédiction des données temporelles. En testant aussi le MAPE et RMSE (Root Mean Squared Error), nous trouvons les meilleurs résultats avec la fonction de perte MAE.

Fonction de perte du XGBoost

Pour ce modèle nous utilisons la fonction de perte Pseudo-Huber, que nous avons codé nous mêmes. Cette fonction est une combinaison de la perte carré et la perte en valeur absolue et elle est très robuste aux outliers dans les problèmes de régression. Contrairement à la fonction de perte de Huber, la fonction Pseudo-Huber est différentiable et même doublement différentiable.

5.2 Validation croisée

Set de validation

Afin d'ajuster les hyper paramètres de certains de nos modèles, nous avons utilisé `GridSearchCV` de `scikit-learn`. Ce module permet en effet d'obtenir un score pour chaque combinaison des paramètres, en essayant toutes les combinaisons possibles. On gardera finalement la combinaison de paramètres qui a obtenu le meilleur score.

En revanche, notre utilisation de `GridSearchCV` n'était pas optimale. En l'utilisant sur nos données, on les découpe en plusieurs sous-échantillons (par défaut cinq, dont quatre d'entraînement et un de test) aléatoirement. Or, nos données ont une dépendance temporelle, donc certaines de ces données peuvent se retrouver dans les données de test, et on perd cette dépendance.

Enfin, nous avons divisé notre échantillon d'entraînement en deux sous-échantillons : un autre sous-échantillon d'entraînement, puis un autre de validation, qui représentent respectivement 75 % et 25 % de l'échantillon original. Cette division nous permet d'améliorer nos résultats : avoir deux échantillons différents évite le sur-apprentissage des modèles, et nous pouvons ajuster nos modèles sur l'échantillon

de validation. De plus, nous pouvons faire cette division car la taille de l'échantillon original est important.

Scorer

Le module `GridSearchCV` nous donne la possibilité de créer un *scorer* selon nos besoins. Étant donné que la fonction que l'on cherche à minimiser avec notre modèle est le MAPE, nous décidons d'utiliser cette fonction comme *scorer*, mais on ajoute +1 aux A_t (cf. équation 1) car il se peut que $A_t = 0$.

6 Modèles de Machine Learning

Dans ce projet, nous nous sommes concentrés sur quatre modèles différents de machine learning : une Random Forest (forêt aléatoire), un Extreme Gradient Boosting (XGBoost), un Multi-Layer Perceptron (MLP) et enfin, une SVM avec noyau linéaire. Ces modèles se prêtent bien aux données tabulaires.

Random Forest

Nous commençons par implémenter un modèle de régression par forêt aléatoire avec la libraire `scikit-learn` de Python.
En effet, ce modèle présente l'avantage d'être facile à implémenter et d'être rapide à faire tourner. Cela nous permet d'avoir des premiers résultats rapidement.

Table 4: Paramètres de la forêt aléatoire.

Nombre d'arbres	Profondeur maximale	Nombre de variables à considérer à chaque nœud
100	`None` (pas de restriction)	`auto` p/3

Extreme Gradient Boosting

Table 5: Paramètres de XGBoost.

Nombre d'arbres	Profondeur maximale	Taux d'apprentissage
300	pas de restriction	0.1

Multi-Layer Perceptron

Notre modèle suivant était un perceptron multi-couches, donc un réseau de neurones, avec la librairie `tensorflow` de Python. Dans ce type de modèle, il y a une grande quantité d'hyper paramètres à régler : nombre de couches cachées, nombre de neurones par couche, ajout ou pas de *dropout*, etc.
Nous avons essayé beaucoup de combinaisons possibles, mais le modèle que nous avons retenu est le suivant car il a donné les meilleurs résultats :

Table 6: Paramètres du réseau de neurones.

Nombre de couches cachées	Nombre de neurones par couche	Taux d'apprentissage
15	32	0.001

La couche de sortie contient uniquement un neurone. La fonction d'activation dans toutes les couches est la fonction `ReLU` : on la garde notamment pour la couche de sortie puisqu'il s'agit d'un problème de régression et que les cumuls de pluie sont positifs ou nuls. On choisit `he_uniform` pour le paramètre `kernel_initializer` et l'optimiseur est `RMSprop` avec `learnng_rate` = 0.001.

Linear SVR

Notre dernier modèle était une SVM avec noyau linéaire : `LinearSVR` dans `scikit-learn`. Nous avons pris ce modèle car en général il s'adapte bien aux tailles de données importantes. En revanche, nous ne l'avons pas ajusté autant que les autres modèles en raison de son temps d'exécution, qui était conséquent, et il ne donnait pas autant de bons résultats.

Table 7: Paramètres de la SVR.

Paramètre de régularisation	Tolérance	Nombre maximal d'itérations
1.0	0.0001	1000

Scores

Pour avoir un premier résultat, nous testons quatre modèles sur l'échantillon de validation et nous trouvons les résultats dans le Tableau 8.

Table 8: Scores des modèles obtenus sur les échantillons de validation. Score : MAPE.

Modèle	Score en validation
Random Forest	35
XGBoost	15,8
SVR	41,2
Multi-Layer Perceptron	16,6

D'après les résultats ci-dessus nous observons que les deux modèles les plus performants sont XGBoost et MPL. Ensuite, nous testons ces deux modèles avec l'échantillon test. Il faut préciser qu'après appliquer ces modèles, il y a des valeurs prédites inférieures à 1. Dans ce cas là, nous arrondissons ces valeurs à 1 avant d'envoyer la prédiction sur Kaggle. Finalement, nous obtenons les résultats dans le Tableau 9 et nous gardons le meilleur score de 26,5 correspondant au MLP. Avec ce score nous avons bien battu le score de la baseline forecast.

Table 9: Scores des meilleurs modèles obtenus sur les échantillons de test. Score : MAPE.

Modèle	Score en test
XGBoost	28,2
Multi-Layer Perceptron	26,5

On peut remarquer que les scores des modèles diminuent entre la validation et le test. En effet, cela peut être lié soit un surajustement, soit cela peut être expliqué par la grande présence de valeurs manquantes dans l'échantillon de test en comparaison avec l'échantillon utilisé pour la validation. Une autre hypothèse est que l'utilisation de GridSearch (sélection aléatoire de données) ne permet pas de choisir des coefficients optimaux dans notre problème où nos données sont liées temporellement.

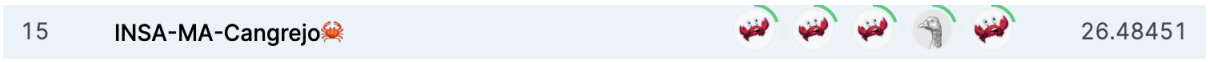

Figure 10: Classement final sur Kaggle