

DEFI IA

Prédiction de précipitations

Etudiants

*Armand Chédozeau, Yasser Hader
Noemi Maleville Collantes, Anthony Gofin
et Zoe Philippon
5GMA*

Table des matières

Introduction et Présentation du Défi IA	2
1 Pre-processing des données	2
1.1 Collecte des données de prédiction météorologique (Arpège 2D)	2
1.2 Sélection des données	3
1.3 Remplissage des valeurs manquantes	3
2 Feature engineering	4
2.1 Jeu de données initial	5
2.2 Données issues de l'ACP	5
2.3 Moyennes journalières des variables	8
3 Méthodes d'apprentissage	9
3.1 K-fold Cross Validation et Split Train/Test	9
3.2 Fonctions de perte utilisées	9
4 Modèles et résultats	10
Conclusion	11

Introduction et présentation du Défi IA

Le Défi IA 2022 a été créé par l'Ecole Nationale de la Météorologie (ENM) dans le but de confronter les méthodes plus traditionnelles de prédictions météorologiques à des méthodes issues de l'Intelligence Artificielle.

L'objectif principal de ce défi était donc la prédition de la totalité des précipitations sur 24 heures au niveau de stations de mesure. Pour ce faire, nous avions de nombreuses données de deux types :

- **Données mesurées aux stations (X_station)**

Les données mesurées au niveau des stations correspondaient à des mesures de température, vent, humidité et autre variables atmosphériques à une date t.

- **Prédictions météorologiques**

Les données de prédictions météorologiques étaient fournies par MeteoFrance et correspondaient à une prédition des variables atmosphériques et des précipitations dans le futur. De nombreuses données de prédition étaient disponible, et parmi celles-ci, nous avons choisi de n'utiliser que les données correspondant au modèle 2D basse résolution (Arpege_2D).

A l'aide de ces données, nous devions réussir à obtenir des prédictions plus précises en utilisant des méthodes d'Intelligence Artificielle. Dans ce but, nous avons dans un premier temps dû traiter les données afin de les avoir sous un format exploitable pour nos algorithmes. Ensuite, nous avons réalisé de nombreux tests, en modifiant les bases de données et en les testant sur des modèles que nous avons également optimisé dans le but d'augmenter l'efficacité de nos prédictions.

1 Pre-processing des données

Afin de pouvoir utiliser les données dans nos modèles, nous les avons mis en forme. Cela a nécessité plusieurs étapes.

1.1 Collecte des données de prédition météorologique (Arpège 2D)

Afin d'avoir l'ensemble des données pour chaque station il a fallu effectuer une correspondance entre les coordonnées des stations et la grille du modèle Arpège 2D. La donnée collectée correspond au point de grille le plus proche de la station comme montré ci-dessous.

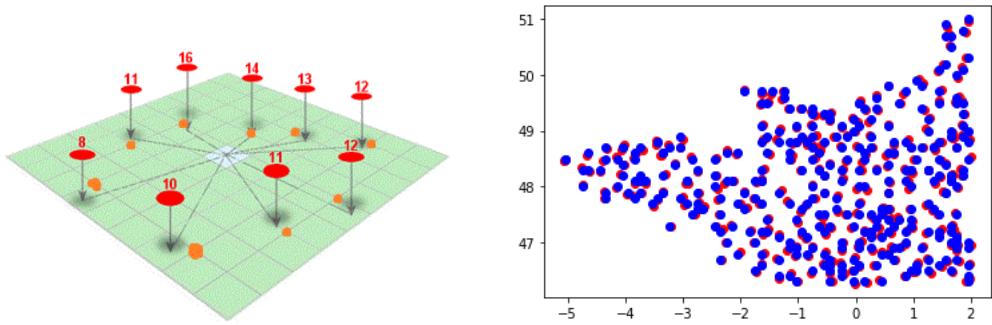


FIGURE 1 – Correspondance entre les stations (en rouge) et les points de grille (orange à gauche, bleu à droite)

1.2 Sélection des données

Dans un premier temps, nous avons constaté que les données mesurées au niveau des stations à toutes les heures ne sont pas toutes présentes. Sur certaines stations, les données devant figurer à une heure donnée sont absentes. Il y en a peu par rapport à la taille totale de notre jeu de données. Nous avons donc décidé de supprimer les données relatives à des stations qui ne possèdent pas d'observations pour les 24 heures.

Cette opération a été réalisée uniquement sur les données d'apprentissage. En effet, celles de test doivent nécessairement être utilisées. Nous avons donc remplacé ces valeurs absentes par des NaNs que nous avons ensuite rempli à l'aide des K-plus proches voisins, comme nous l'expliquerons plus tard.

De plus, dans les données de prédiction du modèle Arpege_2D, il manque l'équivalent de deux mois de prédictions. Afin de pouvoir joindre les deux jeux de données sélectionnés (prédiction et observations atmosphériques), il faut que leurs formats soient identiques. Nous avons supprimé les données d'observations atmosphériques associées aux deux mois manquant des données de prédiction.

1.3 Remplissage des valeurs manquantes

Nous n'avons pas employé les mêmes méthodes de remplissage pour toutes les données.

Pour les données d'observations atmosphériques

Pour les données X_station, nous avons choisi d'utiliser les K-plus proches voisins spatiaux de nos stations. Le fonctionnement est le suivant : nous listons les K (ici 10) voisins les plus proches, en termes de localisation, d'une station où il manque des données. Ensuite, nous complétons la valeur manquante par la moyenne pondérée des valeurs voisines en utilisant la pondération suivante :

Cette pondération permet de prendre en compte la distance à la station dont nous essayons de remplir les données manquantes lors du remplissage. Nous avons choisi $L = \frac{1}{3}^\circ$ afin d'éviter de prendre en considération des stations éloignées de plus de 1° de latitude ou longitude, ce qui correspond au

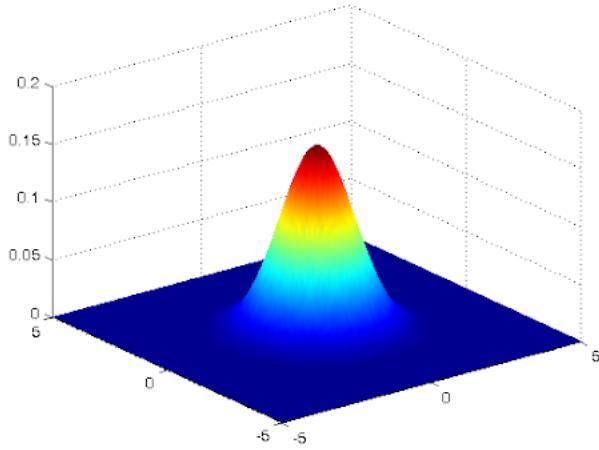


FIGURE 2 – Distribution de la pondération : $e^{\frac{\|x-y\|^2}{L^2}}$

triple de l'écart-type de la distribution de notre pondération.

Pour les données de prédictions météorologiques

Au niveau des données d'entraînement X_forecast, il n'y a pas de données manquantes suite à la sélection mentionnée précédemment. Cependant, sur les données test, il manque ce qui correspond à deux jours entiers de données. Les méthodes utilisées avant, à savoir les K-plus proches voisins ne sont pas exploitables ici. En effet, il est impossible de remplir les données en fonction des stations proches spatialement puisque ces stations elles-mêmes ne présentent aucune données. Le même problème est présent si l'on essaye de prendre les voisins proches en temporalité. Effectivement, les identifiants du jour étant ré-indexés aléatoirement, nous ne pouvons donc pas utiliser les données de la veille ou du lendemain pour remplir les données manquantes.

La solution trouvée a été de moyenner entièrement les colonnes du jeu de données afin d'obtenir une valeur approximative. Une amélioration possible serait de moyenner les informations sur les mois afin de préciser l'approximation.

2 Feature engineering

Nous avons réalisé plusieurs types de feature engineering pour pouvoir améliorer les résultats obtenus.

2.1 Jeu de données initial

Au tout début de nos essais, nous avons utilisé l'entièreté des données brutes et tenté d'utiliser ce jeu sur nos modèles. Le data set complet comprenait les variables horaires, à savoir 24 heures par jour. L'information donnée par le mois était codée par des nombres entiers allant de 1 à 12. Cependant, cette méthode ne permettait pas de constater la périodicité de l'année. Nous perdions donc toute information relative aux saisons, et au fait que les mois dits de "fin d'année" et ceux de "début d'année" avaient une temporalité proche. Pour pallier ce problème, nous avons utilisé le cosinus et le sinus du mois, afin d'obtenir une variable plus cyclique qui modélise mieux la périodicité d'une année. La formule utilisée est la suivante :

$$\cosmonth = \cos\left(\frac{2\pi(month - 1)}{12}\right) \quad (1)$$

La formule est respectivement la même pour le calcul du sinus du mois.

2.2 Données issues de l'ACP

Le nombre de variables dans notre jeu de données est de 362, dû aux 360 features et aux deux variables *cosmonth* et *sinmonth*. Nous avons donc réalisé une analyse en composantes principales pour réduire la dimension, mais aussi pour interpréter nos données. En faisant un analyse en composantes principales sur les variables moyennées sur 24 heures, nous obtenons des résultats similaires. C'est pourquoi nous présenterons cette dernière ACP, dont les graphes sont plus lisibles.

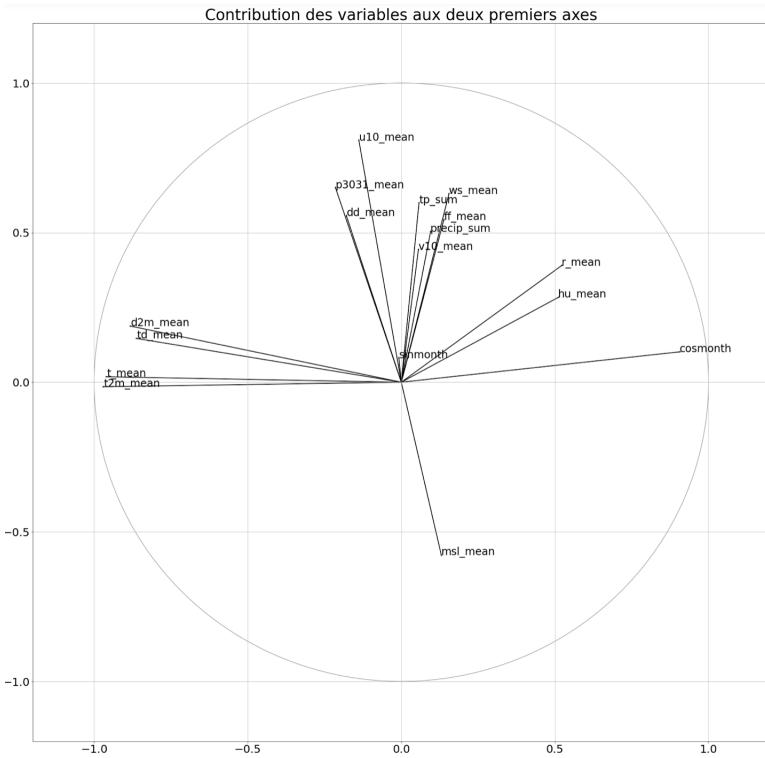


FIGURE 3 – Cercle des corrélations

La figure 2.2 montre la contribution des variables sur les deux premiers axes. La composante 1 est associée à la variables cosmonth, et négativement, aux variables $t2m_mean$, t_mean , td_mean et $d2m_mean$. La composante 1 est quant à elle associé aux variables liées à l'intensité du vent ($u10_mean$, $v10_mean$, ws_mean , dd_mean) ainsi que tp_sum , $p3031_mean$.

A l'aide de la figure 4, on observe que la composante 1 permet de capter les températures observées à chaque saison : les points bleus sont associés aux saisons plus fraîches (novembre, décembre, janvier, février, mars). Le bleu le plus foncé correspond au mois de janvier où les températures sont avoisinent les 0° . Les points rouges eux sont associés aux saisons les plus chaudes (mai, juin, juillet, août, septembre), avec un cosinus minimal pour le mois de juillet, où les températures avoisinent les 25° . L'interprétation pour les autres variables de l'axe 1 (t_mean , td_mean et $d2m_mean$) est analogue.

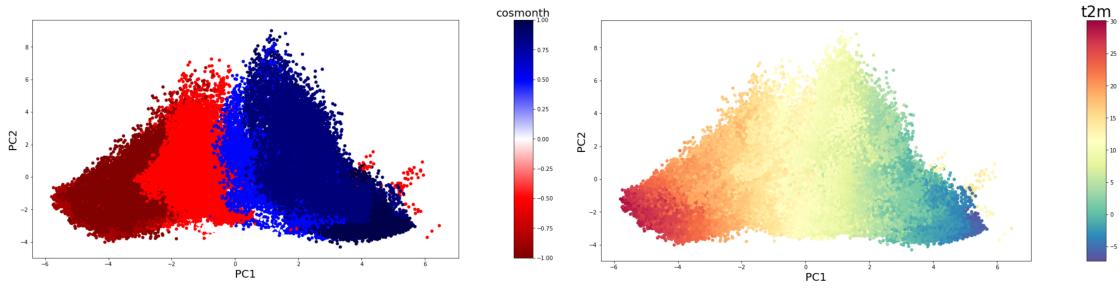


FIGURE 4 – PC2 vs PC1. Les points sont marqués en fonction des valeurs de *cosmonth* (à gauche) et de *t2m* (à droite).

L'axe 2 semble porter l'effet de la variable *Ground_truth*, comme le montre la figure 2.2. En utilisant la même méthode que pour l'axe 1, on observe avec la figure 6 que la pression au niveau de la mer (*msl_mean*) est moindre lorsqu'il pleut, ce qui correspond en fait à un phénomène de dépression. L'intensité du vent (*u10_mean*, *v10_mean*, *ws_mean*) augmente avec *Ground_truth*.

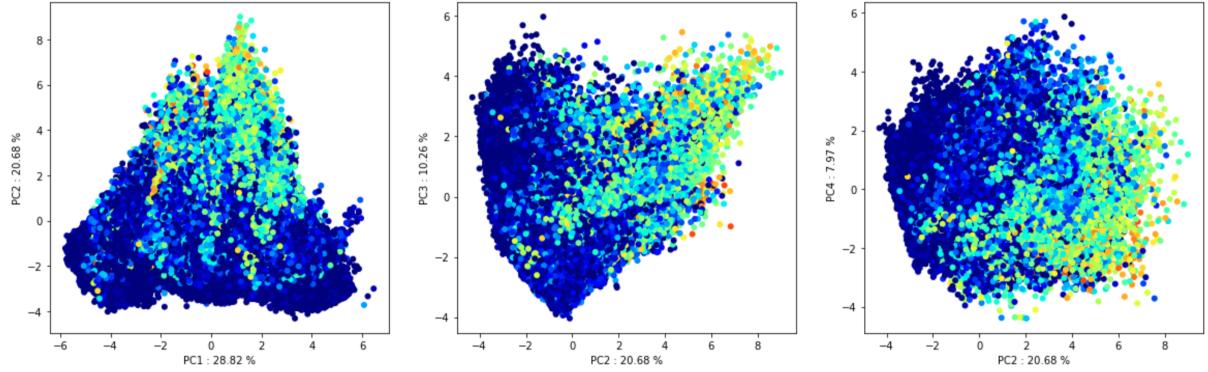


FIGURE 5 – Composante 2 face aux composantes 1, 3 puis 4.

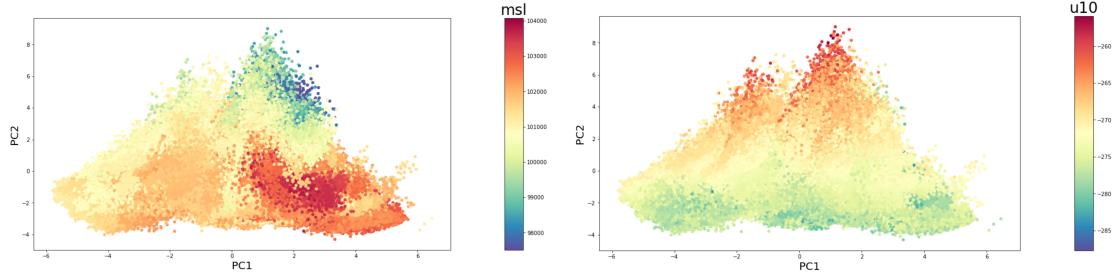


FIGURE 6 – PC2 vs PC1. Les points sont marqués en fonction des valeurs de *msl* (à gauche) et de *u10* (à droite).

2.3 Moyennes journalières des variables

Après avoir réalisé l'ACP et utilisé les données qui en étaient issues, nous avons constaté de légères améliorations dans les performances de nos modèles. Cela nous a donné l'idée de continuer à réduire les dimensions de notre jeu de données. Au lieu de prendre les données brutes qui comprenaient les variables horaires, nous avons réalisé une moyenne journalière sur les données. Ce jeu de données correspond à la dernière étape de feature engineering que nous avons mise en place, et c'est avec cette modification que nous avons obtenu de meilleures performances.

3 Méthodes d'apprentissage

Au niveau des méthodes d'apprentissage, nous sommes également passés par plusieurs étapes.

3.1 K-fold Cross Validation et Split Train/Test

Afin d'évaluer les performances de notre modèle, nous avons réalisé une validation croisée à K=5 folds. Cependant, cette validation croisée ne permettait pas de constater de variations de résultats selon les folds, et nous nous sommes donc concentrés sur un simple split Train/Test (75% / 25%) dans le but de simplifier la procédure puisque cette méthode nous paraissait suffisante.

3.2 Fonctions de perte utilisées

Au début de nos tests, les fonctions de perte utilisées étaient la MSE et la RMSE majoritairement. Puis, nous sommes passés à la fonction MAPE, qui nous a permis d'augmenter les performances de nos modèles.

Mean Squared Error (MSE)

La formule de la MSE est la suivante :

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (2)$$

Ici, Y correspond au vecteur de valeurs observées de taille n, et \hat{Y} celui des valeurs prédictes.

Root Mean Squared Error (RMSE)

On utilise les mêmes notations ici pour la formule de la RMSE :

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{n}} \quad (3)$$

Mean Absolute Percentage Error (MAPE)

La formule de la MAPE :

$$MAPE = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{(Y_i - \hat{Y}_i)}{Y_i} \right| \quad (4)$$

Afin d'utiliser cette formule sans risquer d'obtenir un dénominateur nul, il était nécessaire de rajouter une unité à toutes nos données d'observation. C'est cette fonction de perte qui nous a permis d'améliorer grandement les performances de nos modèles.

4 Modèles et résultats

Nous avons testé trois modèles : Random Forest, Gradient Boosting et MultiLayer Perceptron (MLP). Comme mentionné précédemment, nous avons commencé nos tests sur le jeu de données entier. Les performances de prédiction n'étaient pas à la hauteur de nos attentes, mais nous avons tout de même noté une légère supériorité des performances du MLP.

Nous avons donc continué à exploiter ce modèle, et nous sommes concentrés sur le feature engineering dans le but d'optimiser les performances du MLP. C'est à ce moment là que nous avons testé l'utilisation des données issues de l'ACP. A ce moment là, nous avons également changé notre fonction de perte pour utiliser la MAPE. Ces deux modifications ont causé une amélioration de nos résultats. La réduction de dimensions nous paraissait pertinente, et c'est pourquoi nous avons utilisé le jeu de données constitué des moyennes journalières.

Sur ce jeu de données, nous avons obtenu nos meilleurs résultats avec le MLP. Par la suite, nous avons utilisé ces données sur le modèle de Gradient Boosting, avec une fonction de perte MAPE. C'est là que nos performances ont été les meilleures, comme l'indique le tableau 1.

MultiLayer Perceptron	Random Forest	Gradient Boosting
28.92249	73.50365	26.57299

TABLE 1 – Résultats de nos modèles sur les données journalières moyennes

Nous remarquons que les performances du modèle de Random Forest sont bien plus médiocres que les deux autres modèles. Cela s'explique car il était impossible de changer la fonction de perte de la Random Forest. Puisque nous nous pouvions pas utiliser la MAPE dans ce cadre là, les performances étaient plus faibles.

Afin d'obtenir les résultats ci-dessus, nous avons arrondi nos prédictions à une décimale près pour satisfaire les critères imposés par la soumission au défi Kaggle. Nous avions également certaines prédictions négatives, qui correspondaient théoriquement à des précipitations négatives. Ce phénomène étant impossible dans la pratique, nous avons choisi de ramener ces valeurs négatives à 0. Nous avons supposé que l'une des raisons pour laquelle nous pouvions obtenir de telles prédictions étaient les variations climatiques. En effet, il a fait beaucoup plus chaud en 2018 (13.9° en moyenne) qu'en 2017 (13.4° en moyenne) et nous supposons que ces changements impliquent que la moyenne des prédictions de précipitations sur l'année 2018, pour être similaire aux données de 2017, a nécessité des compensations sous la forme de valeurs négatives.

Conclusion

L'objectif du Défi IA était de prédire les précipitations en utilisant des méthodes de Machine Learning pour voir si ces dernières permettaient d'augmenter l'efficacité des méthodes de prédiction traditionnelles.

Pour ce faire, nous avions accès à un grand nombre de données, qui incluaient des données d'observations météorologiques ainsi que des prédictions fournies par MétéoFrance. Après avoir sélectionné les données et rempli les valeurs manquantes à l'aide des K-plus proches voisins et de moyennes sur les données, ces dernières pouvaient alors être exploitées par nos modèles.

Si les données étaient exploitables, elles n'étaient pas pour autant optimales, et nous avons donc mis en place du feature engineering afin de voir comment nous pouvions améliorer les performances de nos modèles. Nous avons dans un premier temps utilisé les données brutes. Ensuite, nous avons testé comment nos modèles réagissaient lorsque les données d'entrée étaient les 50 premières composantes issues de l'ACP. Cela a augmenté légèrement les performances, et nous avons donc poursuivi la piste de réduction de dimension en utilisant les moyennes journalières des variables de notre jeu de données. Cette modification a été très productive et nous avons conservé ces moyennes pour nos résultats finaux.

En plus de ces altérations de jeu de données, nous avons également utilisé diverses méthodes d'apprentissage. Nous avons utilisé la validation croisée ainsi que le split train/test pour vérifier les performances de nos modèles, et modifié les fonctions de perte pour voir l'efficacité de nos modèles.

En parallèle, nous testions nos modèles et voyions quelles actions de notre part permettaient de les améliorer. Les modèles retenus étaient les suivants : Random Forest, Gradient Boosting et MultiLayer Perceptron. Le Gradient Boosting effectué avec les données journalières moyennes et une fonction de perte MAPE nous a donné les meilleurs résultats.

C'est donc ce modèle que nous avons retenu afin de prédire les précipitations de la manière la plus efficace pour ce défi Kaggle.