Défi IA
Prédire des précipitations quotidiennes sur des stations de mesures au sol

Francisco Piriz

Janvier 2022

1. Introduction

Contexte

Aujourd'hui, les prévisions météo reposent sur un gigantesque réseau de collecte de données. Ce réseau inclut des réseaux de stations de mesures au sol, des bouées, des satellites météorologiques, des radiosondages, des capteurs embarqués sur les avions, etc. Toutes ces sources combinées forment un réseau d'observation de données. Les météorologistes s'appuient sur ces données pour lancer des modèles informatiques pour faire des prévisions météorologiques.

Les modèles météorologiques ne sont que des représentations et des approximations de la réalité, leur habilité à faire des bonnes prévisions météo reste limité par la quantité de données, le temps requis pour les analyser, la puissance de calcul et la complexité des événements météorologiques. Pourtant, il est extrêmement compliqué de prévoir le temps qu'il fera demain.

Aujourd'hui, l'IA est utilisée par METEO FRANCE pour améliorer les prévisions météorologiques. Sur ce contexte, l'École nationale de la météorologie (ENM) lance ce défi qui challenge les prévisions météo utilisant des méthodes classiques (synthèse humaine de différentes sources de données) avec nos algorithmes d'IA.

Objectif du Défi IA

L'objectif principal de ce défi est de prédire des précipitations quotidiennes cumulées (sur 24h) sur des stations de mesures au sol avec des algorithmes d'IA. Pour cela deux sources de données peuvent être utilisés :

- Stations de mesures au sol (X_station) : mesures de 6 paramètres physiques par heure sur des stations météorologiques au sol (vitesse et direction du vent, température, température du point de rosée, humidité, précipitations).
- Les prévisions météo (X_forecast) réalisées par METEO FRANCE.

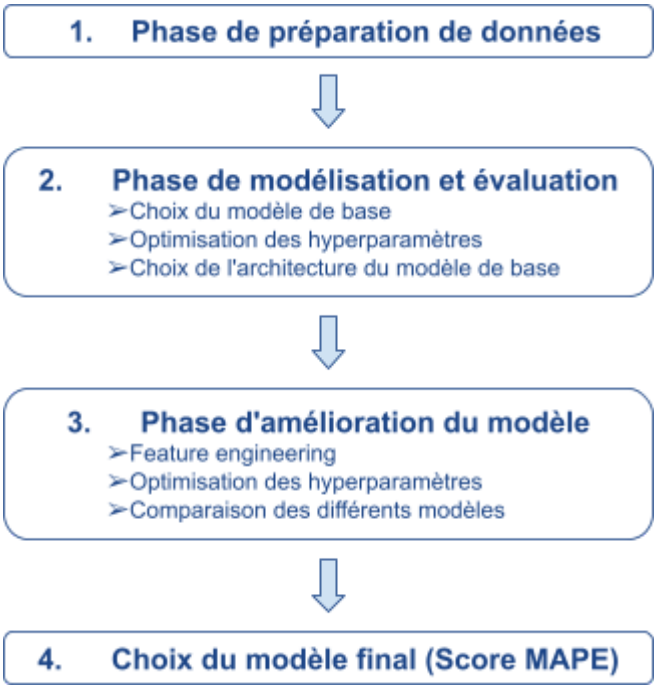
La métrique d'évaluation pour ce défi est l'erreur absolue moyenne en pourcentage (Mean Absolute Percentage Error, alias MAPE). Cet indicateur est la moyenne des écarts en valeur absolue par rapport aux valeurs observées. Le score MAPE est donnée par :

$$MAPE = \frac{100}{N} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right| \qquad (1)$$

Où chaque prévision (Ft) est comparée à la valeur réelle (At) pour le t-ième élément parmi n éléments. Pour éviter un zéro (At), chaque (At) est égal à la mesure réelle d'une station au sol +1. Pour être cohérent, on ajoute +1 a les prédictions réelles.

2. Méthodologie

Plusieurs aspects (data cleaning, feature engineering, optimisation des hyperparamètres, phase d'entraînement, etc) doivent être pris en compte pour créer un modèle performant d'IA. Pour considérer tous ces aspects, le projet sera structuré de la manière suivante :



2.1 Approche du modèle IA

Phase 1 : Préparation de données

La phase de préparation des données est fondamentale car elle permet de passer des données brutes, telles qu'extraites des sources de données, à des données utilisables par les différents algorithmes de Machine Learning. Cette phase est très chronophage car cela a représenté plus de 80% de ce projet.

Deux sources de données sont disponibles : les stations de mesures au sol (X_station) et les prévisions météo (X_Forecast) réalisées par Météo France. Pour simplifier le problème, on va d'abord créer un modèle d'IA de base n'utilisant que les stations de mesures au sol. Les prévisions météo ont été mises de côté pour une étape d'amélioration du modèle. Cette dernière étape d'incorporation des prévisions météo n'a malheureusement pas pu être atteinte dans ce travail.

Les données de mesures au sol sont constituées de 6 paramètres physiques mesurés par heure sur 267 différentes stations météorologiques au sol. Ces paramètres physiques sont la température (t), la température du point de rosée (td), l' humidité (hu), la vitesse et direction du vent (ff et dd) et la précipitation (pp). La base de données fournie par ce défi est séparée en deux : un jeu de données d'entraînement (X_station_train) et un jeu de données de test (X_station_test). Les deux jeux des données ont environ deux ans de mesures (X_station_train : 2016-1-1 → 2017-12-31 & X_station_test : 2018-01-01 → 2019-12-31), ça

veut dire environ 4 millions de mesures par set. La Figure 1 illustre quelques lignes du jeu de données d'entraînement (X_station_train).

	number_sta	date	ff	t	td	hu	dd	precip	Id
4403495	14066001	2017-12-30 19:00:00	9.63	285.38	283.40	87.7	233.0	0.2	14066001_729_19
4403496	14066001	2017-12-30 20:00:00	9.80	285.70	283.16	84.6	230.0	0.0	14066001_729_20
4403497	14066001	2017-12-30 21:00:00	10.67	286.07	282.83	80.8	230.0	0.0	14066001_729_21
4403498	14066001	2017-12-30 22:00:00	10.02	286.53	283.01	79.4	230.0	0.0	14066001_729_22
4403499	14066001	2017-12-30 23:00:00	10.67	286.84	283.22	78.9	230.0	0.0	14066001_729_23

Fig 1. Affichage du jeu de données d'entraînement.

Pour sélectionner la meilleure combinaison d'hyper-paramètres pour le modèle de prédiction de pluie, il ne faut pas évaluer le modèle sur le jeu de test (X_station_test) sinon on risque de sur-apprentissage du test set et un fort biais lors de l'évaluation. Pour cela deux sous-ensembles des données sont créés à partir du jeu d'entraînement de façon aléatoire pour éviter un biais seasonale :

$$X\_train = 0.85 * X\_station\_train \quad (2)$$

$$X\_validation = 0.15 * X\_station\_train \quad (3)$$

Traitement de données manquantes

Pour traiter les données manquantes, il existe plusieurs stratégies, par exemple : supprimer la colonne incriminée, supprimer les lignes contenant des valeurs vides, remplir les valeurs manquantes par une valeur fixe (0, valeur aberrante, moyenne des valeurs, etc), etc. Pour choisir une stratégie, une analyse de données manquantes est faite sur tous les paramètres physiques mesurés sur les stations au sol. La Figure 2 montre la distribution des valeurs manquantes sur chaque station pour chaque paramètre.

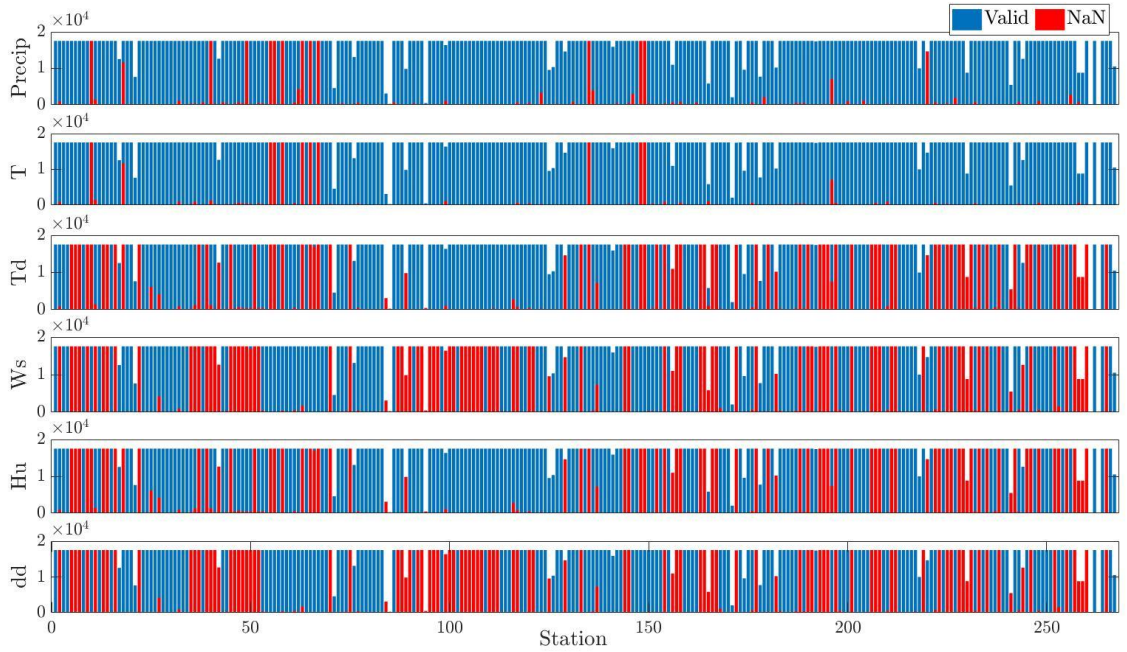


Fig 2. Distribution de valeurs manquantes sur chaque station pour chaque paramètre sur X_train.

A partir de la distribution des valeurs manquantes sur le jeu d'entraînement (X_Train) il est possible d'extraire sur chaque station pour chaque paramètre la quantité de valeurs valides et le pourcentage de valeurs manquant. La précipitation (Precip) et la température (T) ont un pourcentage de NaN nettement inférieur au reste des paramètres.

Pour l'entraînement du modèle on va utiliser seulement les stations dont toutes les données sont disponibles (2 ans de mesures), NaNs incluses. En plus, parmi ces stations on va garder celles qui ont moins de 25% de NaNs dans le paramètre "précipitations". Ensuite les valeurs de NaNs dans chaque paramètre sont remplacées par la moyenne totale de ce paramètre. Cette stratégie est choisie pour avoir une bonne qualité des données d'entraînement, pour simplifier le codage et car le pourcentage de stations filtré par ce critère reste bas (environ 15%).

Normalisation

La normalisation a pour but de ramener toutes les valeurs des variables dans des intervalles comparables. Cela permet d'assurer qu'une variable n'aura pas artificiellement plus d'importance que les autres.

Les variables X sur le data set d'entraînement (X_train) sont normalisées par la normalisation dite "min-max". Elle consiste à ramener toutes les variables entre 0 et 1. Une fois que le X_train est normalisé on sauvegarde le "scaler" pour appliquer a posteriori la même transformation sur le jeu de test au moment de faire les prédictions.

Features structure

On génère un vecteur d'entrée par jour. Ce vecteur consiste à concaténer les 6 paramètres physiques (ff, t, td, hu, dd, precip) qu'on a par heure et on l'ajoute le mois. Cela équivaut à un vecteur xdata de 145 features par jour. La structure des features est la suivante :

$$X_{data}(1day): Concat\left(Month + \sum_{n=1}^{24_{hr}} [ff,\ t\ ,\ td\ ,\ hu\ ,\ dd\ ,\ precip\ ]_n\right) \quad (4)$$

Phase 2 : Modelisation et évaluation

Les hyperparamètres servent à diriger et contrôler l'apprentissage. Selon les valeurs choisies, adaptées ou non, les résultats peuvent varier d'un modèle très performant à un modèle complètement inutilisable. Pour la détermination des hyperparamètres plusieurs combinaisons ont été testé sur les sous-ensembles de jeu de données crée en phase de préparation des données (X_train & X_validation).

La création du modèle s'agit d'un processus fortement itératif selon le cycle illustré sur la Figure 3.

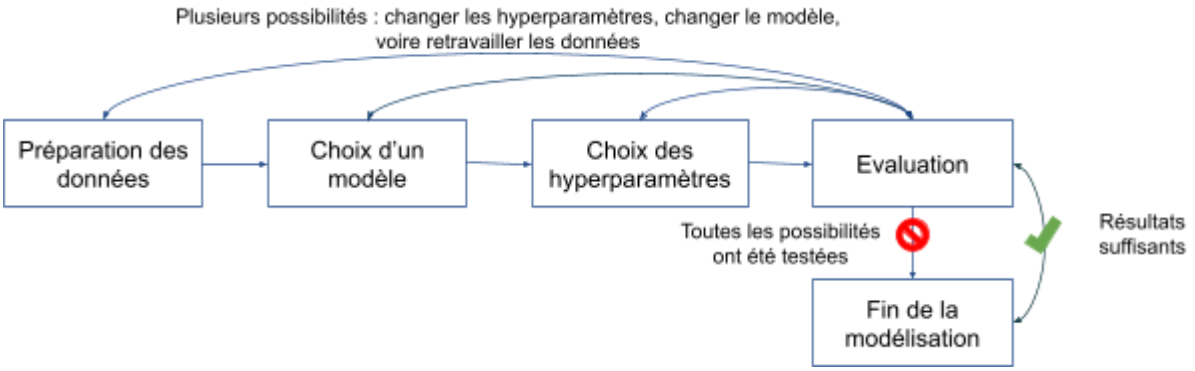


Fig 3. Processus itératif pour la création du modèle IA.

Choix d'un modèle de base

Pour cette étape on a commencé avec un modèle de base de régression avec 4 couches denses de 500,100, 50 et 1 neurones, illustré sur la figure 4.

```
def create_model_(features_size,loss_f='mean_absolute_percentage_error'):
    model = Sequential()
    model.add(Dense(500, input_dim=features_size, activation= "relu"))
    model.add(Dense(100, activation= "relu"))
    model.add(Dense(50, activation= "relu"))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss=loss_f)

    return model
```

Fig 4. Architecture du modèle de base

Choix des hyperparamètres

Dans cette étape on s'est interrogé par rapport à la fonction de loss, classiquement la MSE est utilisée pour ce type des régressions. Par contre le défi impose la fonction MAPE pour le scoring final. A cet effet on a regardé le comportement des deux fonction de perte (MAPE et MSE) sur un petit jeu de données. En même temps, on a aussi analysé l'ordre d'epochs nécessaires dans l'entraînement pour ne pas sur-apprendre le jeu de données. La comparaison des ces deux fonctions de perte sont illustrées sur la Figure 5, a gauche on montre l'évolution quand la loss est MAPE et à droite quand on utilise la MSE. On compare aussi sur la Figure 5 la fonction de perte (en orange) avec le score MAPE (en bleu) obtenu à partir de jeu de données de validation pendant l'entraînement. On remarque que quand on utilise la fonction de perte MSE pour l'apprentissage (Figure 5 à droite) les résultats ne semblent pas prometteurs.

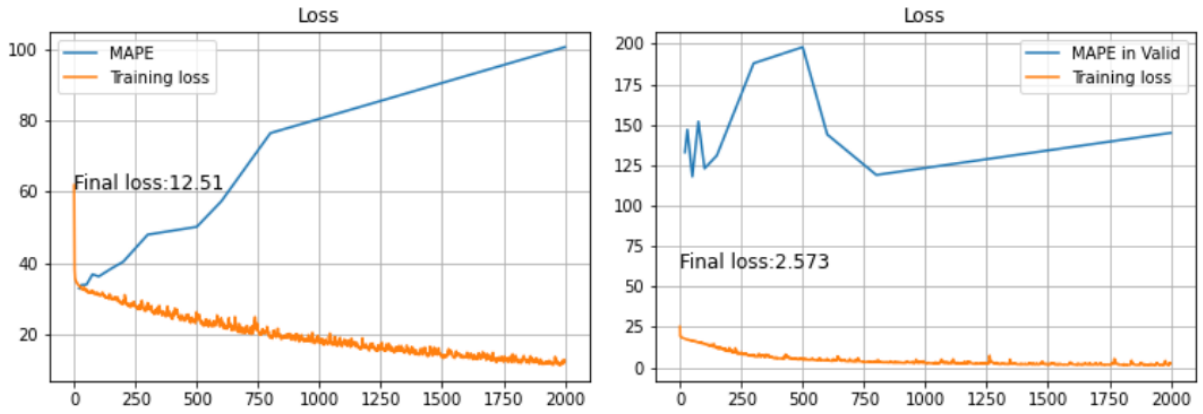


Fig 5 . A **gauche** on montre l'évolution quand la loss est **MAPE** et à **droite** quand on utilise la loss **MSE**. En orange la fonction de perte et en bleu le score MAPE obtenu à partir de jeu de données de validation pendant l'entraînement.

Choix de l'architecture du modèle de base

Même si cela a été étudié sur un jeu de données limité à 10 stations, après cette étude on a gardé MAPE comme fonction de perte et on a choisi l'intervalle entre 50 et 150 epochs comme le plus performant pour ce jeu de données et ce modèle.

Ce modèle de base donne un score sur le défi kaggle de 37.26.

Phase 3 : Amélioration du modèle

Le modèle de base doit maintenant être amélioré, affiné, évalué et comparé de manière à choisir le modèle les plus performant pour soumettre au défi IA. Pour cela deux étapes sur le processus itératif de création du modèle ont été modifiées :

- Phase de préparation des données : création des nouvelles features (features engineering).
- Choix des hyperparamètres : prolonger la phase d'optimisation de hyperparamètres.

Features engineering

Pour l'amélioration du modèle sur la phase de préparation des données on va créer de nouvelles features à partir des existantes. Deux approches ont été analysés et combinés pour créer des features utiles :

- Métier : features orientées à partir d'une expertise métier.
- Brute force : features générées systématiquement en masse. Les transformations simples à partir de features f_1, f_2, f_3, etc peuvent être généré en appliquant des transformations simples de type : ${f_1}^2$, $f_1 f_2$, $f_3 {f_2}^2$, $log(f_1)$, etc . Ces types de transformations simples sont utiles pour les algorithmes basées sur des approches fonctionnelles comme les régressions.

Parmi les paramètres physiques mesurés sur les stations au sol on retrouve des paramètres principaux utilisés par les modèles météorologiques classiques. Ces paramètres sont :

- Humidité (hu) : faibles changements de l' humidité peuvent entraîner de grands écarts de quantités de précipitations, la prévision quantitative de précipitations est donc très sensible à l' humidité.
- Température du point de rosée (Td) : ce paramètre indique à quelle point il faut refroidir un volume d'air, à pression et humidité constantes, afin qu'il devienne saturé. Cette température est fortement liée aux précipitations.
- Le mois est utile pour caractériser la variations saisonnières des précipitations.

- Humidex : est l'indice qui combine humidité et température. Cet indice est très utilisé par les météorologues et peut être calculé à partir de de la température et de la température du point de rosée en utilisant la formule suivante :

$$Humidex = T_{air} + 0.555[6.11 \times e^{5417.7530\,(\frac{1}{273.16} - \frac{1}{273.15+Td})} - 10] \quad (5)$$

- Windchill (WCT) : est l'indice de refroidissement éolien . C'est la sensation de froid ressentie par un organisme sous l'impulsion du vent. Cet indice est utilisé par le modèles météo classique et peut être calculé utilisent la formule suivante où v est la vitesse du vent et Ta la température :

$$WTC = 13.12 + 0.6215 \times Ta - 11.37 \times 0.3965 \times Ta \times v^{0.16} \quad (6)$$

Ces paramètres sont utilisées pour créer les features suivantes :

- $f_1 = hu^2$
- $f_2 = hu * T$
- $f_3 = hu * T * Td$
- $f_4 = T - Td$
- $f_5 = f_5 = T/Td$
- $f_6 = humidex$
- $f_7 = WCT$

La nouvelle structure des features suit l' équation 7. À la donnée des stations (X_data, eq 4) on l'ajoute 7 features créés ($f_{1\to7}$) par heure.

$$X_{data}(1day) = Concat\,(Month + \sum_{n=1}^{24_{hr}} [ff, t, td, hu, dd, precip, f_{1\to7}]_n) \quad (7)$$

Ces nouvelles features sont ajoutées à la phase d'apprentissage pour la création du modèle. Il est possible d'apprécier sur la Figure 6 que son comportement devient beaucoup plus stable qu' avant l'ajout des features (Fig. 5 à gauche). On voit aussi que quand on dépasse autour des 100 epochs le scoring se dégrade, indiquant le sur-apprentissage du jeu d'entraînement.

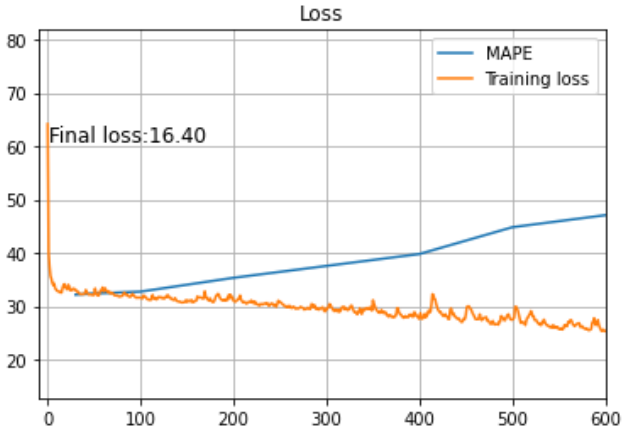


Fig. 6 Évolution de la fonction de perte (MAPE) en ajoutant au modèle les nouvelles features ($f_{1\to7}$).
En orange la fonction de perte et en bleu le score MAPE obtenu à partir de jeu de données de validation pendant l'entraînement.

Optimisation des hyperparamètres

Après avoir amélioré la donnée avec différentes features, il faut optimiser ce qu'on appelle les hyperparamètres. Cette étape requiert d'essayer beaucoup de combinaisons des modèles, profondeur du réseau, nombre de neurones, etc. Malheureusement on est arrivé à cette étape avec peu de temps donc on a conservé le modèle de régression sans tester d'autres modèles (un modèle LSTM est implémenté mais pas encore testé). Parmi les combinaisons testées, celles qui ont montré de meilleures performances (assez proches entre elles) sont illustrées sur le Tableau 1.

Model	Nb Neurones par couche					Scores	
	Couche 1	Couche 2	Couche 3	Couche 4	Couche 5	Loss finale	MAPE in valid set
1	500	100	50	1	-	28.51	38.52
2	200	400	50	1	-	28.30	40.32
3	300	300	100	1	-	28.17	37.31
4	300	400	200	50	1	27.91	41.40
5	300	300	200	50	1	27.78	38.36

Table 1. Résultats des différentes combinaisons d'hyperparateurs.

Phase 4 : Choix du modèle final

Le modèle final est choisi parmi toutes les combinaisons testées dans les étapes précédentes dont le score MAPE est le plus bas possible. L'architecture retenue est illustrée sur la Figure 7 et on a trouvé optimal de l'entraîner avec 150 epochs.

```python
def create_model(features_size,loss_f='mean_absolute_percentage_error'):
    model = Sequential()
    model.add(Dense(300, input_dim=features_size, activation= "relu"))
    model.add(Dense(300, activation= "relu"))
    model.add(Dense(100, activation= "relu"))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mean_absolute_percentage_error')

    return model
```

Fig 7. Architecture du modèle du modèle finale

3. Résultats du modèle finale

Le scoring téléchargé sur le défi **Kaggle (compte : FPiriz) est de 28.3,** voir Figure 8.

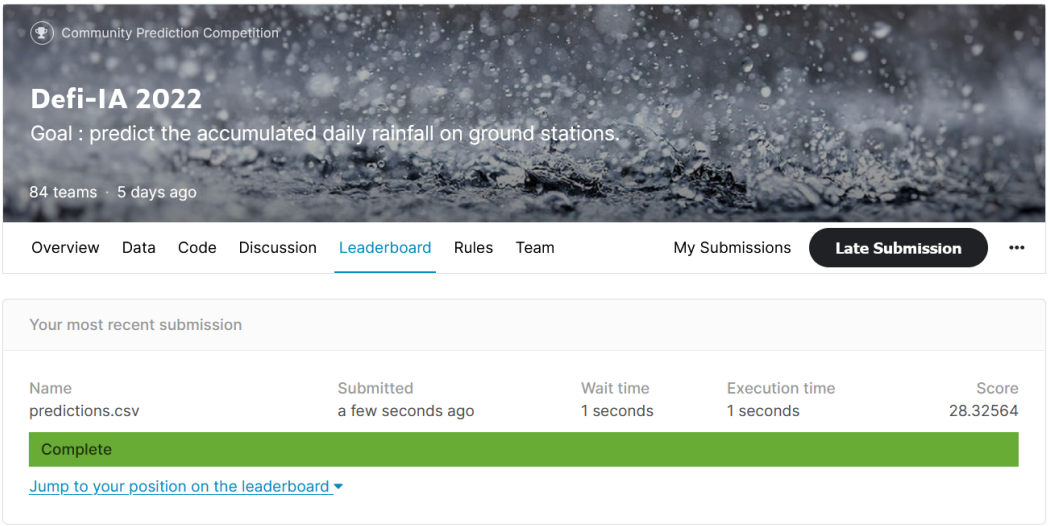


Fig 8. Score défi Kaggle (late submission).

La Figure 9 illustre la prédiction (en rouge) et les vraies mesures (en vert) de précipitations pendant 100 jours non consécutifs (aléatoire). Il est possible d'apprécier sur cette figure que ce modèle tend à sous-estimer les vraies mesures de précipitations. Par contre, la tendance de pics semble suivre celle de vrai mesures.

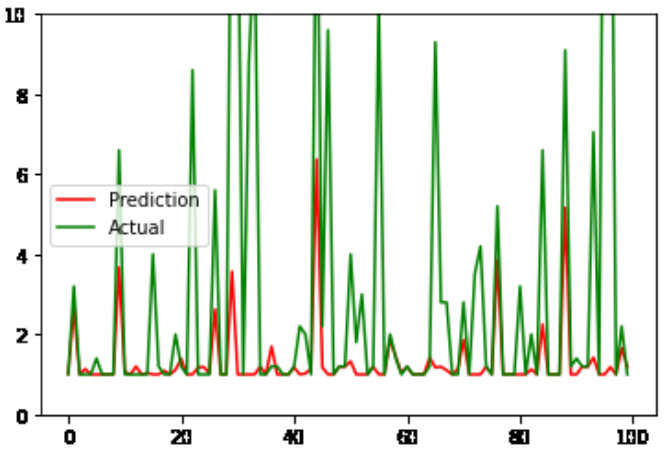


Fig 9. Score défi Kaggle (late submission).

4. Conclusion

On peut conclure qu'on a obtenu avec le modèle final un bon score MAPE sur le défi Kaggle (28.3). On a pu constater une grande amélioration du modèle quand on avait ajouté les features créées à partir de features existantes. Après la phase d'amélioration du modèle on a amélioré le score kaggle de 37.26 (avec le modèle de base) à 28.3.

Malheureusement on n'a pas pu finir les étapes d'amélioration du modèle. On identifie 3 axes fondamentaux d'amélioration :

- Exploitation des données de METEO FRANCE.
- Optimisation des hyperparamètres : les modèles testés ne sont pas suffisants pour conclure qu'on a fini l'étape d'optimisation des hyperparamètres. Par exemple, il faut tester différentes fonction d'activation, d'autres architectures et notamment essayer un série temporelle par heure avec LSTM.
- La stratégie appliquée aux données manquantes peut modifier fortement les résultats obtenues en modélisation, il est donc nécessaire d'évaluer ses conséquences en testant différentes stratégies.

Pendant le déroulement de ce projet on a constaté que la phase de préparation de données a pris plus de 80% du temps de travail. Prévoir plus de temps pour cette phase est un apprentissage pour des projets futurs.

Analysant les résultats obtenues par la modélisation IA et les vraies mesures, ce modèle ne pourra pas remplacer les prévisions météorologiques classiques mais il servira à renforcer et développer les méthodes classiques actuelles.

Un question qu'il reste ouverte est quelle est la raison de la différence de score MAPE sur le jeu de validation, 37.31 (voir Tableau 1) et les scores sur le kaggle, 28.3.