Prévision météo en Australie

Un projet de *data science*

R A P P O R T F I N A L

Omar Choa

Alexandre Winger

Décembre 2023

Table des matières

1. Introduction au projet 1
   1.1. Contexte 1
   1.2. Objectifs 1
2. Compréhension et manipulation des données 3
   2.1. Présentation globale du jeu de données 3
      2.1.1. Étude géographique 3
      2.1.2. Contenu du jeu de données 4
   2.2. Étude des valeurs manquantes 7
      2.2.1. Mise en évidence des valeurs manquantes 7
      2.2.2. Représentation des valeurs manquantes par variable et par station 8
      2.2.3. Représentation de la distribution spatio-temporelle des valeurs manquantes . 9
   2.3. Description statistique 11
      2.3.1. Variables numériques 11
      2.3.2. Variables catégorielles 15
      2.3.3. Répartition des valeurs moyennes par station 16
   2.4. Corrélations 18
      2.4.1. Variables numériques 18
      2.4.2. Variables catégorielles 19
   2.5. Trous chronologiques 20
      2.5.1. Mise en évidence 20
      2.5.2. Analyse des trous chronologiques 22
      2.5.3. Conclusion 24
3. Nettoyage des données 26
   3.1. Quels choix effectuer ? 26
      3.1.1. Variables numériques 26
      3.1.2. Variables catégorielles 31
   3.2. *Preprocessing* 34
      3.2.1. Variables numériques 34
      3.2.2. Variables catégorielles 34
      3.2.3. *Feature engineering* 35
4. Modélisation 35
   4.1. Méthodologie 35
      4.1.1. Phases 35
      4.1.2. Approches 38
   4.2. Résultats 52

4.3. Discussion 54
4.3.1. Interprétation des résultats 54
4.3.2. Remise en contexte 56
5. Conclusion et perspectives 58
6. Références 60

1. Introduction au projet

1.1. Contexte

Notre projet fait partie du cursus de notre formation Data Scientist avec DataScientest.

Notre projet est issu d'une compétition Kaggle, *Rain in Australia*, qui fournit un ensemble de données contenant environ 10 ans d'observations météorologiques quotidiennes provenant de nombreux endroits en Australie. L'objectif de cette compétition Kaggle était de prédire s'il va pleuvoir le jour suivant.

La météorologie est une science avec des applications dans des domaines très divers comme les besoins militaires, la production d'énergie, les transports (aériens, maritimes et terrestres), l'agriculture, la médecine, la construction, la photographie aérienne ou le cinéma. Elle est également appliquée pour la prévision de la qualité de l'air ou de plusieurs risques naturels d'origine atmosphérique.

La météorologie étudie les phénomènes atmosphériques tels que les nuages, les précipitations ou le vent dans le but de comprendre comment ils se forment et évoluent en fonction des paramètres mesurés tels que la pression, la température et l'humidité. Cette discipline scientifique s'appuie sur notamment sur la mécanique des fluides, la thermodynamique, la chimie et les mathématiques.
Grâce à l'informatique et aux simulations numériques, la météorologie moderne permet d'établir des prévisions de l'évolution du temps en s'appuyant sur des modèles mathématiques à court comme à long terme qui assimilent des données de nombreuses sources dont les stations, les satellites et les radars météorologiques.

1.2. Objectifs

Notre projet reprend l'objectif initial de la compétition Kaggle qui l'a inspiré : prédire s'il va pleuvoir le jour suivant. Cet objectif est enrichi de deux autres objectifs : les objectifs secondaires sont de prédire le vent et la température pour le jour suivant.

Chacun de nous a un bagage scientifique, mais aucun de nous n'a d'expérience professionnelle dans le domaine de la météorologie.

Notre projet consiste à résoudre un problème d'apprentissage automatique (*machine learning*), qui comprend des problèmes de classification (prédire s'il va pleuvoir le jour suivant, prédire la direction du vent) ainsi que des problèmes de régression (prédire la vitesse du vent, prédire la température).

Utiliser des techniques d'apprentissage automatique à la météorologie permet de créer un outil qui répond à un besoin quotidien (connaître les prévisions météo) et qui vient compléter les outils existants (simulations numériques des phénomènes atmosphériques). Un objectif supplémentaire serait d'avoir un outil qui consomme moins de ressources informatiques (temps de calcul, capacité de stockage) que les simulations numériques qui peuvent être lourdes et gourmandes en ressources.

2. Compréhension et manipulation des données

2.1. Présentation globale du jeu de données

2.1.1. Étude géographique

Le jeu de données initiales contient environ 10 années d'observations quotidiennes de la météo pour 49 stations en Australie. L'Australie est un pays sec, il y a donc de grandes étendues sans aucune station météo, les stations sont réparties dans les zones fertiles du pays, comme le montre la carte ci-dessous.

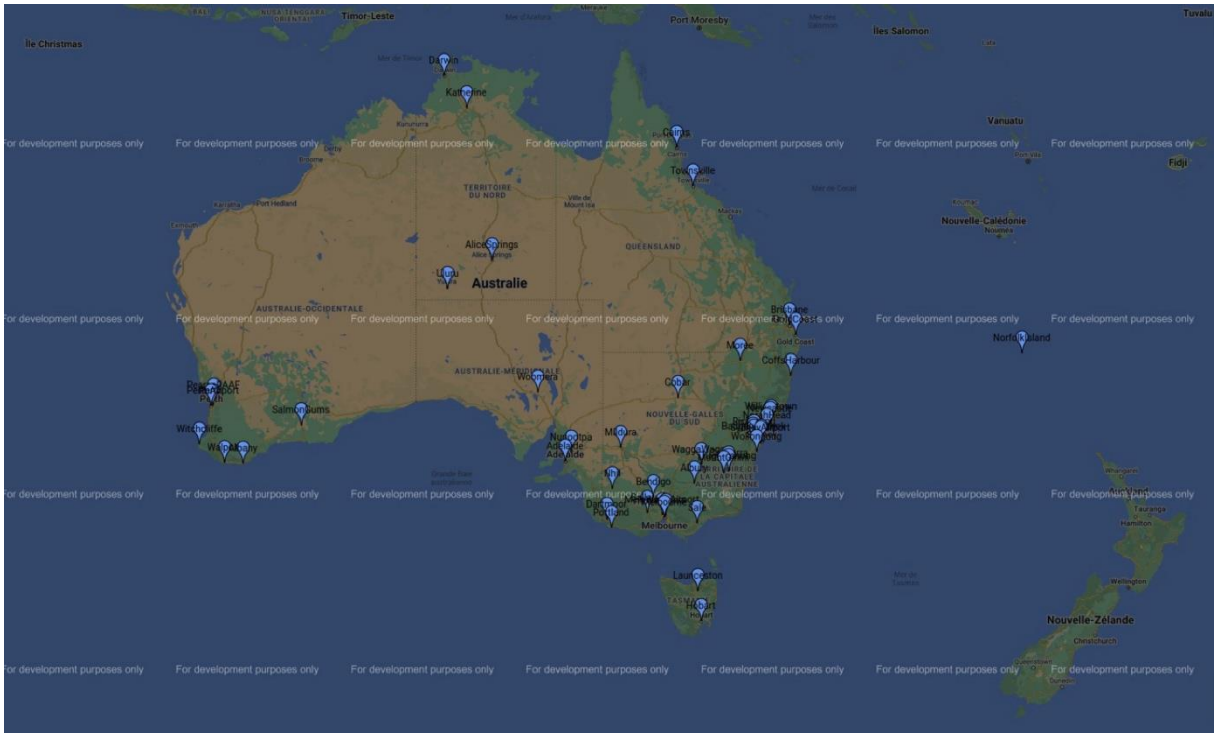


Figure 1. Carte de l'Australie indiquant la répartition géographique des 49 stations météorologiques (indiquées par des repères bleus) de notre jeu de données.

Nous regroupons les stations sur 7 différents territoires qui ont une cohérence géographique. Chacun de ces territoires peut être subdivisé en différentes zones. Ces différentes zones regroupent des stations qui sont géographiquement proches et donc corrélées spatialement. Ci-dessous, la liste des territoires et des zones associées :

- Territoire Sud-Est (33) :
    - Zone de Brisbane (4) : Brisbane, Coffs Harbour, Gold Coast, Moree
    - Zone de Sydney (9) : Badgerys Creek, Newcastle, Norah Head, Perth, Richmond, Sydney, Sidney Airport, William Town, Wollongong
    - Zone de Canberra (5) : Albury, Canberra, Mount Ginini, Tuggeranong, Wagga Wagga
    - Zone de Melbourne (6) : Ballarat, Bendigo, Melbourne, Melbourne Airport, Sale, Watsonia
    - Zone de Portland (3) : Dartmoor, Mount Gambier, Nhill, Portland
    - Zone d'Adelaïde (2) : Adelaide, Nuriootpa
    - Cobar (1)
    - Mildura (1)
    - Woomera (1)
- Territoire Sud-Ouest (7) :
    - Zone de Perth (3) : Perth, Perth Airport, Pearce RAAF
    - Pointe Sud-Ouest (3) : Albany, Walpole, Witchcliffe
    - Salmon Gums
- Territoire Centre (2) : Alice Spring, Uluru
- Territoire Plein Nord (2) : Darwin, Katherine
- Territoire Nord-Est (2) : Cairns, Townsville
- Territoire de l'Ile de Tasmanie (2) : Hobart, Launceston

Ces corrélations spatiales peuvent induire des corrélations dans les observations météo. Nous prévoyons de tester cette hypothèse.

2.1.2. Contenu du jeu de données

Le jeu de données contient 145 460 lignes et 23 colonnes. La volumétrie est plutôt légère (14,1 Mo).

Les deux premières colonnes sont particulières. La première colonne est la date du jour. La deuxième colonne est le nom de la station où sont faites les mesures. Les 21 autres colonnes sont des variables, numériques (quantitatives) ou catégorielles (qualitatives), dont la variable cible.

Chaque ligne du jeu de données correspond à une observation des 21 variables pour une station donnée et un jour donné.

Dans ces 23 variables, nous avons 16 variables numériques et 7 variables catégorielles.

Liste des 16 variables numériques :

- `Cloud3pm` : fraction du ciel obscurcie par les nuages à 9h (octas)
- `Cloud9am` : fraction du ciel obscurcie par les nuages à 15h (octas)
- `Evaporation` : quantité d'eau évaporée dans un bac de classe A sur les dernières 24h, mesurée à 9h00 (mm)
- `Humidity3pm` : humidité mesurée à 15h (%)
- `Humidity9am` : humidité mesurée à 9h (%)
- `MaxTemp` : température maximale enregistrée ce jour (°C)
- `MinTemp` : température minimale enregistrée ce jour (°C)
- `Pressure3pm` : pression atmosphérique mesurée à 15h et corrigée au niveau de la mer (hPa)
- `Pressure9am` : pression atmosphérique mesurée à 9h et corrigée au niveau de la mer (hPa)
- `Rainfall` : quantité d'eau de pluie enregistrée ce jour (mm)
- `Sunshine` : nombre d'heures d'ensoleillement dans la journée (heures)
- `Temp3pm` : température à 15h (°C)
- `Temp9am` : température à 9h (°C)
- `WindGustSpeed` : vitesse de la plus forte rafale de vent sur les dernières 24h (km/h)
- `WindSpeed3pm` : vitesse moyenne du vent de 14h50 à 15h (km/h)
- `WindSpeed9am` : vitesse moyenne du vent de 8h50 à 9h (km/h)

L’humidité mesurée est l’humidité relative : c’est un pourcentage qui représente le rapport de la pression partielle de la vapeur d'eau contenue dans l'air sur la pression de vapeur saturante (ou tension de vapeur) à la même température. Elle est donc une mesure du rapport entre le contenu en vapeur d'eau de l'air et sa capacité maximale à en contenir dans ces conditions.

Liste des 7 variables catégorielles :

- `Date` : date du jour des mesures
- `Location` : nom de la station où sont faites les mesures
- `WindDir3pm` : direction du vent à 15h (rose des vents à 16 directions)
- `WindDir9am` : direction du vent à 9h (rose des vents à 16 directions)
- `WindGustDir` : direction de la plus forte rafale de vent enregistrée sur les dernières 24h (rose des vents à 16 directions)
- `RainToday` : réponse à la question : « a-t-il plu ce jour ? » (booléen)
- `RainTomorrow` : réponse à la question : « a-t-il plu le lendemain ? » (booléen). C'est notre variable cible.

Nous nous sommes demandé si le problème pouvait être subdivisé en 49 problèmes, avec un sous-ensemble de données par station. Mais cette stratégie aurait pour conséquence de faire perdre toutes les informations sur les corrélations spatiales entre les différents sous-ensembles de données. Par exemple, le sous-ensemble « Melbourne » serait complètement scindé du sous-ensemble « Melbourne Airport », alors qu'il est tout à fait raisonnable de penser que les observations de ces deux sous-ensembles soient corrélées, puisque ces deux stations sont séparées par une distance d'une quinzaine de kilomètres.

Une stratégie plus fine pourrait être d'avoir des sous-ensembles qui regroupent les stations qui sont géographiquement proches. Il y aura probablement des corrélations sur les observations entre Melbourne et Melbourne Airport qui sont séparés par une distance d'une quinzaine de kilomètres, par contre il serait étonnant d'avoir des corrélations entre Melbourne et Perth qui sont séparés par une distance supérieure à 3 000 km.

2.2. Étude des valeurs manquantes

2.2.1. Mise en évidence des valeurs manquantes

Les premières visualisations du jeu de données montrent qu'il y a un certain nombre de valeurs manquantes. Le graphique ci-dessous représente la distribution des données par variable.

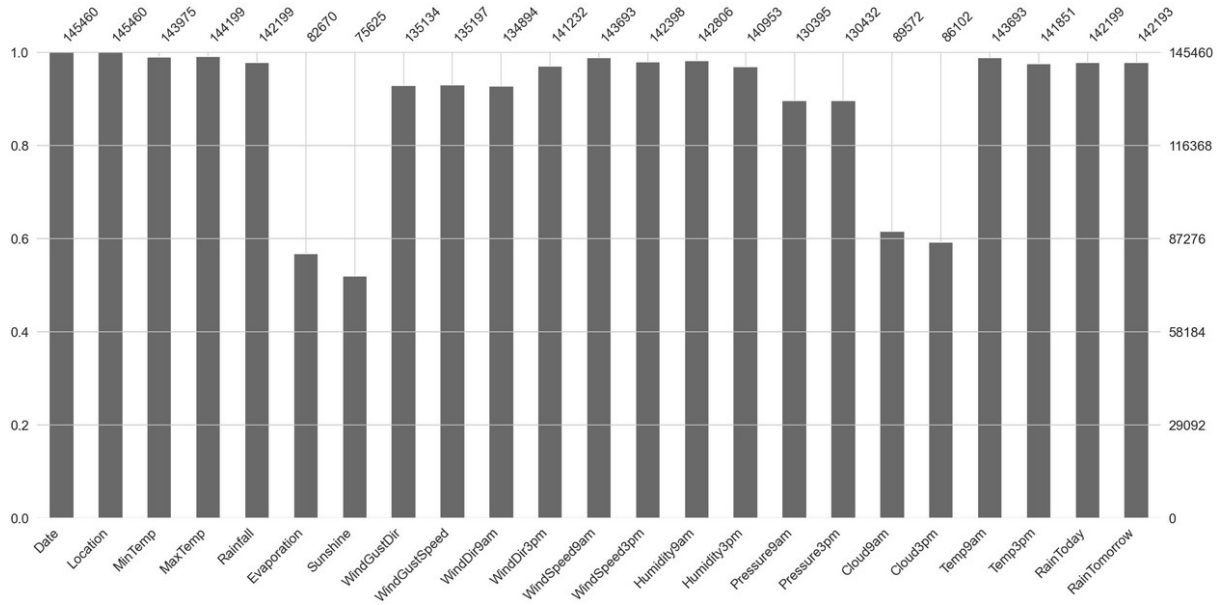


Figure 2. Distribution des données par variable sous forme de *bar plot*.

Nous constatons que 4 variables (`Evaporation`, `Sunshine`, `Cloud9am`, `Cloud3pm`) sont très peu mesurées, avec environ 40 % de valeurs manquantes.

La figure ci-dessous montre que la variable `Evaporation` est systématiquement absente d'un certain nombre de stations.

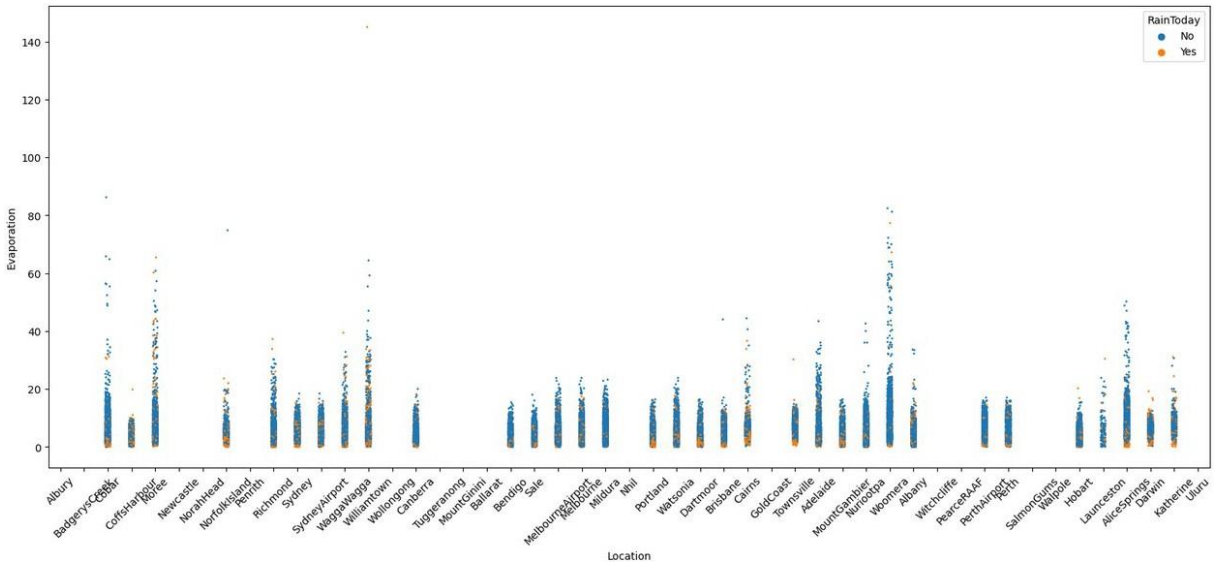


Figure 3. Distribution des données par station météorologique sous forme de *strip plot*.

2.2.2. Représentation des valeurs manquantes par variable et par station

Pour systématiser l'étude, nous avons représenté une *heatmap* des valeurs manquantes par variable et par station, dans la figure suivante.

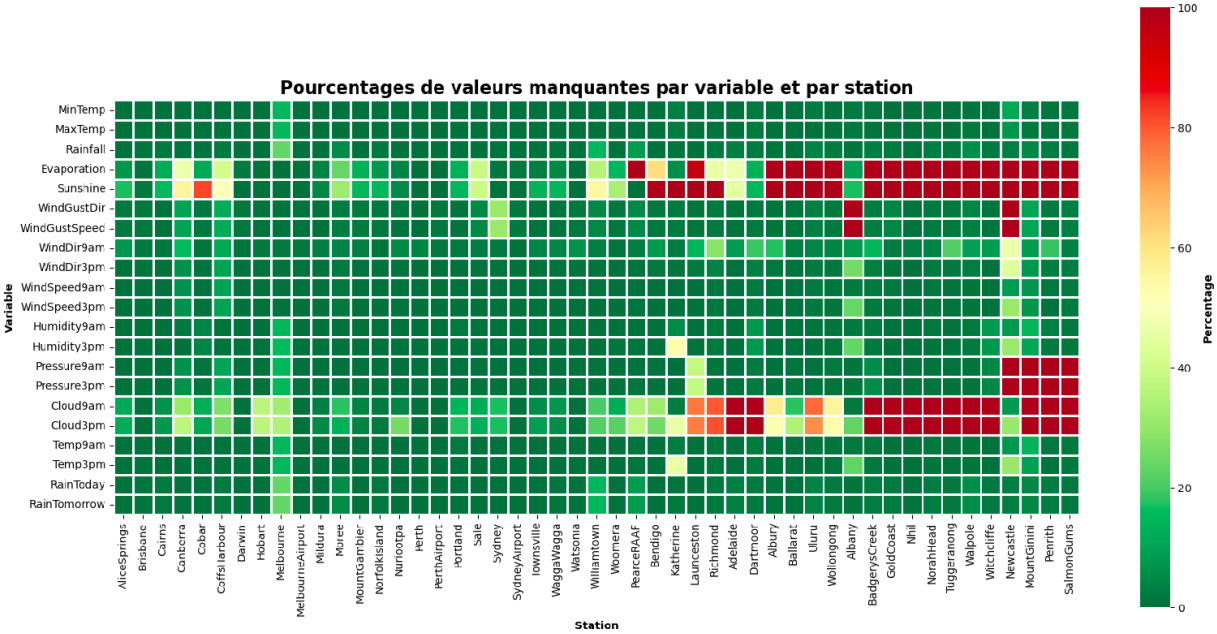


Figure 4. *Heatmap* des valeurs manquantes par variable et par station.

Dans certaines stations, des variables sont systématiquement non mesurées, pour des raisons que nous ignorons. Il est probable que ce soit dû à un manque de moyens matériels : des stations peuvent ne pas disposer des instruments de mesure nécessaires.

Nous avons distingué 9 groupes de stations en fonction des variables mesurées. Cette distinction donne lieu à une division du jeu de données en 9 sous-ensembles de données.

- Groupe 1 : toutes les variables sont mesurées. 26 stations.
- Groupe 2 : seule la variable `Evaporation` n'est pas mesurée. 1 station.
- Groupe 3 : seule la variable `Sunshine` n'est pas mesurée. 4 stations.
- Groupe 4 : seules les variables `Cloud3pm` et `Cloud9am` ne sont pas mesurées. 2 stations.
- Groupe 5 : seules les variables `Evaporation` et `Sunshine` ne sont pas mesurées. 4 stations.
- Groupe 6 : seules les variables `WindGustDir` et `WindGustSpeed` ne sont pas mesurées. 1 station.
- Groupe 7 : les variables `Cloud3pm`, `Cloud9am`, `Evaporation`, `Sunshine` ne sont pas mesurées. 7 stations.
- Groupe 8 : les variables `Evaporation`, `Pressure3pm`, `Pressure9am`, `Sunshine`, `WindGustDir`, `WindGustSpeed` ne sont pas mesurées. 1 station.
- Groupe 9 : les variables `Cloud3pm`, `Cloud9am`, `Evaporation`, `Pressure3pm`, `Pressure9am`, `Sunshine` ne sont pas mesurées. 3 stations.

Ces 9 groupes peuvent donner lieu à 9 familles de modèles pour le travail de modélisation et de prédiction.

2.2.3. Représentation de la distribution spatio-temporelle des valeurs manquantes

Nous avons aussi investigué la distribution des valeurs manquantes dans l'espace et le temps.

La figure suivante montre la distribution spatio-temporelle des données pour la variable `Evaporation`.

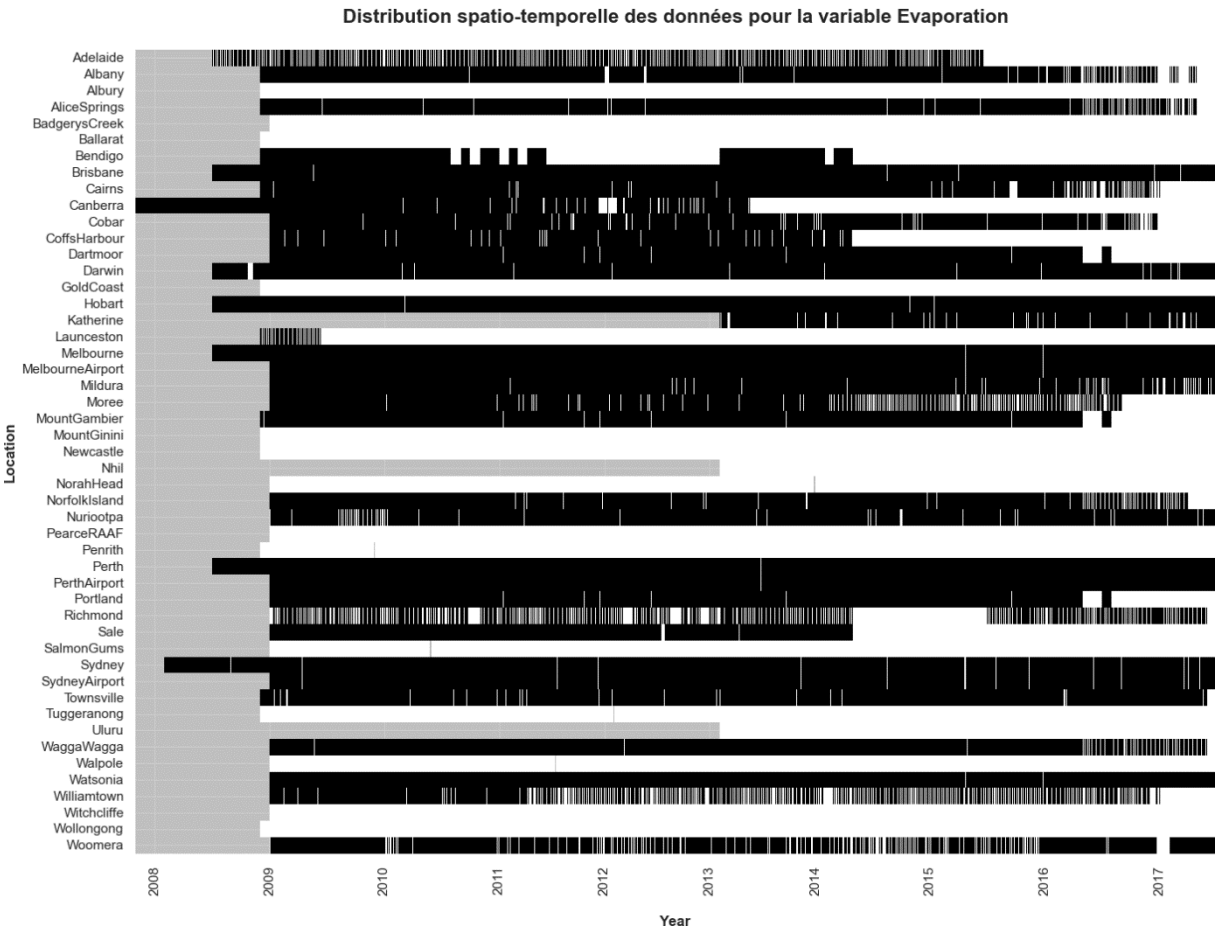


Figure 5. Distribution spatio-temporelle des données pour la variable `Evaporation`. Les bandes noires représentent des données, les bandes blanches, des valeurs manquantes, et les bandes grises, des périodes antérieures au début de l'enregistrement des données.

Les stations qui ne mesurent pas du tout la variable `Evaporation` sont bien visibles avec des bandes blanches sur toute la longueur.

Nous remarquons que des stations comme Bendigo et CoffsHarbour arrêtent de mesurer la variable `Evaporation` à partir de 2014. Il n'y a aucune explication sur l'arrêt de ces mesures.

2.3. Description statistique

2.3.1. Variables numériques

Comme indiqué en introduction, le jeu de données contient **16 variables numériques.**

Table 1 présente des statistiques descriptives pour ces variables.

Table 1. Statistiques descriptives pour les 16 variables numériques.

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine
count	143975.000000	144199.000000	142199.000000	82670.000000	75625.000000
mean	12.194034	23.221348	2.360918	5.468232	7.611178
std	6.398495	7.119049	8.478060	4.193704	3.785483
min	-8.500000	-4.800000	0.000000	0.000000	0.000000
25%	7.600000	17.900000	0.000000	2.600000	4.800000
50%	12.000000	22.600000	0.000000	4.800000	8.400000
75%	16.900000	28.200000	0.800000	7.400000	10.600000
max	33.900000	48.100000	371.000000	145.000000	14.500000

	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm
count	135197.000000	143693.000000	142398.000000	142806.000000	140953.000000
mean	40.035230	14.043426	18.662657	68.880831	51.539116
std	13.607062	8.915375	8.809800	19.029164	20.795902
min	6.000000	0.000000	0.000000	0.000000	0.000000
25%	31.000000	7.000000	13.000000	57.000000	37.000000
50%	39.000000	13.000000	19.000000	70.000000	52.000000
75%	48.000000	19.000000	24.000000	83.000000	66.000000
max	135.000000	130.000000	87.000000	100.000000	100.000000

	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm
count	130395.00000	130432.000000	89572.000000	86102.000000	143693.000000	141851.00000
mean	1017.64994	1015.255889	4.447461	4.509930	16.990631	21.68339
std	7.10653	7.037414	2.887159	2.720357	6.488753	6.93665
min	980.50000	977.100000	0.000000	0.000000	-7.200000	-5.40000
25%	1012.90000	1010.400000	1.000000	2.000000	12.300000	16.60000
50%	1017.60000	1015.200000	5.000000	5.000000	16.700000	21.10000
75%	1022.40000	1020.000000	7.000000	7.000000	21.600000	26.40000
max	1041.00000	1039.600000	9.000000	9.000000	40.200000	46.70000

À première vue, tout paraît cohérent. Par exemple, les valeurs minimales et maximales semblent plausibles.

Toutefois, en regardant de plus près, nous décelons **quelques irrégularités.** Par exemple, pour la variable `Rainfall`, 75% des valeurs se trouvent en dessous de 0.8, alors que le maximum est de 371. Son écart-type (8.48) est également plusieurs fois plus grand que sa

moyenne (2.36). Nous pouvons constater un phénomène similaire pour la variable `Evaporation`, dont 75% des valeurs se trouvent en dessous de 7.4, alors que le maximum est de 145.

Ces observations semblent **difficilement imputables aux valeurs manquantes.** À titre d'exemple, la variable `Sunshine`, dont le taux de NaN est de 48.01%, présente une distribution tout à fait cohérente.

Afin de mieux appréhender ces tendances, il est utile de les **visualiser** après suppression des valeurs manquantes et mise à l'échelle par standardisation :

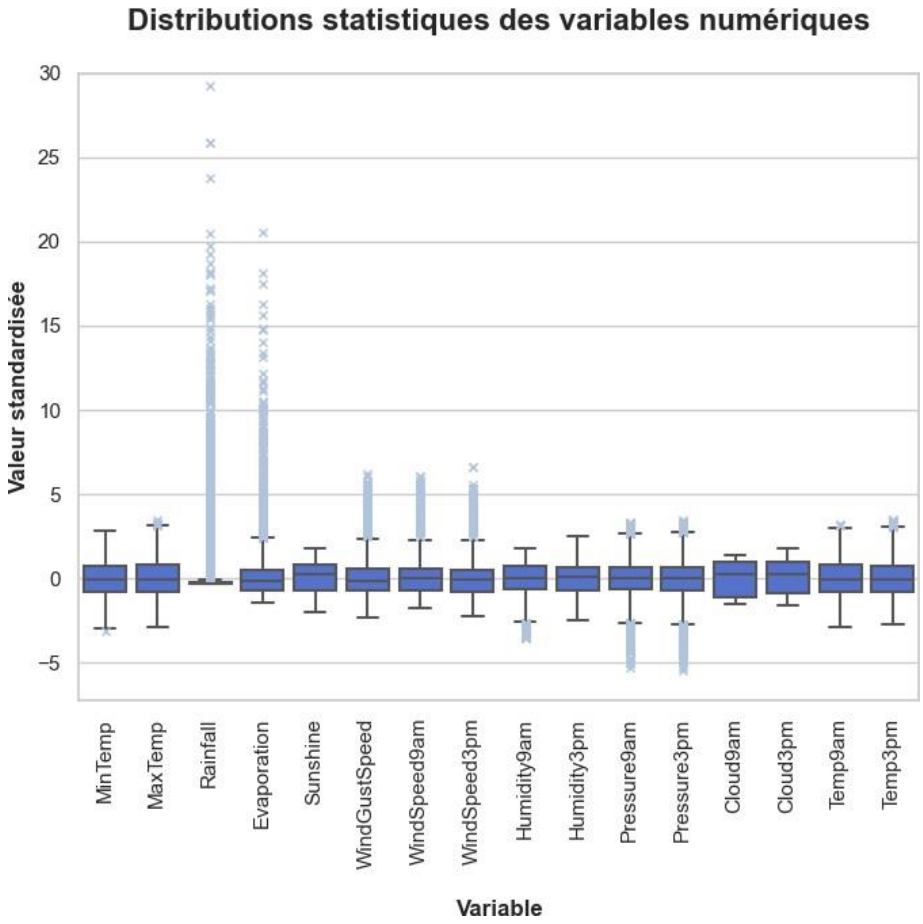


Figure 6. Les distributions statistiques des 16 variables numériques.

Figure 6 met en évidence la présence de **plusieurs distributions** aux **schémas inhabituels,** caractérisés par de **nombreuses valeurs extrêmes,** notamment dans le cas des variables suivantes :

- `Rainfall`
- `Evaporation`
- `WindGustSpeed`
- `WindSpeed9am`
- `WindSpeed3pm`
- `Humidity9am`
- `Pressure9am`
- `Pressure3pm`

Regardons une de ces variables de plus près.

Figure 7 présente la distribution de `Evaporation` (en valeurs absolues, et non plus standardisées).

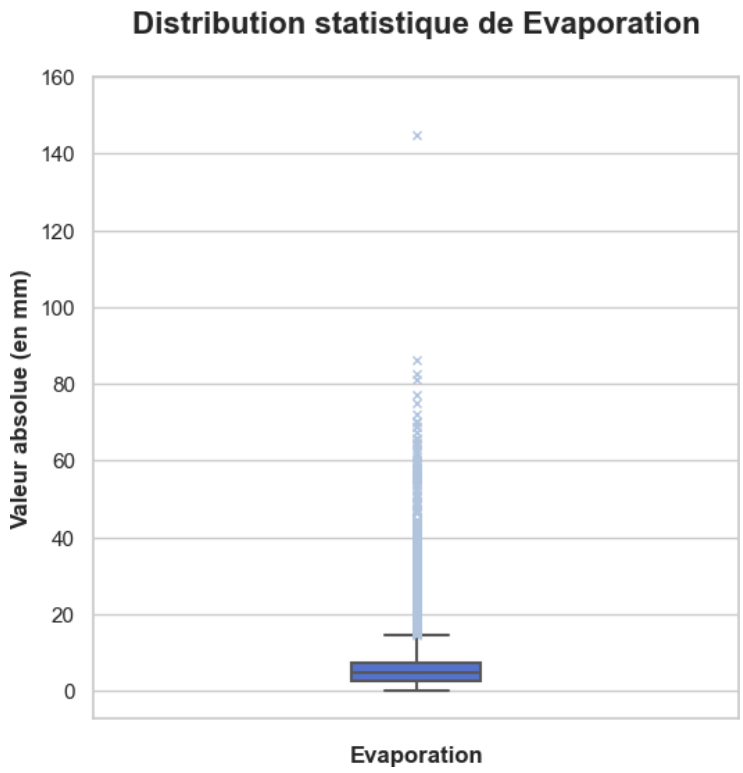


Figure 7. La distribution statistique de `Evaporation`.

Ce graphique rejoint les premières observations effectuées à partir des statistiques descriptives, à savoir, qu'une grande majorité des valeurs (75%) se trouvent en dessous d'un seuil très faible (7.4).

Les variables sous étude étant des **paramètres météorologiques,** nous pourrions supposer qu'elles soient **fonction de la date et du lieu** d'enregistrement des données ; une décomposition sur ces axes pourrait donc se révéler instructive.

Figure 8 présente la distribution de `Evaporation` selon les stations météorologiques. Elle montre que **les valeurs extrêmes**—presque toutes supérieures—**varient effectivement en fonction des stations.** Plus important, elle permet aussi de constater que **certaines stations ne disposent d'aucune entrée** pour `Evaporation` dans le jeu de données.

Ces informations sont importantes à prendre en compte dans la gestion des valeurs manquantes, les choix d'algorithmes et de modèles d'apprentissage automatique et l'interprétation des résultats.

Concernant les valeurs manquantes en particulier, **deux stratégies** se dessinent :

- L'imputation d'un critère statistique de position (moyenne / médiane / mode), ou
- La suppression.

Nous tenterons d'explorer ces deux pistes en parallèle dans la suite du projet.

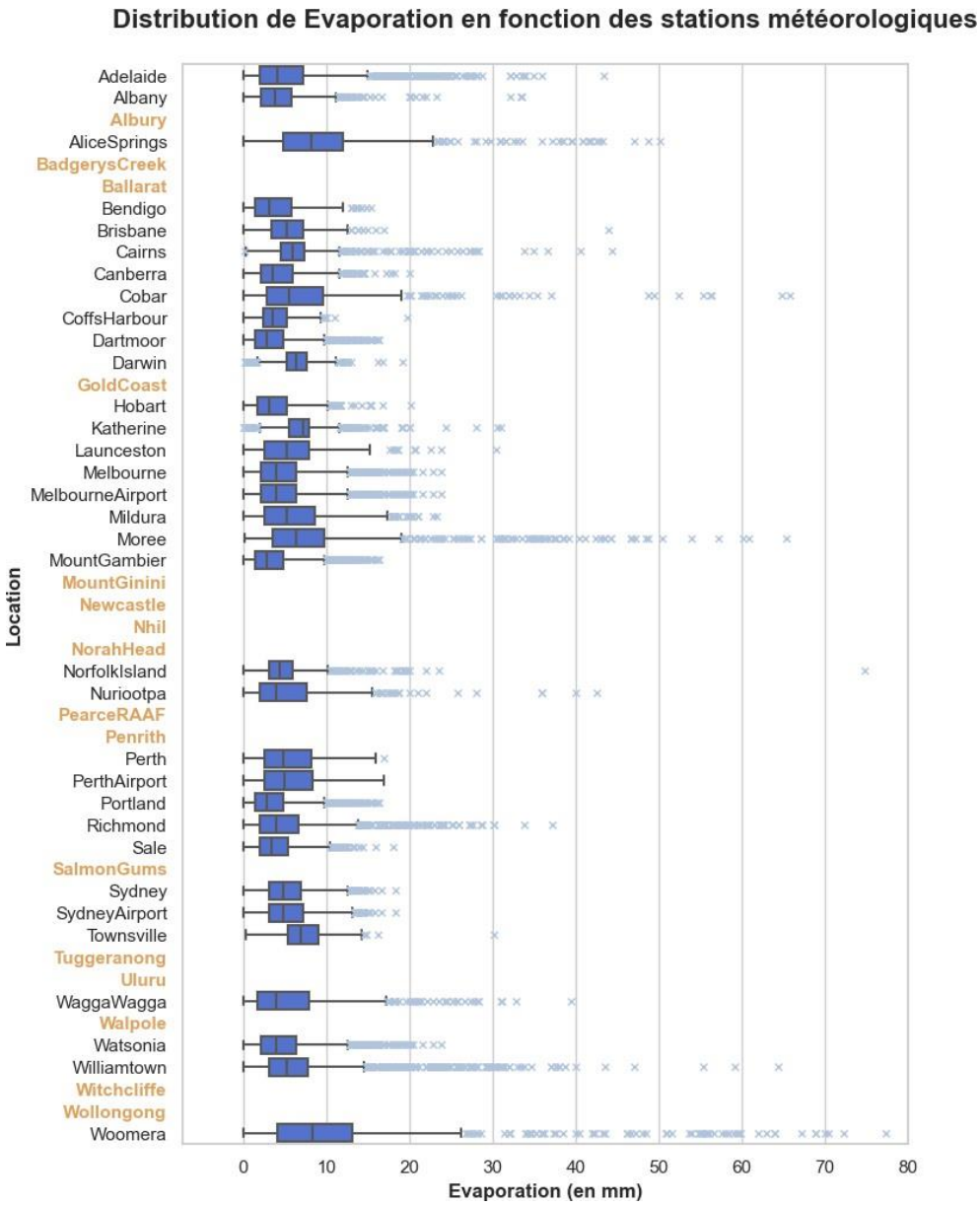


Figure 8. La distribution de `Evaporation` en fonction des stations météorologiques. Les noms des stations ne disposant d'aucune donnée pour cette grandeur sont surlignés. Afin d'améliorer la lisibilité du graphique, l'abscisse a été arrêtée à 80, excluant 4 valeurs extrêmes supérieures.

2.3.2. Variables catégorielles

Les variables catégorielles sont les suivantes :

- `WindGustDir`
- `WindDir9am`
- `WindDir3pm`
- `RainToday`
- `RainTomorrow`

Il n'y a rien d'inattendu dans la distribution des 16 modalités (correspondant aux 16 points cardinaux) des 3 variables liées au vent (`WindGustDir`, `WindDir9am`, `WindDir3pm`).

En revanche, les 2 variables liées à la pluie (`RainToday`, `RainTomorrow`) présentent un **fort déséquilibre,** comme illustré dans Figure 9.

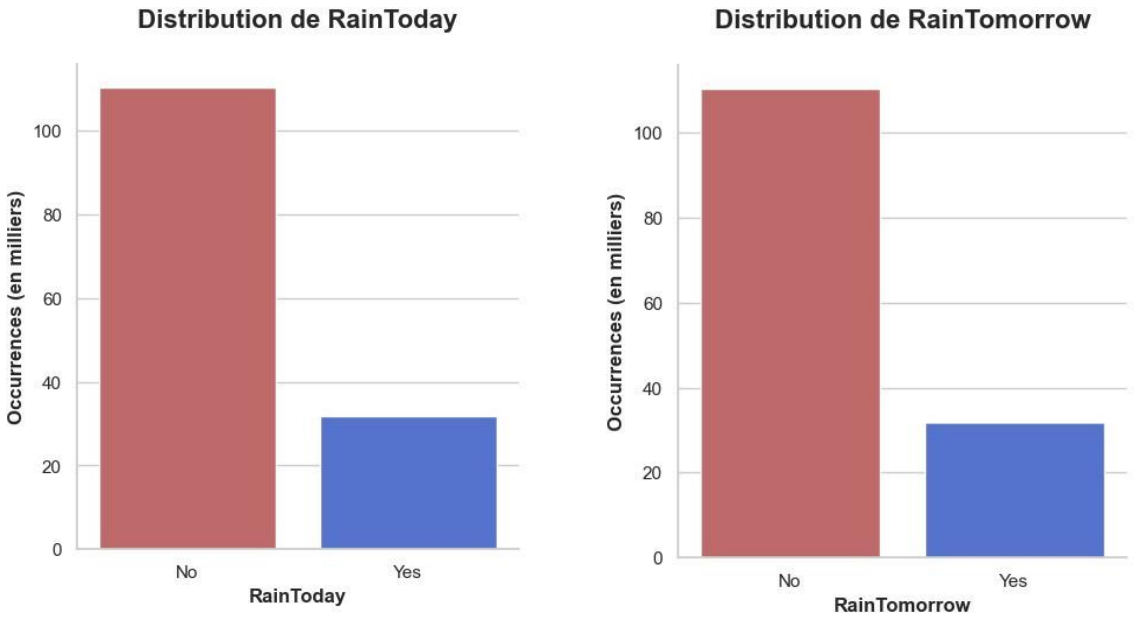


Figure 9. Distributions de `RainToday` et de `RainTomorrow`.

Comme `RainToday` est la variable cible principale de l'étude, il sera nécessaire de mobiliser des méthodes conçues pour le traitement des échantillons contenant une forte disparité entre les classes à prédire (par exemple : rééchantillonnage, techniques de classification avancée).

En outre, ces deux variables catégorielles sont intimement liées à la variable numérique `Rainfall` (dont la distribution reste à clarifier) : les 3 sont toujours présentes ou absentes ensemble.

2.3.3. Répartition des valeurs moyennes par station

Pour toutes les variables numériques, nous avons regardé la répartition des valeurs moyennes par station, et nous avons fait le même exercice après standardisation (valeurs moyennes « scalées »), comme c'est illustré par la figure ci- dessous.

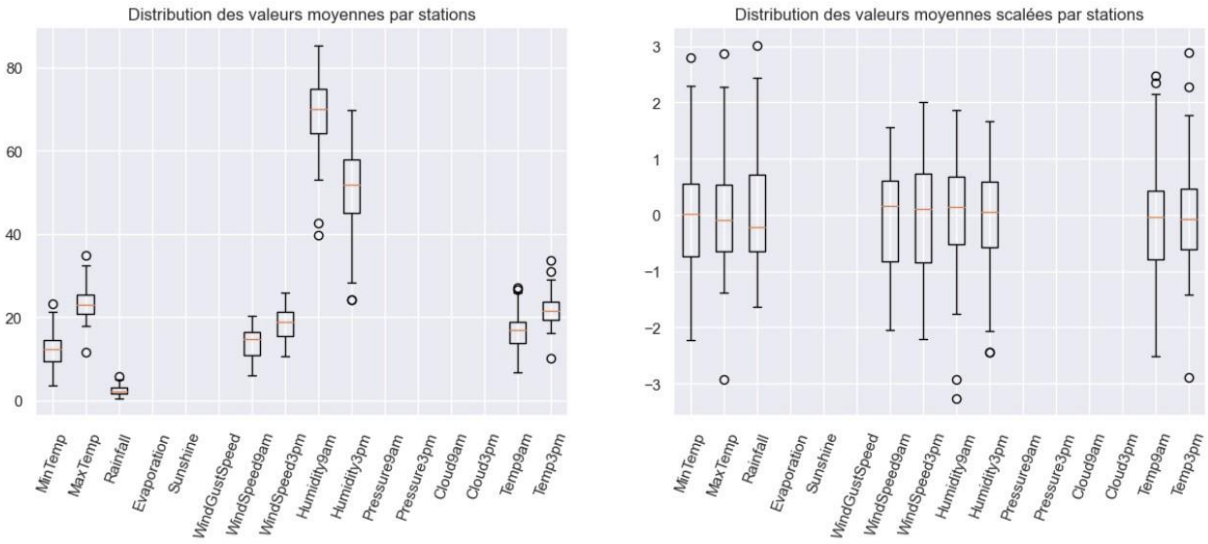


Figure 10. Distribution des valeurs moyennes par station sans (gauche) et avec (droite) standardisation.

Nous constatons qu'il y a une grande dispersion des valeurs moyennes. Les mesures d'humidité ont des valeurs nettement supérieures aux autres variables mesurées. Comme attendu, la standardisation permet d'avoir des valeurs comparables.

Nous avons répété cet exercice sur le jeu de données après avoir enlevé toutes les lignes contenant au moins une valeur manquante. Ce qui permet d'avoir la figure ci-dessous.

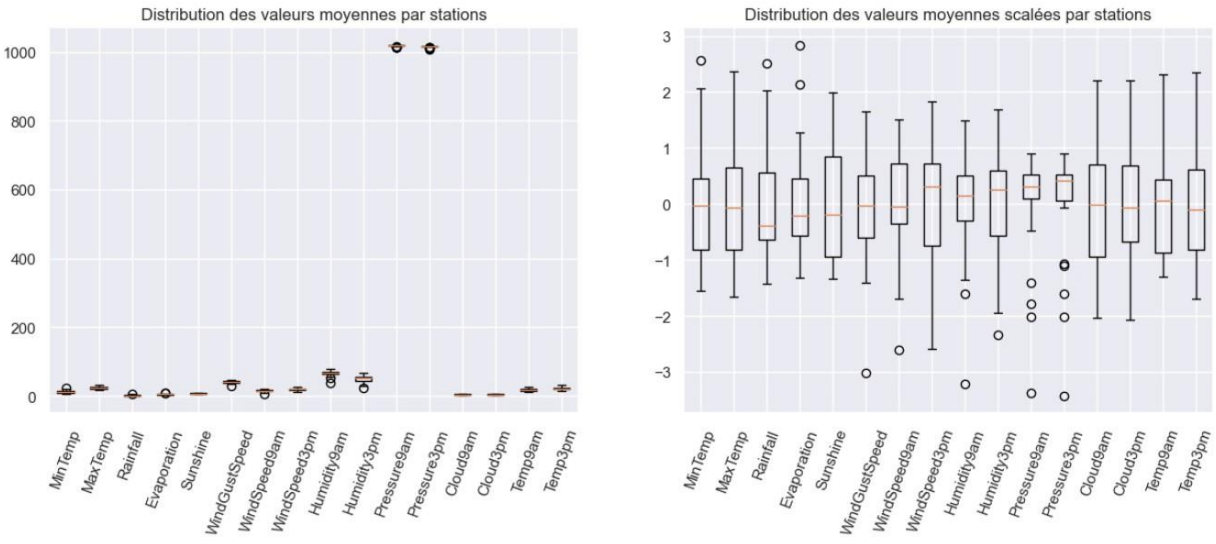


Figure 11. Distribution des valeurs moyennes par station après suppression de toutes les lignes contenant des valeurs manquantes, sans (gauche) et avec (droite) standardisation.

Cette analyse complémentaire permet de constater que les mesures de pression sont bien supérieures aux autres mesures, s'il n'y a pas de normalisation.

2.4. Corrélations

2.4.1. Variables numériques

Figure 12 présente une carte de corrélation des variables numériques avec `RainToday` et `RainTomorrow` établie en première approche, avec :

- La binarisation de ces deux variables catégorielles sous les noms `RainTodayNum` et `RainTomorrowNum` ;
- La simple suppression des valeurs manquantes ; et
- L'utilisation du coefficient de corrélation de Pearson ρ comme métrique.

La ligne et la colonne qui correspondent à `RainTomorrowNum`, laquelle représente la **variable cible** `RainTomorrow`, ne montre pas de corrélation particulièrement importante : la plus forte est celle avec `Sunshine` (–0.45), suggérant la nécessité d'apporter un soin particulier à la stratégie de gestion des valeurs manquantes de cette dernière.

En considérant uniquement les variables **explicatives,** nous pouvons formuler les observations suivantes :

- Les variables liées à la température (`MinTemp`, `MaxTemp`, `Temp9am`, `Temp3pm`) semblent être relativement corrélées entre elles ($\rho > 0.7$).
- L'ensoleillement (`Sunshine`) et la couverture nuageuse (`Cloud9am`, `Cloud3pm`) semblent être négativement corrélés ($\rho \cong -0.7$).
- L'ensoleillement (`Sunshine`) et le taux d'humidité à 15 h (`Humidity3pm`) semblent également être négativement corrélés ($\rho \cong -0.7$).

Ces observations pourraient informer la stratégie de simplification du jeu de données à travers la gestion des valeurs manquantes (en justifiant la suppression de certaines variables si elles peuvent être « représentées » par d'autres variables avec lesquelles elles sont fortement corrélées), la réduction de dimensions et d'autres méthodes.

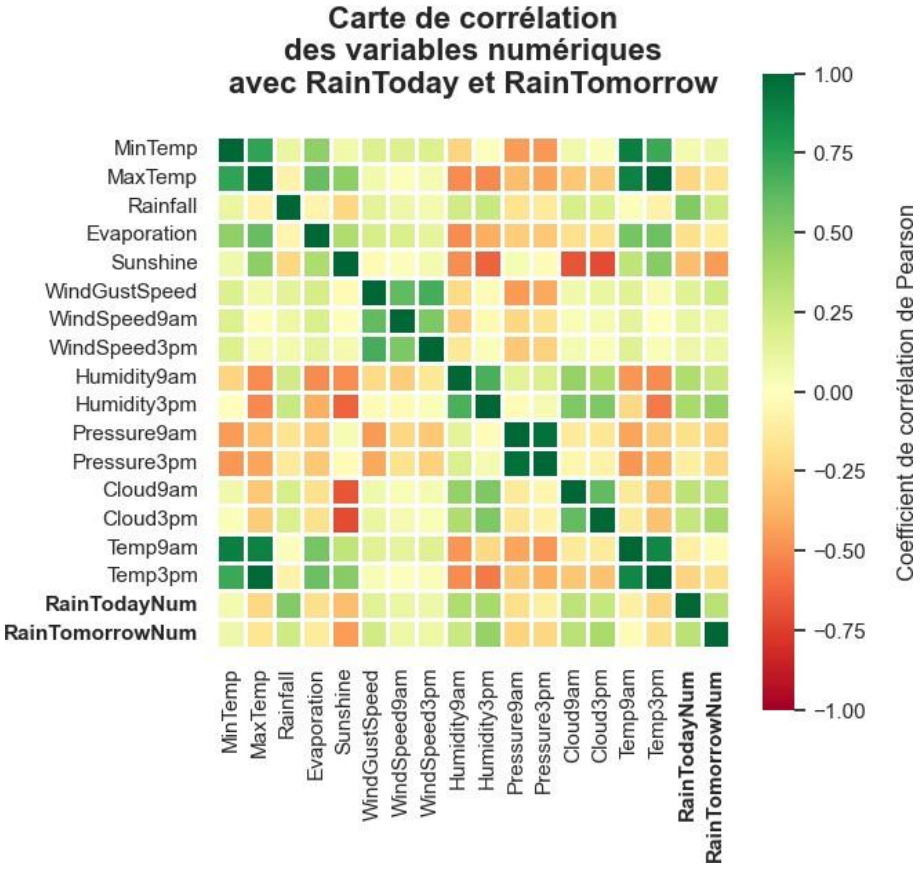


Figure 12. Carte de corrélation des variables numériques avec `RainToday` et de `RainTomorrow`.

2.4.2. Variables catégorielles

Table 2 présente les résultats du test de χ^2 d'indépendance visant à déterminer si un lien statistique existe entre chacune des variables catégorielles explicatives et la variable cible `RainTomorrow`, elle aussi catégorielle.

Les hypothèses formulées sont les suivantes :

$$\begin{cases} H_0 : \text{la variable explicative est indépendante de la variable cible} \\ H_1 : \text{la variable explicative n'est pas indépendante de la variable cible} \end{cases}$$

Le seuil de la valeur-p fixé pour le rejet de l'hypothèse nulle est de 0.05.

Comme les valeurs-p sont toutes en-dessous de ce seuil, nous rejetons l'hypothèse H_0 et concluons à l'hypothèse H_1 : les variables explicatives ne sont pas indépendantes de `RainTomorrow`.

Table 2. Résultats du test de χ^2 d'indépendance
entre les variables catégorielles explicatives et `RainTomorrow`.

Variable	Statistique du test χ^2	Valeur-p
`Date`	16735	0.0
`Location`	3563	0.0
`WindGustDir`	1517	0.0
`WindDir9am`	2178	0.0
`WindDir3pm`	1283	0.0
`RainToday`	13799	0.0

2.5. Trous chronologiques

2.5.1. Mise en évidence

Lors de l'exploration de notre jeu de données, nous avons souhaité effectuer une visualisation de l'évolution de certaines grandeurs au cours du temps pour une station. Le jeu de données original contenant encore des valeurs manquantes (NaN), il a fallu choisir une grandeur qui ne contienne aucun NaN pour une station donnée. Un rapide coup d'œil à la *heatmap* des NaN nous fait porter notre choix sur la grandeur `MaxTemp` pour la station Cairns, qui a l'air très propre. Une analyse plus fine confirme ce choix : aucun NaN n'est présent dans cette colonne.

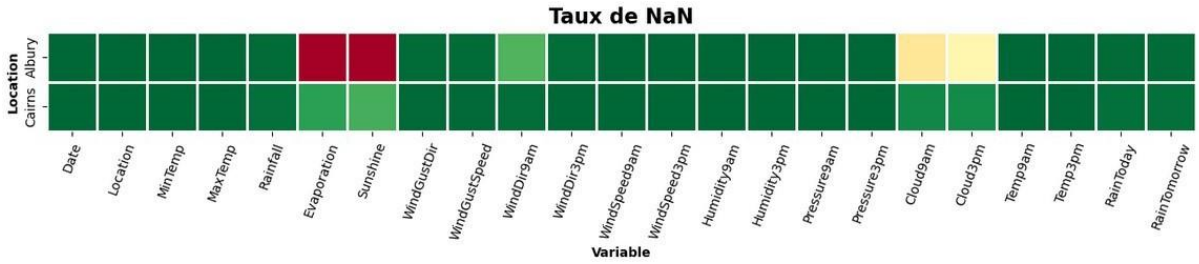


Figure 13. Taux de NaN par variable pour les stations météorologiques Albury et Cairns (rouge = 100%).

Comme il y a beaucoup de dates (3040) pour cette grandeur, nous allons faire une moyenne mensuelle pour `MaxTemp`, et afficher cette moyenne mensuelle au cours du temps.

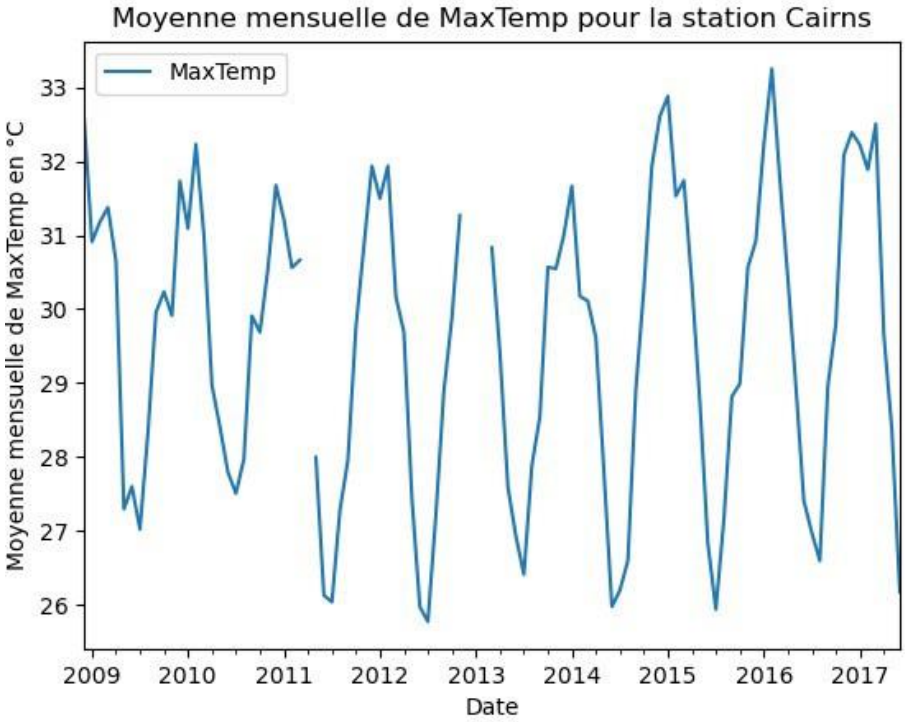


Figure 14. Moyenne mensuelle de `MaxTemp` pour la station météorologique Cairns.

Nous voyons apparaitre des « trous » sur ce graphique, autour du premier trimestre 2011, et à la frontière entre 2012 et 2013. Ceci est très surprenant : `MaxTemp` ne contient pas de NaN pour la station Cairns. Vérifions dans le tableau des valeurs moyennées.

2011-03-31	30.664516
2011-04-30	NaN
2011-05-31	27.996774

2012-12-31	NaN
2013-01-31	32.677419
2013-02-28	NaN

Figure 15. Extraits du tableau des moyennes mensuelles de `MaxTemp` pour la station météorologique Cairns.

Il y a 3 intervalles de NaN en avril 2011, en décembre 2012 et en février 2013 qui ont été créés lors du processus de moyenne mensuelle. Cela signifie que pour ces dates, il n'y a aucune entrée dans le *dataset* original. C'est préoccupant, et cela nécessite des investigations supplémentaires autour de ce que nous appellerons désormais des trous chronologiques. Combien y en a-t-il ? Comment sont-ils distribués ?

2.5.2. Analyse des trous chronologiques

Un décompte pour la station Cairns effectué sur la base des dates enregistrées donne le résultat suivant :

```
Pour la station Cairns:
Première date enregistrée: 2008-12-01
Dernière date enregistrée: 2017-06-25
Nombre de dates mesurées: 3040
Nombre total de jours entre le début et la fin: 3128
Il manque donc 88 jours pour cette station.
```

Figure 16. Calcul du nombre de jours manquants pour la station météorologique Cairns.

Ce résultat est indépendant de la nature de la grandeur mesurée puisqu'il se base uniquement sur les dates du tableau. Comme la série des dates ne contient pas de NaN, on en déduit que certaines dates sont totalement absentes du tableau. Ainsi, cela signifie que la station Cairns n'a pas enregistré de mesures pendant 88 jours. Qu'en est-il des autres stations ?

Table 3. Calcul du nombre de jours manquants pour d'autres stations météorologiques.

	endroit	date_debut	date_fin	n_jours_tot	n_jours_mes	n_jours_miss
0	Albury	2008-12-01	2017-06-25	3128 days	3040	88
1	BadgerysCreek	2009-01-01	2017-06-25	3097 days	3009	88
2	Cobar	2009-01-01	2017-06-25	3097 days	3009	88
3	CoffsHarbour	2009-01-01	2017-06-25	3097 days	3009	88
4	Moree	2009-01-01	2017-06-25	3097 days	3009	88
5	Newcastle	2008-12-01	2017-06-24	3127 days	3039	88
6	NorahHead	2009-01-01	2017-06-25	3097 days	3004	93
7	NorfolkIsland	2009-01-01	2017-06-25	3097 days	3009	88
8	Penrith	2008-12-01	2017-06-25	3128 days	3039	89

Étonnamment, on retrouve exactement le même nombre de jours manquants, à quelques exceptions près. Il y en a en général 88. Mais où sont-ils distribués ?

On trace l'évolution d'une grandeur (ici `WindGustSpeed`) au cours du temps pour les 49 stations, en superposant 7 courbes sur 7 graphiques différents.

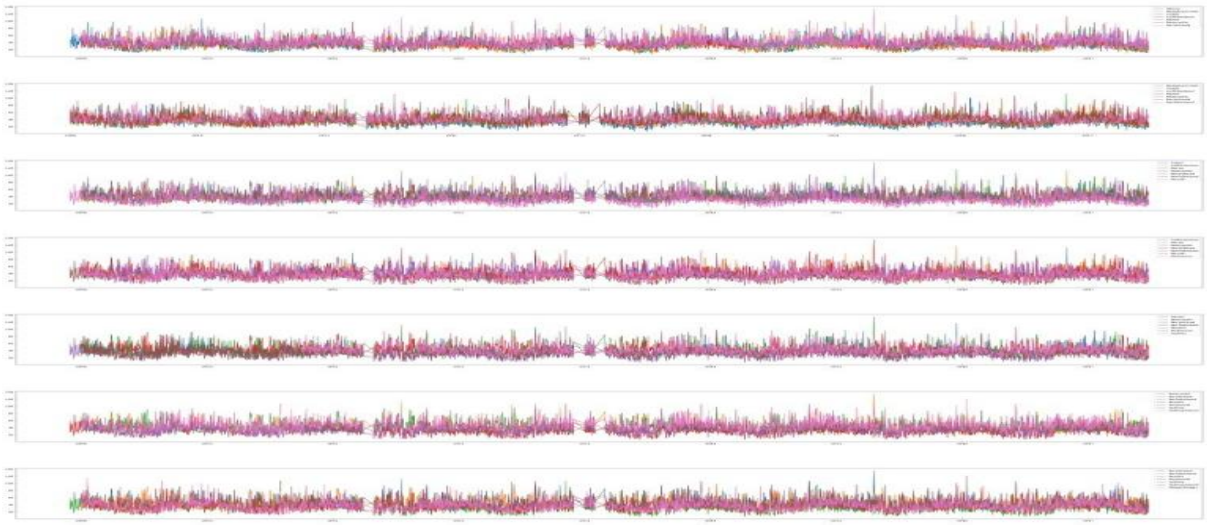


Figure 17. Évolution de `WindGustSpeed` au cours du temps pour les 49 stations météorologiques.

On voit clairement 3 zones de trous, qui ont l'air de se superposer : les dates manquantes ne sont pas distribuées aléatoirement selon les stations, elles se concentrent toutes aux mêmes endroits. Nous cherchons à déterminer les dates de ces trous.

2.5.2.1. Premier trou : avril 2011

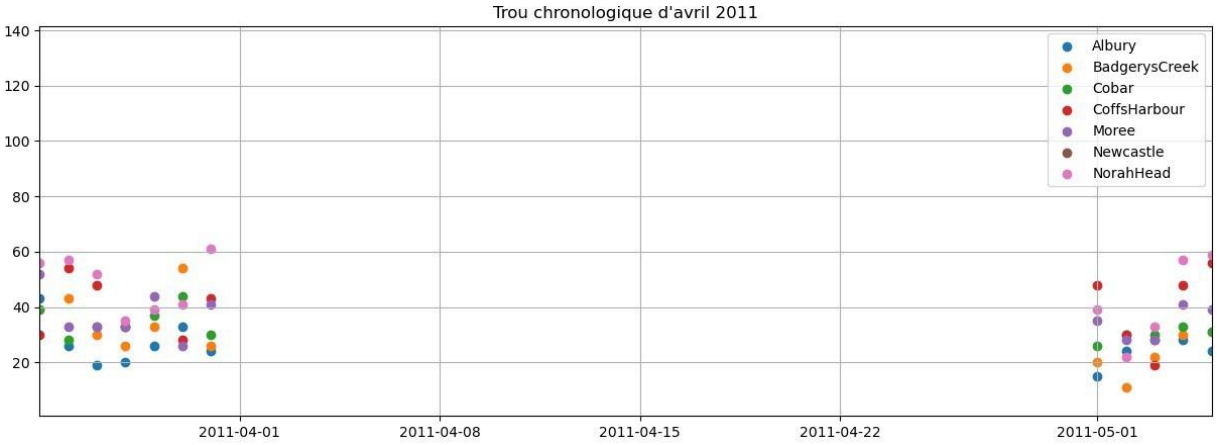


Figure 18. Trou chronologique d'avril 2011.

C'est très clair : le début et la fin des trous sont situés exactement aux mêmes dates. Les mesures s'arrêtent au premier avril 2011 pour reprendre le premier mai 2011. Au final, c'est l'intégralité du mois d'avril 2011 qui est absent du tableau.

2.5.2.2. Autres trous

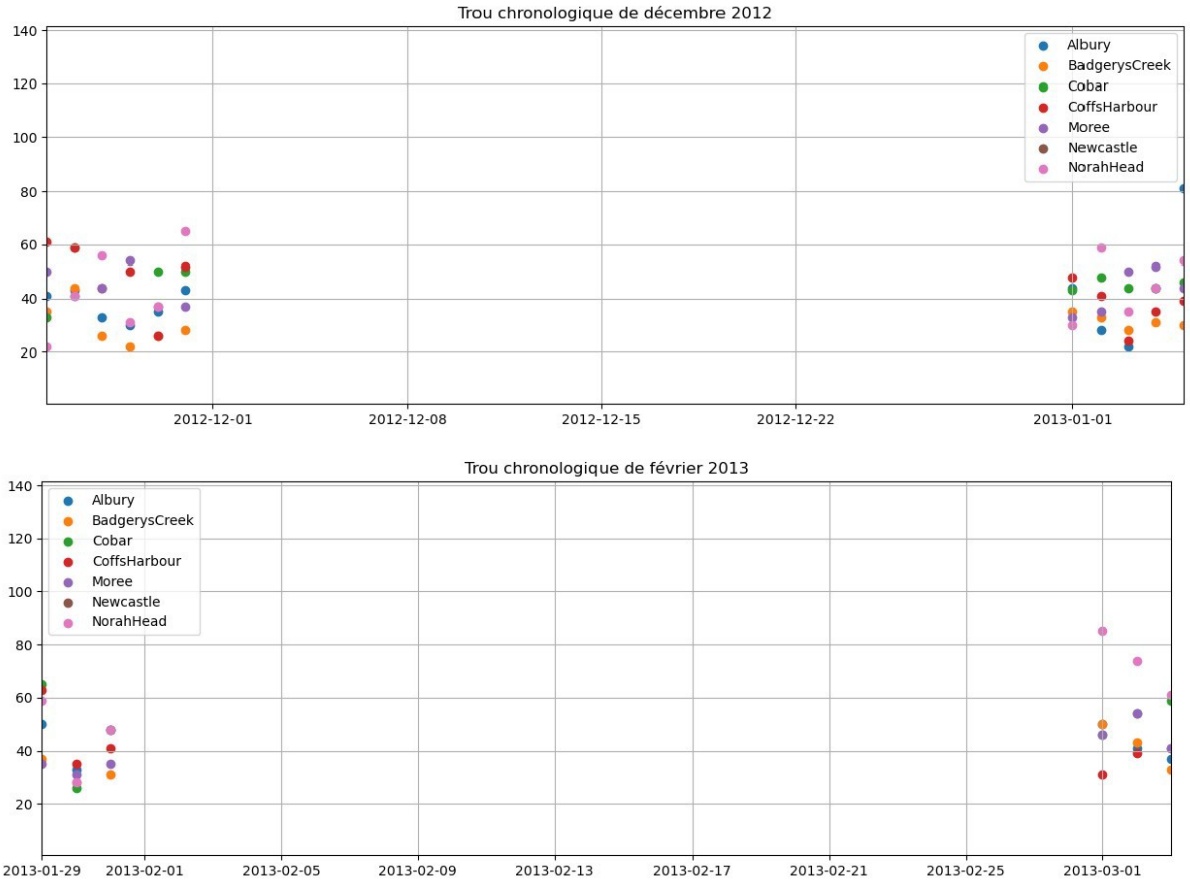


Figure 19. Trous chronologiques de décembre 2012 et février 2013.

On retrouve le même schéma : ce sont des mois pleins et entiers qui sont absents, ici décembre 2012 et février 2013.

2.5.3. Conclusion

La parfaite synchronicité des trous laisse à penser que des périodes d'interruptions de mesure ont été décidées pour l'ensemble des stations, en avril 2011, décembre 2012 et février 2013. La raison de ces interruptions est inconnue. Il est intéressant d'avoir ces informations dans le cas d'une éventuelle modélisation de séries temporelles en vue d'effectuer des projections. Faudra-t-il trouver un moyen de gérer ces valeurs manquantes ? En tout cas, nous en connaissons le nombre et la distribution.

Une question demeure : avril compte 30 jours, décembre 31, et février 2013, 28, ce qui fait un total de 89 jour manquant, et pas 88. Est-ce un problème dans le code comptant les jours manquants ? Certaines stations affichent un nombre de jours manquants égal à -1, ce qui est surprenant : on devrait avoir zéro. En tenant compte d'un éventuel problème de décalage, on retrouverait bien 89 jours.

Ces stations sont au nombre de 3 : Nhill, Katherine et Uluru, et leur mise en service à toutes les trois dates précisément du 1er mars 2013, soit après la dernière période d'interruption enregistrée. Était-ce là la raison de l'interruption générale de toutes les autres stations ?

Les données enregistrées démarrent à des dates différentes, mais se terminent toutes à la même date, ou presque : soit le 24 juin 2017, soit le 25 juin 2017.

3. Nettoyage des données

Il est temps de se préoccuper de la gestion des NaN, pour ensuite préparer les 9 tableaux déjà crées à la modélisation via les procédés de *preprocessing* et de *feature engineering*.

3.1. Quels choix effectuer ?

3.1.1. Variables numériques

3.1.1.1. Mini-introduction

Nous avons un total de 16 variables numériques à gérer :

- Un bloc de température : `MinTemp`, `MaxTemp`, `Temp9am`, `Temp3pm`
- Un bloc de vitesses de vent : `WindGustSpeed`, `WindSpeed9am`, `WindSpeed3pm`
- Un bloc humidité : `Humidity9am`, `Humidity3pm`
- Un bloc pression : Pression 9am, Pression3pm
- Un bloc couverture nuageuse : `Cloud9am`, `Cloud3pm`.
- Des mesures plus hétéroclites : `Rainfall`, `Evaporation`, `Sunshine`.

Toutes ces variables présentent une statistique cohérente : peu d'*outliers*, des valeurs maximales et minimales qui possèdent un sens physique, des distributions plausibles, nous l'avons vu précédemment. Seule la variable `Rainfall` pose problème à cause de sa distribution très hétéroclite, nous verrons plus tard que le problème se règlera de lui-même lorsque nous évoquerons le traitement des variables catégorielles `RainToday` et `RainTomorrow`.

Nous voudrons travailler sur deux modalités différentes de gestion des NaN pour ces grandeurs :

- Soit le remplacement des NaN par la moyenne par station, qui sera le meilleur estimateur des valeurs manquantes, au sens mathématique et statistique du terme, et permettra de conserver des données.
- Soit leur suppression pure et simple, si notre jeu de données est suffisant. Cette suppression s'effectuera à l'échelle des 9 tableaux correspondant au regroupement par différentes modalités de mesure. Alors qu'une suppression massive et aveugle sur le tableau d'origine aurait conduit mécaniquement à la disparition pure et simple de toutes les stations qui ne mesurent pas au moins une grandeur, cette méthode est plus sélective et permet néanmoins de conserver de l'information.

À ce stade de notre réflexion, les informations dont nous disposons ne nous permettent pas de trancher nettement en faveur de l'une ou l'autre. Nous prévoyons de tester les deux lors de la phase de modélisation. Voici un résumé des avantages et inconvénients de chaque méthode :

- **Méthode n°1 :** Replacement par la moyenne par station. Cette méthode conserve des données en remplaçant les NaN par des valeurs robustes et cohérentes, mais peut créer des biais.
- **Méthode n°2 :** Suppression des NaN. Les données finales sont de qualités, mais peuvent être en nombre insuffisant.

Détaillons maintenant notre analyse sur la méthode n°1.

Comme nous l'avons vu, il existe deux grandes catégories pour la distribution des NaN :

- Une aléatoire, où certaines valeurs enregistrées sont absentes ponctuellement ici où là, pour des raisons inconnues ;
- Une systématique, lorsque la station cesse de mesurer une grandeur pendant une période plus ou moins longue.

Exemple sur le bloc température mesurées à Melbourne :

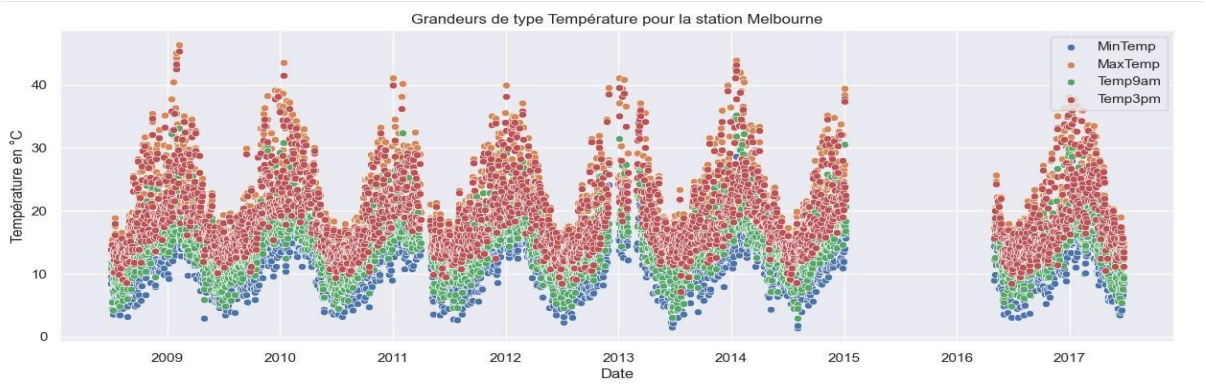


Figure 20. Grandeurs de type température pour la station météorologique Melbourne.

Nous pouvons repérer trois choses :

- Les trois trous chronologiques apparaissent sous forme de fines bandes. Ce ne sont pas des NaN.
- Une large bande de NaN pendant un an et demi environ, de janvier 2015 à avril 2016. L'appareil unique de mesure des températures servant à alimenter en données nos 4 variable du bloc température était probablement défectueux ou désactivé pendant la période.
- Les NaN ponctuels ne sont pas visibles sur cette représentation.

Nous avons tout d'abord pensé à remplacer les NaN par interpolation, au vu de la régularité des mesures, notamment leur périodicité saisonnière. Cela donnerait :

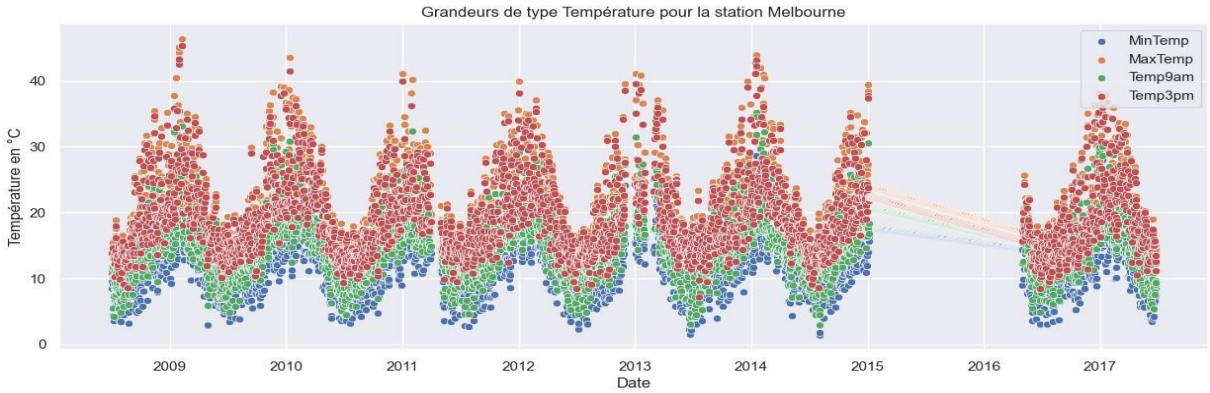


Figure 21. Grandeurs de type température pour la station météorologique Melbourne après remplacement des NaN par interpolation.

Comme prévu, les trous chronologiques ne sont pas concernés, car ce ne sont pas des NaN. En revanche, la méthode `interpolate` se contente, sans plus d'arguments, de relier la première date non mesurée à la dernière en utilisant un modèle linéaire. On voit bien sur le graphique que cette méthode n'est pas satisfaisante : l'interpolation n'est pas fidèle à ce que l'on aurait pu observer. Par manque de temps et de de connaissances, nous éliminons cette méthode au profit du remplacement par la moyenne.

Voici le résultat, toujours pour Melbourne :

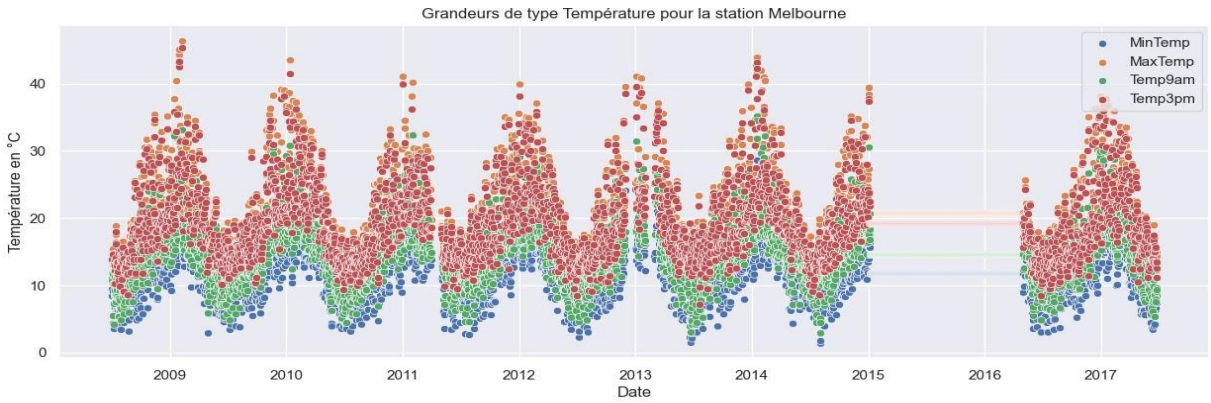


Figure 22. Grandeurs de type température pour la station météorologique Melbourne après remplacement des NaN par la moyenne.

Comme prévu, ce n'est pas l'idéal, mais cela permet de conserver toutes les autres données des lignes contenant des NaN dans les colonnes température pour cette station.

Regardons ce que ce choix implique pour une autre station et pour une autre grandeur. Nous choisissons Canberra, et l'évaporation, pour laquelle le taux de NaN est important (environ 47%).

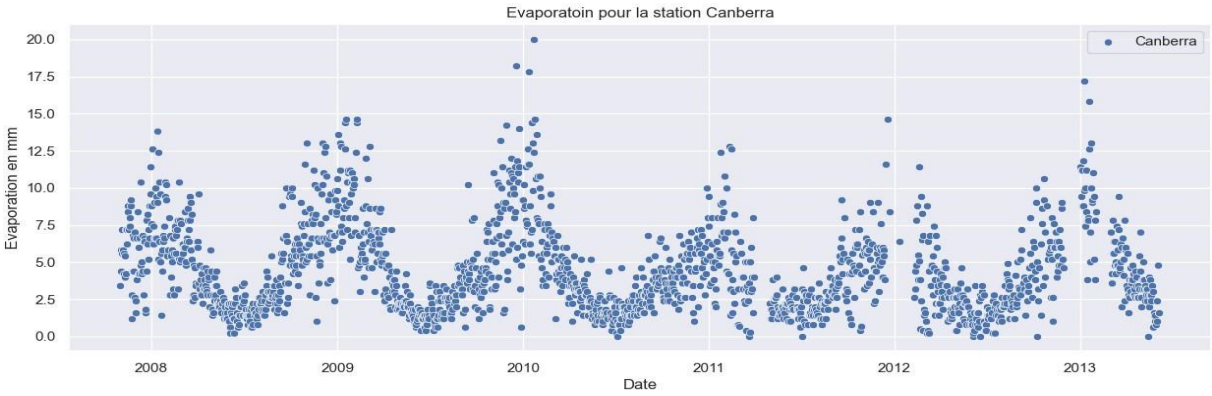


Figure 23. Distribution temporelle de `Evaporation` pour la station météorologique Canberra.

Ici les NaN aléatoires sont davantage visibles.

Remplaçons-les par la moyenne :

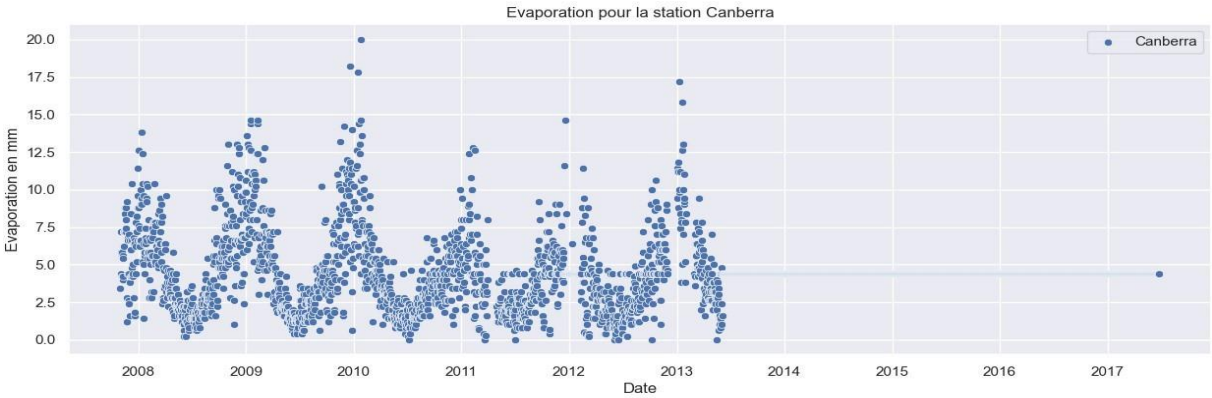


Figure 24. Distribution temporelle de `Evaporation` pour la station météorologique Canberra après remplacement des NaN par la moyenne.

Surprise ! Nous n'avions pas pris en compte la nécessité d'intégrer à l'affichage des dates fixes de début de de fin. On voit ici le résultat du remplacement. Canberra a cessé de mesurer l'évaporation à partir de mi 2013. On voit aussi « apparaitre » les NaN aléatoires entre 2011 et 2013 (N.B. : un test d'interpolation illustre graphiquement le fait que l'interpolation semble mieux fonctionner pour les NaN aléatoires que le remplacement par la moyenne, comme on le voit ici).

On comprend bien le problème désormais :

- La méthode n°1 peut conduire à de vastes remplacements comme ici dont on peut se poser la question de la pertinence.
- Le choix de la méthode pourrait donc dépendre du taux de NaN par mesure :
    - Taux faible : on remplace par la moyenne ou on supprime, les deux ayant leurs avantages et inconvénients déjà mentionnés
    - Taux élevés : on supprime les données pour éviter de trop biaiser. Le problème, c'est que les fenêtres de non-mesures de grandeurs ne sont pas simultanées et on pourrait potentiellement perdre beaucoup de données à fonctionner ainsi.

3.1.1.2. Mini-conclusion

Par manque de temps et d'expérience, nous ne pouvons pas pour l'instant nous lancer dans des modes de gestions plus sophistiqués. Nous devons avancer et faire des choix, quitte à y revenir plus tard après une itération par la phase modélisation. Nous retenons donc les deux méthodes 1 et 2, à savoir :

- Méthode n°1 : Replacement par la moyenne par station. Cette méthode conserve des données en remplaçant les NaN par des valeurs robustes et cohérentes, mais peut créer des biais.
- Méthode n°2 : Suppression des NaN. Les données finales sont de qualité, mais peuvent être en nombre insuffisant.

Remarque : les mesures de couverture nuageuse sont en réalité des catégories numériques de 0 à 9. Nous remplaçons les NaN de cette variable par la moyenne par station arrondie à l'entier le plus proche.

3.1.2. Variables catégorielles

3.1.2.1. Mini-introduction

Nous avons un total de 7 variables catégorielles à gérer :

- `Date` et `Location`, qui ne contiennent aucun NaN : aucune décision particulière n'est à prendre.
- Un bloc concernant la direction vent : `WindGustDir`, `WindDir9am`, `WindDir3pm`
- Deux variables binaires concernant la pluie : `RainToday` et `RainTomorrow`.

3.1.2.2. Bloc des directions des vents

L'analyse concernant les variables numériques précédentes reste valable : on doit choisir entre la suppression pure et simple ou bien le remplacement par la modalité la plus fréquente, avec toujours les mêmes enjeux.

Nous avons tracé pour chaque station la rose des vents, qui est un histogramme en diagramme polaire, dont voici un extrait :

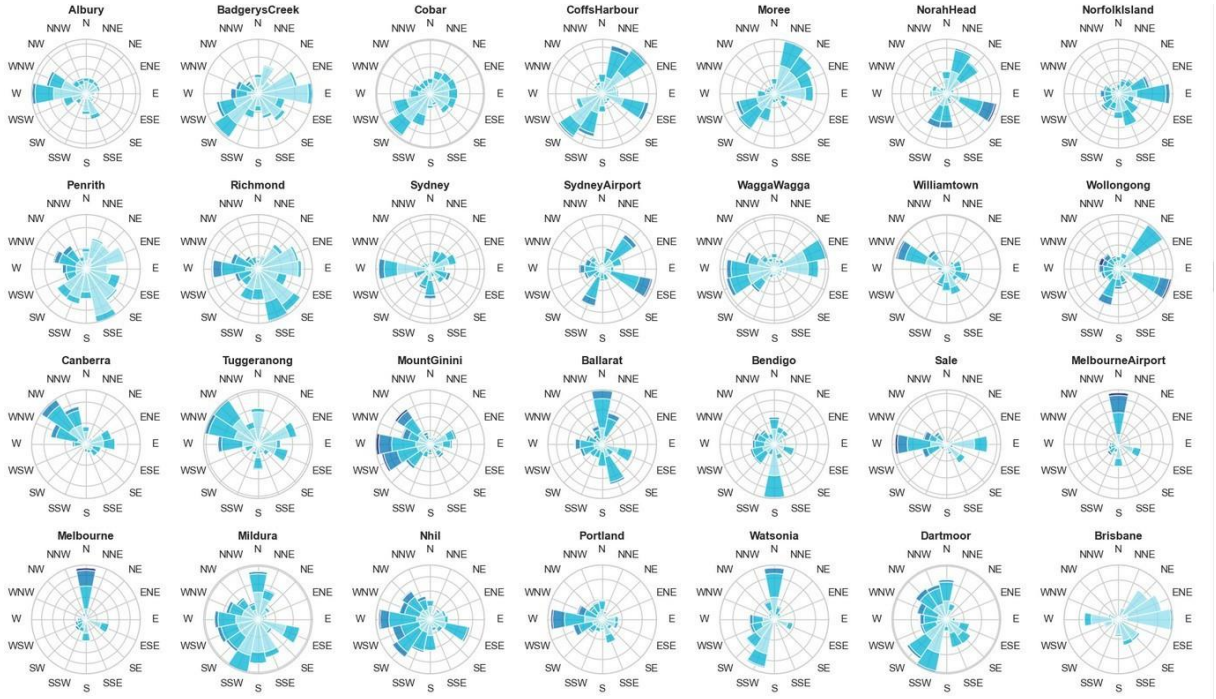


Figure 25. Roses des vents pour 28 des 49 stations météorologiques.

Dans certains cas, la méthode 1 a du sens : la modalité la plus fréquente correspond bien à l'idée que l'on se fait d'une moyenne dans le cas spécifique de la direction des vents (exemples : Albury, Melbourne, Cobar, Richmond). Dans d'autres, on aboutira à des résultats plus problématiques (exemples : CoffsHarbour, WaggaWagga).

Une autre information supplémentaire aidant à prendre une décision pour ce bloc se trouve dans l'analyse des corrélations avec la variable cible `RainTomorrow`. Les directions du vent semblent peu corrélées à la cible, ainsi, il peut être judicieux de remplacer les NaN par la modalité la plus fréquente - quitte à mettre des valeurs problématiques - afin de préserver les autres valeurs (qui corrèlent davantage que le vent) des conséquences de la suppression de la ligne entière. Nous pourrons éventuellement finir par supprimer ce bloc de vents de nos tableaux nettoyés (nous reviendrons sur le sujet spécifique du bloc des vents dans la partie *preprocessing*).

3.1.2.3. Bloc pluie

Ici, c'est plus facile : l'une des variables est notre variable cible. On supprime les NaN correspondants, on ne peut pas se permettre de biaiser notre modèle à ce niveau-là. D'autant que le taux de NaN pour `RainTomorrow` et relativement faible, on ne perd ainsi pas beaucoup de données.

On s'aperçoit que ce choix impacte directement les colonnes `RainToday` et `Rainfall`. En effet, `RainTomorrow` contient la valeur stockée dans `RainToday` du lendemain. Si l'instrument de mesure de la quantité de pluie tombée ne fonctionne pas, il n'y aura rien dans `Rainfall`. Et rien non plus dans `RainToday`, qui se remplit automatiquement (selon la règle : Yes si les précipitations dépassent 1 mm sur la journée, No dans le cas contraire), et donc rien non plus dans le `RainTomorrow` correspondant. Ainsi, en cas de période prolongée de non-mesure de la pluie, supprimer les NaN dans `RainTomorrow` règle le problème des NaN dans les autres colonnes du bloc.

Parfois, cela n'est pas suffisant, il reste en effet le cas des NaN aléatoires et des effets de bords des NaN systématiques. Pour bien comprendre, imaginons n'avoir qu'une seule ligne avec un NaN pour `RainTomorrow`. On la supprime. Mais s'il y avait un NaN ici, c'est parce qu'il y avait deux NaN dans la ligne précédente (la veille, sur `Rainfall` et `RainToday`), et eux n'ont pas été supprimés. On règle ce problème en effectuant une suppression sur `RainToday` aussi.

A l'issue du processus, il n'y a plus de NaN dans la colonne `Rainfall`, ce qui règle le problème de sa distribution compliquée dont nous parlions plus tôt.

3.1.2.4. Mini-conclusion

Nos tableaux sont prêts, nettoyés selon la méthode suivante :

- Il n'y a rien à faire pour `Date` et `Location` ;
- Toutes les variables numériques sauf `Rainfall` sont complétées par la moyenne par station ;
- Les variables catégorielles de la direction du vent sont remplacées par la modalité la plus fréquente ;
- On supprime les NaN sur `RainTomorrow` et `RainToday`
- Cela règle le problème de la gestion de la colonne `Rainfall`
- On garde la possibilité de travailler en amont sur chacun des 9 tableaux pour supprimer directement tous les NaN, ce qui est radical, mais moins que de le faire sur le tableau original, ce qui nous priverait des données de stations entières.

À titre d'exemple, sur le tableau regroupant les stations qui mesures toutes les grandeurs, l'application des méthodes donne :

- *Pour la méthode n°1 : on passe de 80 040 lignes à 77 728, soit une perte de 2,9% des données.*
- *En appliquant la méthode n°2, on conserve 56 420 lignes, soit une perte de 29 % des données.*
- *En faisant un* `dropna` *en amont sur le tableau originel, on passe de 145 460 lignes à 56 420, soit une perte de 61% des données (il est logique d'avoir le même nombre de lignes restantes entre les points 2 et 3 car un* `dropna` *sur tout le tableau élimine automatiquement toutes les stations qui ne mesurent pas l'intégralité des grandeurs).*

3.2. *Preprocessing*

3.2.1. Variables numériques

Comme vu précédemment dans la section décrivant les statistiques de ces valeurs, il sera nécessaire de procéder à une standardisation des données quantitatives, les valeurs numériques s'échelonnant sur des intervalles très différents. La pression, notamment, aboutit à des valeurs très élevées (autour de 1000), tandis que la vitesse du vent se situe typiquement autour de la dizaine ou centaine de km/h, ou encore les précipitations qui peuvent être faibles (0.5 mm).

3.2.2. Variables catégorielles

Afin de pouvoir utiliser les variables catégorielles, il va falloir les gérer :

- **Pour le bloc pluie** (`RainToday`, `RainTomorrow`) : Ce sont des variables binaires, un simple remplacement par 0 ou 1 convient.
- **Pour le bloc des vents** (3 colonnes) : chaque grandeur peut prendre 16 modalités différentes, correspondant chacune à une direction de la rose des vents. Une dichotomisation créerait donc 16 x 3 = 48 colonnes supplémentaires. A ce stade, c'est la seule méthode que nous connaissons, et nous nous interrogeons sur le poids que cette surcharge de colonne entrainerait sur la recherche de modèles. D'autant plus que les analyses préliminaires montrent que ces variables semblent peu corrélées à la variable cible. Ainsi se posent deux questions :
    - La suppression pure et simple des colonnes
    - La recherche d'une autre méthode de gestion. Un flow chart sur un article traitant de cette question nous propose 3 méthodes : *one-hot encoding*, *binary encoding* et *feature hashing encoding*. Mais à ce stade, nous n'avons pas eu le temps de creuser les investigations.

- **Pour la colonne des dates :** voir en *feature engineering* (création de colonnes dates)
- **Pour la colonne** `Location`, qui contient la ville où est implantée la station, plusieurs options s'offrent à nous :
    - Entrainer un modèle par station, ce qui évacue la question du *preprocessing*
    - Il y a 49 stations différentes, la dichotomisation via `get_dummies` créera 49 colonnes supplémentaires. On rencontre le même problème que pour la colonne des vents. Gestion par *one-hot encoding*, *binary encoding* ou *feature hashing encoding* ?
    - Ou Supprimer la colonne. On perd la dépendance avec la géographie, mais on garde quand même l'information pertinente sur les grandeurs physiques.

3.2.3. *Feature engineering*

En l'état actuel de nos connaissances, nous n'envisageons que la création de 3 colonnes supplémentaires : l'année, le mois et le jour. D'autres investigations sont à réaliser pour savoir si une métrique spécifique à la météorologie pourrait être créée.

4. Modélisation

4.1. Méthodologie

4.1.1. Phases

4.1.1.1. Phase A : travail sur neuf sous-tableaux

À l'issue de la première phase de notre travail, nous avons construit neuf sous tableaux différents, chacun correspondant à un ensemble de données non mesurée par les stations. Cela répondait à une problématique de gestion des valeurs manquantes, où nous préférerions ne pas biaiser notre jeu de données initiales. Par exemple, la station Adélaïde ne mesure pas la couverture nuageuse : ainsi, mettre une valeur dans cette colonne nous paraissait peu pertinent.

Voici un résumé visuel de l'ensemble des étapes de cette première phase de préparation des données :

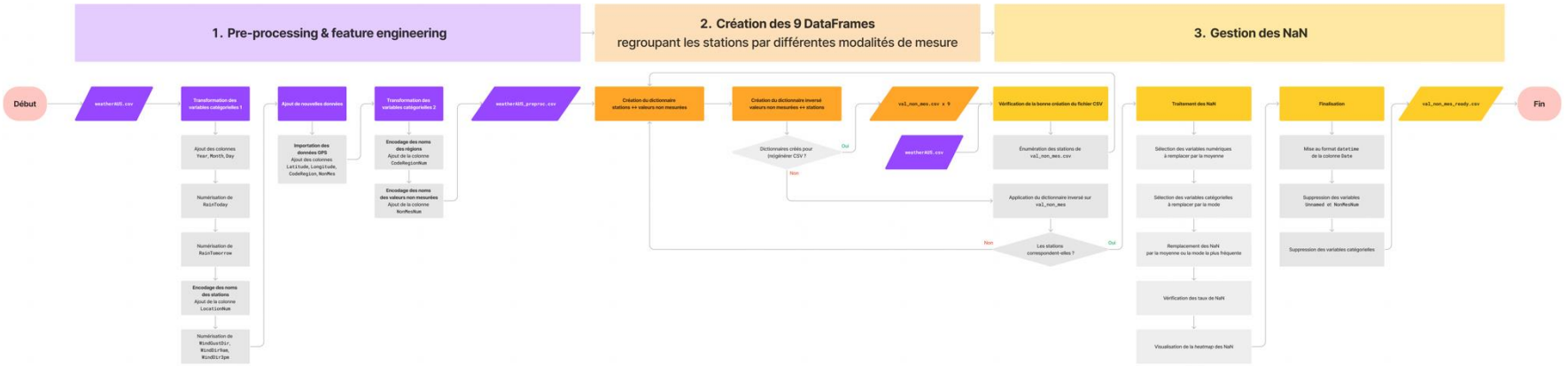


Figure 26. Premier *pipeline* de préparation des données.

4.1.1.2. Phase B : retour à un tableau unique

L'objectif pour la phase de modélisation était ensuite de se répartir le travail à trois, chacun d'entre-nous ayant en charge la modélisation de trois tableaux. Il s'est avéré que cette méthodologie était peu fructueuse pour plusieurs raisons :

- Chercher le bon modèle pour chaque tableau prend déjà beaucoup de temps. Le chercher pour trois en prend encore plus, car nous voulons choisir le modèle le plus adapté à chaque tableau.
- Notre groupe est passé de trois personnes à deux personnes en cours de travail, ce qui alourdit le travail pour les deux personnes restantes.
- Un des neuf tableaux mesure l'intégralité des grandeurs, et nous avons analysé quelles grandeurs sont les plus importantes pour le modèle. Parmi ces grandeurs, on retrouve l'humidité, la pression, la couverture nuageuse, la vitesse du vent et parfois la température. Les tableaux qui ne mesurent pas ces grandeurs donnent des résultats médiocres.

Compte tenu de toutes ces contraintes, nous avons changé de méthode en cours de route, et sommes partis sur la gestion d'un tableau entier dont nous allons rapidement vous présenter les choix de production.

Nous avons choisi, comme précédemment, de supprimer les valeurs manquantes des colonnes `RainToday` et `RainTomorrow`. Cela règle le problème de la colonne `Rainfall`.
Pour les autres valeurs manquantes, nous avons choisi de les remplacer par la valeur moyenne sur l'ensemble des tableaux pour les variables numériques. Nous avons fait de même avec les variables catégorielles, en remplaçant les valeurs manquantes par la modernité la plus fréquente.
Nous avons ajouté au tableau d'autres données, notamment les coordonnées GPS des stations météo (latitude et longitude), leur région d'appartenance, la liste concaténée des grandeurs non mesurées, et trois colonnes : une pour jour, une pour le mois, et une pour l'année.
Enfin, nous avons convertit toutes les grandeurs catégorielles en variable numériques. Nous disposons à présent d'un tableau complet prêt à être modéliser.

4.1.1.3. Phase C : choix de métrique et optimisation

Notre travail consiste en une classification binaire visant à prédire, à partir des données journalières, s'il pleuvra le lendemain ou non. En l'absence de cahier des charges qui pourrait correspondre à des souhaits particuliers de clients, nous avons dû faire un choix de métrique, tant pour l'optimisation des modèles que pour leur évaluation. Notre jeu de données étant déséquilibré, se baser sur une simple *accuracy* nous a initialement paru peu

pertinent. Ainsi, nous avons choisi d'entraîner nos modèles sur le f1 score moyen.
Après analyse une certaine quantité de matrices de confusion, nous nous sommes rendu compte que ce qui importait le plus pour nous était la précision des prédictions des classes positives et négatives. Nous avons donc fixé un seuil de 83% pour ces deux classes, ce qui correspond à cinq prédictions juste sur six. Comme nous le verrons, cela conduit parfois avoir des rappels sur la classe positive quelque peu décevants ce qui sera un critère supplémentaire pour choisir un modèle.

Notre jeu de données initiale est déséquilibré : il fait beau dans 78 % des cas, et il pleut les 22% restants. Il nous a paru important d'en tenir compte, en employant deux méthodes :

- La première consiste à s'assurer que les proportions des deux classes soient identiques dans le jeu d'entraînement et de test. Pour cela, nous indiquons dans le `train_test_split` l'argument `stratify = y`.
- La deuxième vise à ré échantillonner les données, soit par *undersampling*, soit par *oversampling*. Nous avons choisi de tester les deux méthodes afin de voir leur impact sur la modélisation et de choisir la meilleure des deux. Nous partons sur `SMOTE` pour l'*oversampling*, et sur `ClusterCentroid` pour l'*undersampling*.

Nous avons aussi normalisé nos données en utilisant la fonction `StandardScaler`.
Au final, le bilan est mitigé, certains modèles fonctionnant mieux sans rééchantillonnage (les séries temporelles et les réseaux de neurones notamment).

4.1.2. Approches

4.1.2.1. Classification : algorithmes classiques

Notre première approche a consisté à réutiliser les premiers algorithmes de classification que nous avons étudié en cours. Nous les avons classés en fonction de leur interprétabilité, du plus simple au plus complexe, à savoir :

- LogReg (*logistic regression*)
- DT (*decision tree*)
- RF (*random forest*)
- KNN (*k-nearest neighbors*)
- SVM (*support vector machines*)

4.1.2.1.1. Premier essai

Nous avons structuré notre premier travail autour de la régression logistique.
Nous avons appliqué un `GridSearchCV` pour trouver les meilleurs paramètres de ce modèle.
Une fois la sauvegarde du `GridSearchCV` effectuée, nous l'utilisons pour créer le meilleur

modèle de régression logistique, qui sera entraîné deux fois : une première fois sur les données sur échantillonnées par `SMOTE`, une deuxième fois sur les données sous échantillonnées par `ClusterCentroid`. Ces deux modèles seront ensuite sauvegardés afin d'éviter d'avoir à les refaire tourner. Nous évaluons ensuite les performances des modèle grâce à une matrice de confusion et grâce à un classification report, sur les ensembles d'entraînement et de tests. L'objectif ici étant de choisir la meilleure méthode de rééchantillonnage, et de détecter un éventuel sur-apprentissage.

4.1.2.1.2. Deuxième essai

Une fois cette méthode élaborée pour un modèle, il est aisé d'automatiser la procédure pour les quatre autres modèles, afin de créer un tableau regroupant tous les résultats de tous les modèles entraînés pour chaque méthode de ré échantillonnage. Nous obtenons ainsi le tableau suivant :

Table 4. Grille des expériences pour les quatre modèles de classification.

	Modele	Sampling	Evaluation	accuracy	precision_0	recall_0	f1_0	precision_1	recall_1	f1_1	f1_macro_avg	f1_weighted_avg
0	logreg	SMOTE	train	78.9	78.4	79.7	79.0	79.4	78.0	78.7	78.9	78.9
1	logreg	SMOTE	test	79.4	92.6	80.0	85.8	52.5	77.4	62.5	74.2	80.7
0	logreg	CC	train	76.5	76.3	77.0	76.6	76.7	76.0	76.4	76.5	76.5
1	logreg	CC	test	78.8	92.2	79.5	85.4	51.5	76.4	61.5	73.4	80.1
0	dt	SMOTE	train	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
1	dt	SMOTE	test	78.5	87.8	84.1	85.9	51.4	59.0	54.9	70.4	79.0
0	dt	CC	train	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
1	dt	CC	test	63.6	90.7	59.3	71.8	35.5	78.7	48.9	60.3	66.7
0	rdf	SMOTE	train	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
1	rdf	SMOTE	test	85.8	90.6	91.3	90.9	68.5	66.6	67.6	79.2	85.7
0	rdf	CC	train	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
1	rdf	CC	test	71.5	95.2	66.8	78.5	43.0	88.1	57.8	68.1	73.9
0	knn	SMOTE	train	90.2	94.4	85.4	89.7	86.7	94.9	90.6	90.1	90.1
1	knn	SMOTE	test	80.6	91.9	82.4	86.9	54.6	74.6	63.1	75.0	81.6
0	knn	CC	train	82.0	79.3	86.5	82.8	85.2	77.4	81.1	81.9	81.9
1	knn	CC	test	81.7	91.8	84.0	87.7	56.6	73.5	64.0	75.8	82.4

Remarque : le modèle SVM a été étudié sur les petits tableaux, et ne donne généralement pas de meilleurs résultats que les autres. Nous avons estimé qu'appliquer cette méthode sur le tableau entier prendrait 8h de temps de calcul, aussi avons-nous choisi d'exclure SVM de l'analyse sur le tableau entier.

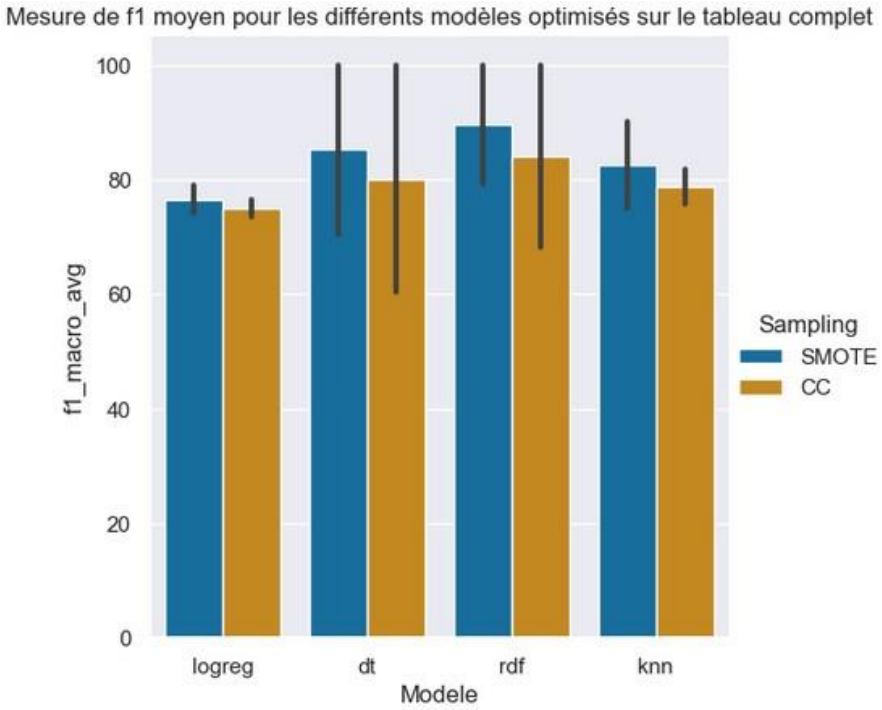


Figure 27. Moyennes des *f1-scores* des quatre modèles de classification, réparties selon les deux méthodes de rééchantillonnage.

Résultats :

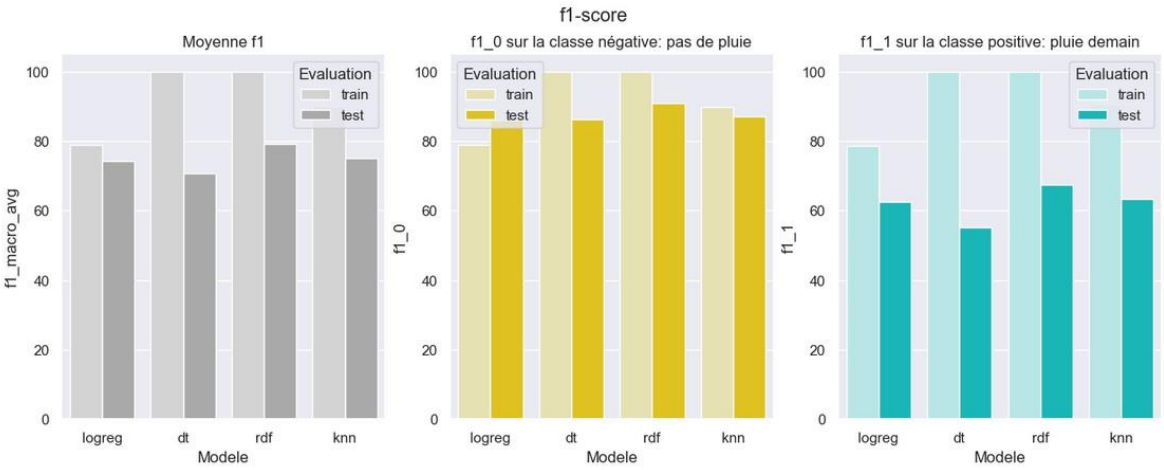


Figure 28. Moyennes des *f1-scores* des quatre modèles de classification, réparties selon les jeux d'entraînement et de test, et selon les deux classes de la variable cible.

Il apparaît de cette analyse que les meilleurs modèles sont la régression logistique et les forêts aléatoires, avec le ré échantillonnage `SMOTE`. Pour autant, ils ne donnent pas le résultat attendu de 83 % de précision sur les deux classes. On constate un certain déséquilibre : la classe positive (il pleut demain) à une précision modeste, et aux alentours de 67 %.

Nous constatons aussi un fort sur-apprentissage des modèles, le modèle des forêts aléatoires allant même jusqu'à obtenir un score de 100 % sur le jeu d'entraînement. Nous pensons à ce stade que réduire le sur-apprentissage permettrait sans doute d'améliorer les prédictions du modèle sur la classe positive. Notre prochaine étape consiste à analyser les grandeurs que le modèle considère comme pertinentes, afin d'effectuer une sélection sur les variables explicatives. Notre hypothèse est qu'enlever les variables les moins pertinentes permettrait aux modèles de gagner en précision dans une certaine mesure jusqu'à un optimum au-delà duquel continuer à enlever des variables conduirait à une baisse significative de la précision. C'est ce que nous analyserons dans la partie suivante.

Nous nous focalisons maintenant sur le modèle de régression logistique et sur le modèle des forêts aléatoires. Pour chacun des deux modèles, nous affichons la liste des variables explicatives, de la plus importante à la moins importante.

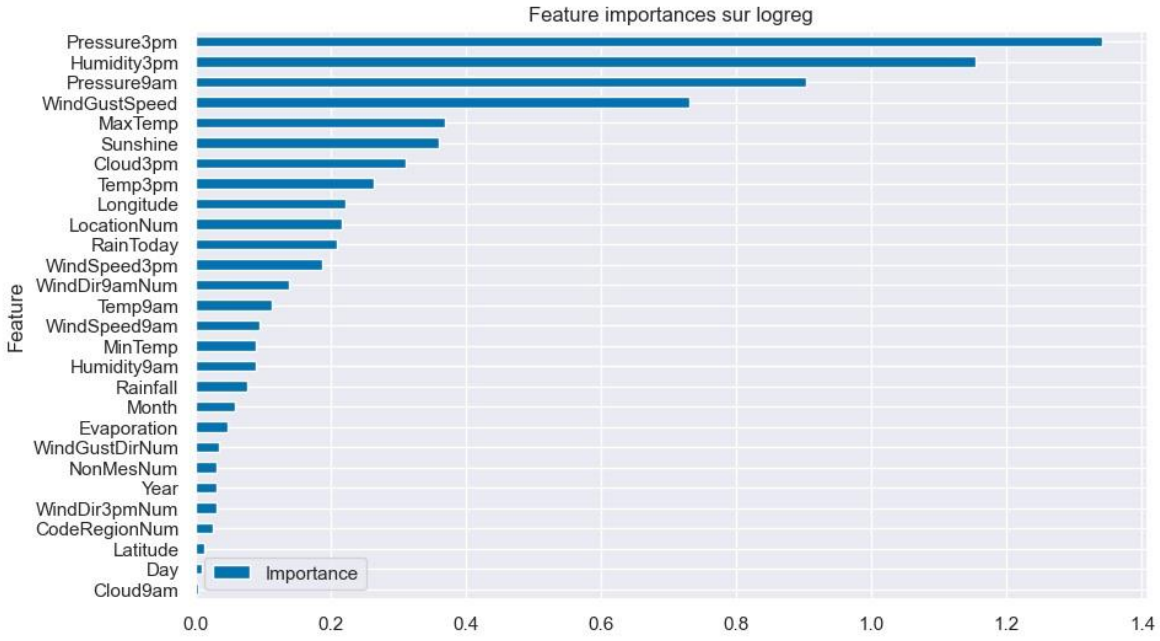


Figure 29. *Features importance* pour le modèle LogReg.

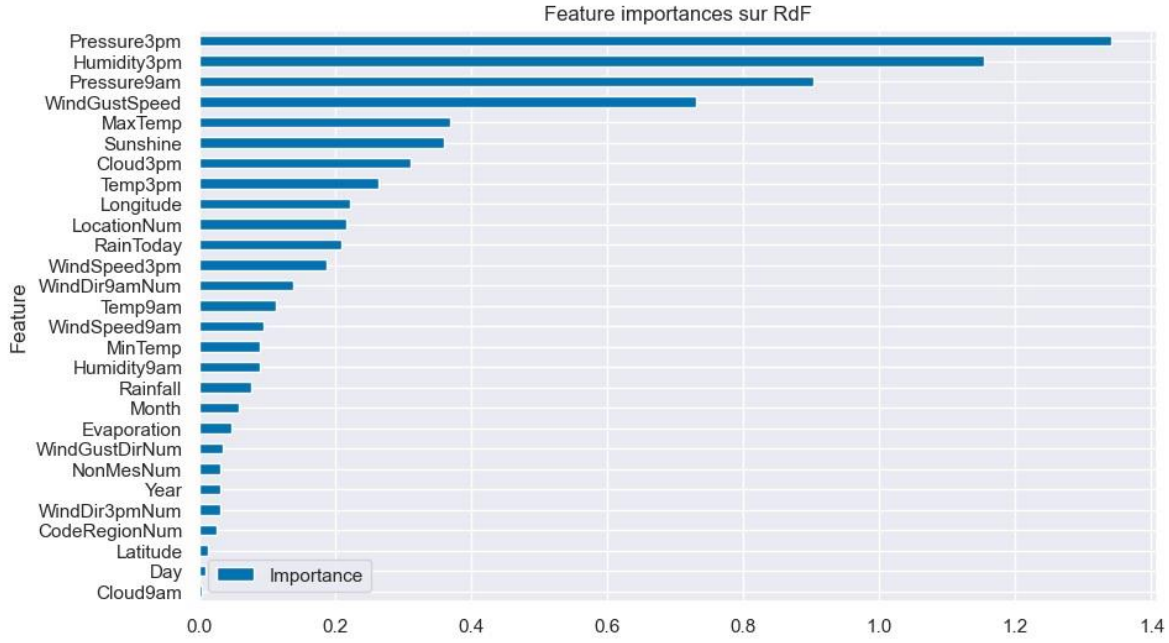


Figure 30. *Features importance* pour le modèle RF.

Nous construisons ensuite un programme qui va enlever les variables explicatives une par une, de la moins importante à la plus importante. A chaque étape, les modèles sont entraînés sur les variables restantes, et nous affichons le f1-score moyen sur le jeu de tests et le jeu d'entraînement. Nous obtenons les graphiques suivants :

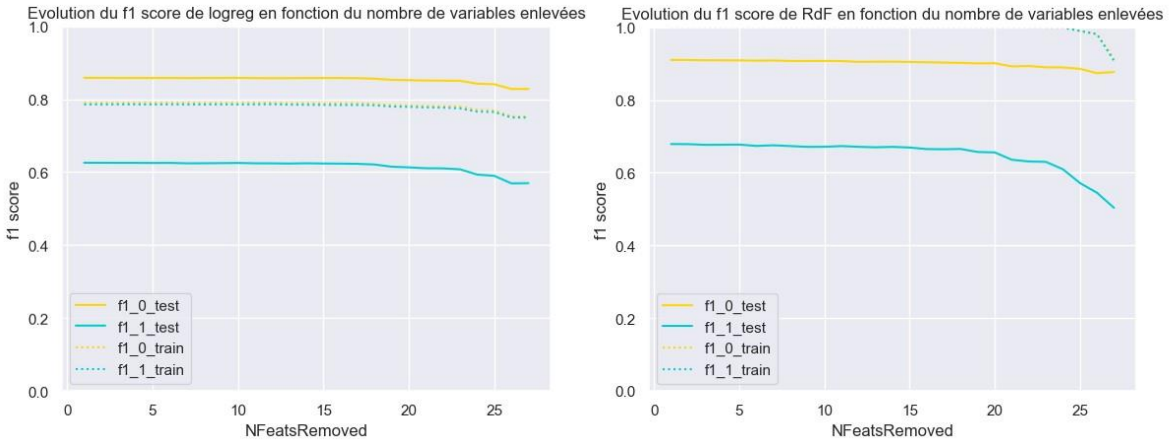


Figure 31. Évolution du *f1-score* des modèles LogReg et RF
en fonction du nombre de variables enlevées.

Notre hypothèse de départ est invalidée : de manière très surprenante, nous constatons que le score est quasiment constant, jusqu'à subir une baisse brutale lorsque nous enlevons les variables explicatives les plus importantes. De plus, cette sélection progressive ne réduit pas le sur-apprentissage. Il nous faut donc une autre approche afin de régler ce problème.

4.1.2.2. Classification : algorithmes des séries temporelles

Nos jeux de données étant constitués d'observations pour plusieurs variables dans le temps, chacun d'entre eux peut être considéré comme une série temporelle multivariée.

Cependant, les algorithmes de `scikit-learn` supposent que les données soient structurées sous forme de tableau, et que chaque colonne représente des variables i.i.d. (indépendantes et identiquement distribuées).

Les séries temporelles ne remplissent pas ces critères car elles sont, par nature, caractérisées par la corrélation entre des observations proches dans le temps.

Afin de procéder aux modélisations de nos jeux de données en tant que séries temporelles, nous avons donc mobilisé `sktime` [1], une bibliothèque qui étend l'API `scikit-learn` aux tâches d'apprentissage automatique avec des séries temporelles. [2]

Parmi les algorithmes de classification implémentés par `sktime`, nous avons choisi trois pour cette étude :

- KNN-DTW
- TSF
- ROCKET

Enfin, avant de construire les modélisations, quelques étapes complémentaires de prétraitement des données s'imposent afin d'assurer leur compatibilité avec `sktime` et garantir la cohérence méthodologique. Ces étapes sont détaillées dans la section 4.1.2.2.4 (Prétraitement complémentaire des données).

4.1.2.2.1. KNN-DTW

La méthode KNN-DTW repose sur le couplage de l'algorithme classique KNN avec l'utilisation de la DTW (*dynamic time warping*, ou « déformation temporelle dynamique » en français) comme métrique de distance. Elle fait partie des approches de classification des séries temporelles basées sur les **distances** (*distance-based approaches*). [3]

La DTW permet de mesurer la similarité entre deux séries temporelles pouvant évoluer à des vitesses différentes. [4] Elle calcule la correspondance optimale entre les séries en les déformant de manière non linéaire dans le temps (Figure 32).

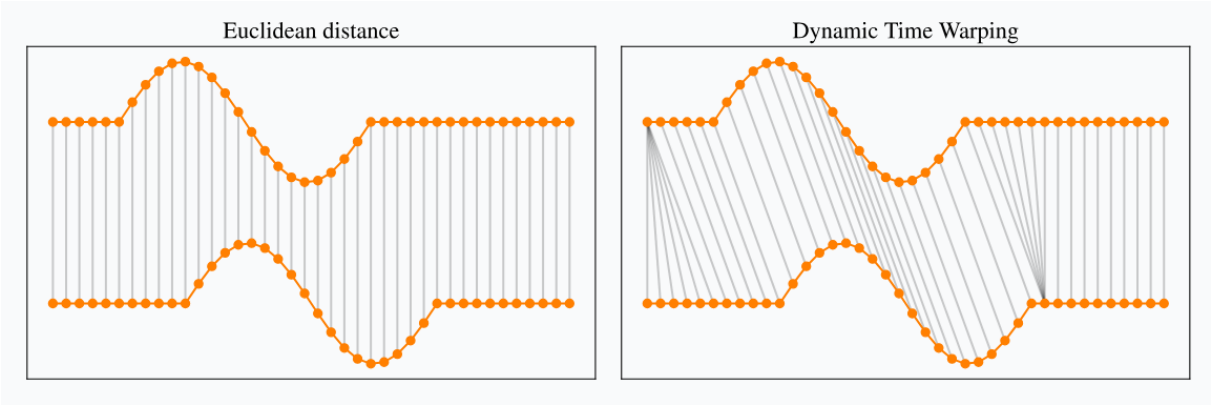


Figure 32. Comparaison entre la distance euclidienne, utilisée dans l'implémentation classique de la KNN, et la DTW, utilisée dans l'adaptation de la KNN aux séries temporelles. [5]

La déclinaison 1NN de la méthode est notamment utilisée comme *benchmark* grâce à sa simplicité et sa robustesse : « *DTW in conjunction with 1-NN has been the gold standard for time series classification for the past decade and is almost always used as a comparative algorithm in benchmarking studies* ». [3]

Toutefois, la méthode peut être gourmande en calcul et fournit peu d'informations sur les critères d'affectation de classe.

Elle est implémentée dans `sktime` par le classificateur `KNeighborsTimeSeriesClassifier` du module `classification.distance_based`, configuré avec les paramètres suivants :

- `n_neighbors = 1`
- `distance = "dtw"`

4.1.2.2.2. TSF

La méthode TSF (*time series forest*) appartient à la famille des approches de classification des séries temporelles basées sur les **intervalles** (*interval-based approaches*). [3]

Il s'agit d'une adaptation aux séries temporelles de l'algorithme RF (*random forest*). [6]

La TSF utilise un arbre de décision pour chaque intervalle, les arbres de décision agrégés constituant la forêt. Chaque arbre de décision est un modèle d'apprentissage automatique qui attribue ensuite une classe à son intervalle de données. Étant donné que les arbres de décision s'entraînent sur un intervalle différent de la série temporelle globale, ils peuvent ne pas produire la même classification, d'où la nécessité du vote d'ensemble à la fin du processus. [3]

Des études expérimentales ont montré que la TSF est capable de produire de meilleurs résultats que la KNN-DTW. [7]

La méthode est implémentée dans `sktime` par le classificateur `TimeSeriesForestClassifier` du module `classification.interval_based`.

4.1.2.2.3. ROCKET

La méthode ROCKET (« *RandOm Convolutional KErnel Transform* ») s'inspire des approches de classification des séries temporelles basées sur les **dictionnaires** (*dictionary-based approaches*) [3], ainsi que des **réseaux de neurones convolutifs** (*convolutional neural networks* / CNN). [8]

La méthode consiste en deux étapes principales [8] :

- **Transformation** des séries temporelles à l'aide de noyaux de convolution aléatoires (*random convolutional kernels*)
- **Entraînement** d'un classificateur linéaire sur les séries temporelles transformées

Contrairement aux noyaux des CNN, ceux de ROCKET sont extrêmement variés car leurs paramètres (p. ex. longueur, dilatation, *padding*) font l'objet d'un échantillonnage aléatoire (qui reste toutefois contrôlé). [9] Cette diversité, couplée au très grand nombre de noyaux (10 000 par défaut), permet à la méthode de capturer des motifs à des fréquences et des échelles contrastées. [8]

La transformation des séries temporelles se fait par leur convolution avec les noyaux, générant une *feature map* qui est ensuite agrégée en deux *features* par noyau (soit 20 000 *features* au total) [8] :

- La **valeur maximale** (*maximum value* ; similaire au *global max pooling* des CNN), et
- La **proportion de valeurs positives** (*ppv*).

Une fois transformées, les séries temporelles sont passées à un classificateur linéaire dont le choix dépend de la taille du jeu d'entraînement [9] :

- S'il y a moins de 20 000 lignes, il est conseillé d'utiliser un classificateur de régression *ridge*. En effet, plus l'écart entre le nombre de *features* (ici, 20 000) et la taille du jeu d'entraînement est grand, plus la régularisation est importante, et la capacité d'un classificateur de régression *ridge* à exploiter la validation croisée généralisée permet de trouver un paramètre de régularisation adapté assez rapidement.
- S'il y a plus de 20 000 lignes, il est conseillé de se tourner plutôt vers un classificateur de régression logistique entraîné à l'aide de la descente de gradient stochastique (*stochastic gradient descent* / SGD). En effet, l'entraînement de ce classificateur est relativement rapide pour un coût de mémoire fixe.

Des études ayant comparé ROCKET aux algorithmes concurrents, notamment les CNN, ont démontré que ROCKET peut atteindre le même niveau de *accuracy* en une fraction du temps. [9]

La méthode est implémentée dans `sktime` par le transformeur `Rocket` du module `transformations.panel.rocket`.

Pour les classificateurs linéaires, nous avons utilisé les suivants, issus du module `linear_model` de `sklearn` :

- Jeux d'entraînement avec moins de 20 000 lignes : `RidgeClassifierCV(alphas = np.logspace(-3, 3, 10))`
- Jeux d'entraînement avec plus de 20 000 lignes : `SGDClassifier(loss = "log_loss")`

4.1.2.2.4. Prétraitement complémentaire des données

4.1.2.2.4.1. Tri

Pour le bon fonctionnement de `sktime`, et comme dans toute analyse de séries temporelles, il est impératif que les données soient classées par ordre chronologique.

Or, lors de leur chargement, les données sont d'abord classées selon la variable `Location` avant de l'être selon la variable `Date`. Cela n'a aucune incidence sur les jeux qui ne sont composés que d'une seule station météorologique (p. ex. les notebooks 1.X.X), mais pour les autres, cela pourrait provoquer des dysfonctionnements.

Afin d'éliminer ce risque, à chaque chargement des données dans un DataFrame pandas, nous les avons indexés par la variable `Date` (`index_col = 1`) et avons ajouté un tri selon cet index (`.sort_index()`).

Cette opération est systématiquement appliquée à partir des notebooks 2.X.X.

4.1.2.2.4.2. Découpage

Afin de préserver l'ordre chronologique des données lors de leur découpage en jeux d'entraînement et de test, nous avons utilisé le validateur croisé `TimeSeriesSplit` du module `model_selection` de `sklearn`.

`TimeSeriesSplit` est une variation adaptée aux séries temporelles de l'itérateur de validation croisée `KFold` de la même bibliothèque.

Bien que ces objets soient principalement destinés au découpage des données lors d'un processus de validation croisée, ils peuvent également être appliqués au jeu d'origine pour créer les jeux d'entraînement et de test. D'ailleurs, l'itérateur `train_test_split`, qui est habituellement utilisé à cette étape, n'est autre qu'un *wrapper* autour de l'itérateur `ShuffleSplit`.

Contrairement aux autres itérateurs, `TimeSeriesSplit` génère des jeux d'entraînement successifs qui sont des surensembles (*supersets*) de ceux qui les précèdent. Par ailleurs, les jeux d'entraînement sont systématiquement plus anciens que les jeux de test. Ce comportement permet de prendre en compte la particularité structurelle des séries temporelles (Figure 33).

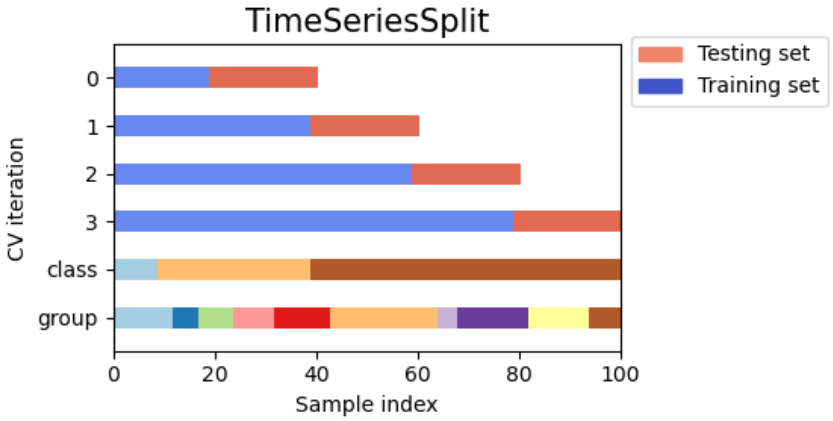


Figure 33. Découpage d'un jeu de données en jeux d'entraînement et de test par `TimeSeriesSplit` sur plusieurs itérations de validation croisée. [10]

4.1.2.2.4.3. Conversion

Bien qu'il soit globalement facile à utiliser, `sktime` comporte une contrainte importante : les données doivent d'abord être converties en un format compatible. [11]

Pour KNN-DTW et TSF, nous avons effectué cette conversion avec la fonction artisanale `numpyfy`. Elle prend en argument les jeux d'entraînement et de test (`X_train`, `X_test`, `y_train`, `y_test`) et les transforme en *arrays* NumPy de 3 dimensions, définies comme suit :

- Dimension 1 : nombre d'**instances** de séries temporelles. Pour cette étude, dont l'unité temporelle est le jour, nous considérons 1 jour comme 1 instance.
- Dimension 2 : nombre de **variables explicatives** par instance de série temporelle.
- Dimension 3 : nombre de **points temporels** observés par instance de série temporelle. Pour cette étude, comme l'unité temporelle est le jour, chaque instance de série temporelle correspond à 1 seul point temporel.

Pour TSF, il est nécessaire en plus de transformer les jeux de données multivariées en univariées. Afin de réaliser cette opération, nous avons utilisé le transformeur `ColumnConcatenator` du module `transformations.panel.compose` de `sktime`.

Enfin, pour ROCKET, nous ne sommes pas parvenus à faire fonctionner la méthode avec `numpyfy`. Nous avons donc repensé la structure des données afin de les convertir en format `"Panel"`. [11] Cela implique le changement de paradigme suivant :

- Chaque date (ligne) est toujours considérée comme une **instance** (un échantillon).
- En revanche, pour chaque instance, toutes les **variables explicatives** sont concaténées en une seule.
- Chaque concaténation est considérée comme une série de **points temporels**, disposés *horizontalement*.

Cette formulation n'est pas intuitive : en effet, il est plus naturel de définir une série temporelle *verticalement*, en suivant l'index chronologique.

Elle est néanmoins utile car elle permet d'adapter le jeu de données à la méthode ROCKET.

Elle s'inspire de l'analyse des images en tant que séries temporelles par le déroulement de leur contour sur un hypothétique axe temporel, permettant par la suite d'y appliquer des méthodes réservées aux séries temporelles (Figure 34).

Pour implémenter cette transformation, nous avons utilisé `tslearn`, une autre bibliothèque Python qui fournit des outils d'apprentissage automatique pour l'analyse de séries temporelles. [12]

Il propose, dans son module `utils`, la fonction `to_sktime_dataset`, qui permet de convertir un jeu de données vers un format de *scitype* `"Panel"` et de *mtype* `"nested_univ"` tels que définis par `sktime`. [11] Ce format correspond également à celui du jeu de données dans la démonstration de ROCKET par `sktime`. [13]

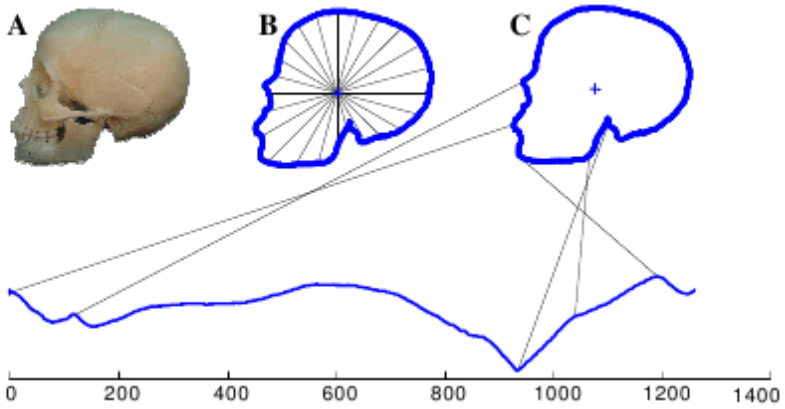


Figure 34. Conversion du contour d'une image en une série temporelle pour analyse. [14]

4.1.2.3. Méthodes d'ensemble

Une autre façon de réduire le sur-apprentissage consiste à utiliser un ensemble de classifieurs faibles et à agréger leurs résultats. Nous étudierons ainsi par la suite des techniques de Voting, Stacking et XGBoost.

4.1.2.3.1. Voting

Nous utilisons les quatre modèles optimisés précédents, à savoir la régression logistique, les arbres de décision, les forêts aléatoires et la méthode des plus proches voisins. Deux choix s'offrent ensuite à nous : le Hard Voting où chaque classifieur vote pour une classe et c'est la majorité qui l'emporte, ou le Soft Voting, où chaque classifieur rend les probabilités d'appartenance à chaque classe, et le Voting choisit celui qui renvoie la plus forte probabilité : c'est celui-là qui déterminera la classe d'appartenance finale.

4.1.2.3.2. Stacking

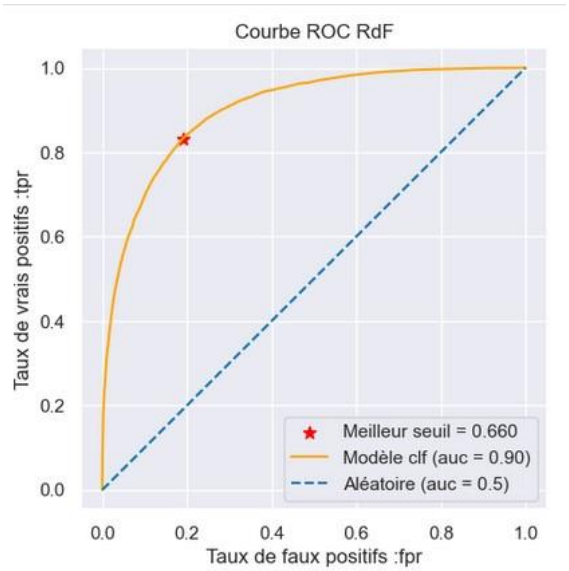
C'est la même idée que la méthode précédente, mais on envoie en sortie un autre classifieur. Nous avons choisi la régression logistique comme estimateur final.

4.1.2.3.3. XGBoost

Nous avons aussi essayé d'implémenter l'algorithme XGBoost, mais sa complexité en termes d'hyperparamètres et le peu de temps à notre disposition ne nous ont pas permis d'optimiser pleinement les fonctionnalités de cet algorithme. Pourtant comme on va le voir, les résultats sont plutôt satisfaisants.

4.1.2.3.4. Ajustement des prédictions par recherche du meilleur seuil

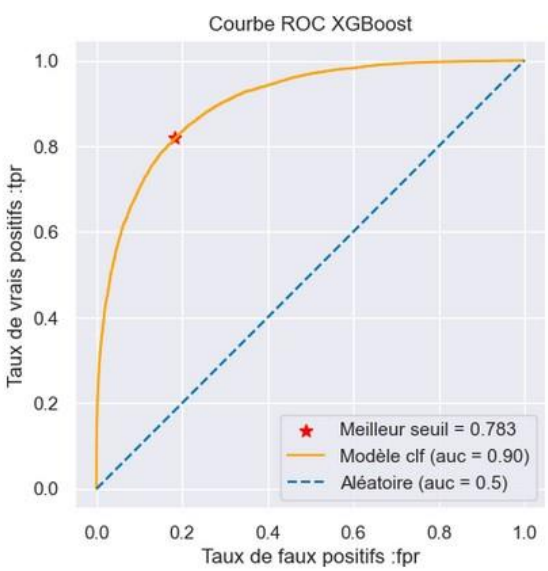
Malgré toutes ces recherches, un déséquilibre persiste entre les classes positives et négatives. N'ayant pas pu réduire le sur-apprentissage autant que nous l'espérions, nous choisissons de jouer sur le seuil (qui par défaut vaut 0,5) permettant d'affecter un résultat à une classe donnée. Pour cela nous utilisant la méthode ROC pour chacun de nos classifieurs, et cherchons le seuil optimal, c'est-à-dire celui qui se rapproche le plus du point de coordonnées (0, 1) dans le diagramme. Ensuite, nous implémentons ce seuil aux classifieurs déjà obtenu. Nous illustrons les résultats sur deux de nos meilleurs modèles : Les forêts aléatoires et XGBoost.



Report RdF

	precision	recall	f1-score	support
0	0.87	0.97	0.92	27397
1	0.80	0.50	0.61	7800
accuracy			0.86	35197
macro avg	0.84	0.73	0.76	35197
weighted avg	0.86	0.86	0.85	35197

[[26454 943]
[3938 3862]]



Report XGBoost

	precision	recall	f1-score	support
0	0.84	0.99	0.91	27397
1	0.91	0.32	0.47	7800
accuracy			0.84	35197
macro avg	0.87	0.65	0.69	35197
weighted avg	0.85	0.84	0.81	35197

[[27159 238]
[5321 2479]]

Figure 35. Courbes ROC pour les modèles RF et XGBoost.

À ce stade, trois modèles retiennent désormais notre attention :

- Les forêts aléatoires optimisées et seuillées
- Le Voting Hard avec les modèles optimisés
- XGBoost seuillé et légèrement optimisé

Table 5. Résultats du Hard Voting.

	precision	recall	f1-score	support
0	0.86	0.97	0.91	27397
1	0.81	0.45	0.58	7800
accuracy			0.86	35197
macro avg	0.84	0.71	0.75	35197
weighted avg	0.85	0.86	0.84	35197

4.1.2.4. Réseaux de neurones

Nous avons également créé un classifieur binaire en utilisant un réseau de neurones denses. L'architecture de base comprenait :

- Une couche d'entrée de dimension 28 (une par variable explicative)
- Trois couches cachées comprenant 16 neurones chacune avec fonction d'activation `relu`,
- Une couche de sortie avec un neurone et une sigmoïde.

Nous avons ajouté deux callbacks :

- Un `EarlyStopping`,
- Un `ReduceLROnPlateau`

Pour l'entrainement du modèle, nous choisissons l'entropie croisée binaire comme fonction de perte, et réglons le *learning rate* initial de l'optimiseur `Adam` sur 10^{-5}. Le modèle s'entraine sur 100 *epochs*, avec des *batchs* de taille 32. La métrique suivie est l'*accuracy*.

Nous avons testé différentes configurations :

- La mise à l'échelle des données : la standardisation via `StandardScaler` donne de meilleurs résultats que la normalisation `MinMaxScaler`.
- La fonction d'activation `relu` donne de meilleurs résultats que la fonction tangente hyperbolique.
- Augmenter le nombre de neurones par couches donne des résultats légèrement meilleurs, jusqu'à atteindre un plafond au-delà duquel il n'est plus nécessaire de l'augmenter.
- Ajouter une quatrième couche cachée supplémentaire n'améliore pas non plus les résultats.
- Rééquilibrer les classes en amont avec `SMOTE`, ce qui affaiblit les résultats
- Créer une fonction de perte personnalisée basée sur l'entropie croisée binaire, où des résultats erronés sur la classe positive seraient davantage pénalisés, afin de tenter de réduire l'écart de performances entre la précision de la classe négative, toujours supérieure à la précision de la classe positive.
- Diminuer la taille des *batchs*, ce qui améliore la précision mais augmente le temps de calcul.

Le résultat de ces expériences nous donne l'architecture finale, sans rééquilibrage `SMOTE`, avec `StandardScaler`, 3 couches cachées à 128 neurones, et la fonction de pertes `binary_crossentropy`.

4.2. Résultats

Table 6 présente nos modèles et les scores tirés de leurs rapports de classification.

Dans un souci de comparabilité et de focalisation, nous nous limitons ici aux 19 expériences réalisées sur le tableau unique `model_weatherAUS.csv` et ne présentons pas l'intégralité des 71 expériences réalisées sur l'ensemble des 10 tableaux.

Table 6. Synthèse des rapports de classification triée par ordre décroissant selon l'*accuracy* (**acc** dans la table : les noms des métriques dans les en-têtes ont été raccourcis pour faciliter la mise en page).

Nº	Algorithme	Variation	acc	prec_0	prec_1	rec_0	rec_1	f1_0	f1_1
1	Hard Voting optimisé	`StandardScaler` + `SMOTE`	0,86	0,86	0,81	0,97	0,45	0,91	0,58
2	RF	`StandardScaler` + `SMOTE` + ROC	0,86	0,87	0,80	0,97	0,50	0,92	0,61
3	XGBoost	`StandardScaler` + `SMOTE` + ROC	0,85	0,84	0,88	0,99	0,35	0,91	0,50
4	DNN	`StandardScaler`	0,85	0,88	0,73	0,94	0,54	0,91	0,62
5	Stacking optimisé	`StandardScaler` + `SMOTE` + ROC	0,84	0,83	0,91	0,99	0,29	0,90	0,44
6	TSF	sort + TSS + `SMOTE`	0,83	0,84	0,75	0,96	0,37	0,90	0,49
7	TSF	sort + TSS + `StandardScaler` + `SMOTE`	0,83	0,87	0,64	0,91	0,54	0,89	0,59
8	RF	`StandardScaler` + `SMOTE`	0,83	0,89	0,67	0,88	0,70	0,88	0,68
9	Soft Voting optimisé	`StandardScaler` + `SMOTE` + ROC	0,82	0,82	0,94	1,00	0,22	0,90	0,36
10	TSF	sort + TSS	0,82	0,83	0,79	0,98	0,30	0,89	0,43
11	TSF	sort + TSS	0,82	0,82	0,79	0,98	0,30	0,89	0,43
12	TSF	sort + TSS + `StandardScaler`	0,82	0,83	0,77	0,97	0,31	0,89	0,44
13	LogReg	`StandardScaler` + `SMOTE`	0,82	0,91	0,63	0,84	0,77	0,87	0,70
14	ROCKET	sort + TSS + `tslearn`	0,80	0,80	0,92	1,00	0,15	0,89	0,26
15	ROCKET	sort + TSS + `MinMaxScaler` + `tslearn`	0,80	0,80	0,89	0,99	0,15	0,89	0,26
16	KNN-DTW	sort + TSS + `StandardScaler`	0,78	0,85	0,51	0,86	0,49	0,86	0,50
17	KNN-DTW	sort + TSS + `MinMaxScaler`	0,78	0,85	0,51	0,87	0,47	0,86	0,49
18	KNN-DTW	sort + TSS + `StandardScaler` + `RandomUnderSampler`	0,70	0,89	0,41	0,70	0,71	0,79	0,52
19	ROCKET	sort + TSS + `StandardScaler` + `RandomUnderSampler` + `tslearn`	0,25	0,99	0,23	0,03	1,00	0,05	0,38

L'application souple du seuil de 83% à l'*accuracy*, en admettant un écart de 1%, permet de séparer la table en deux parties :

- Partie A : nº 1 à 13
- Partie B : nº 14 à 19

La partie B, qui est exclusivement composée d'algorithmes issus de la **classification des séries temporelles,** peut être éliminée compte tenu des écarts extrêmement importants (> 80%) dans les scores de *recall* (pour ROCKET), ainsi que l'éloignement non négligeable (≥ 5%) des scores d'*accuracy* par rapport au seuil de 83% (pour KNN-DTW).

Dans la partie A, nous observons que les **méthodes d'ensemble** (Voting, XGBoost, Stacking) sont très bien classées et semblent bien fonctionner sur notre jeu de données. Nous remarquons toutefois un écart important entre les deux implémentations du Voting, avec un meilleur équilibre des scores pour le Hard Voting (nº 1) comparé au Soft Voting (nº 9) : l'écart entre les scores de *precision* du premier est de 5% contre 12% pour le second, tandis que l'écart entre les scores de *recall* du premier est de 52% contre 78% pour le second.

Les approches basées sur les **arbres de décision** (RF et TSF) affichent également des performances très correctes. L'algorithme RF semble grandement bénéficier de l'optimisation du seuil (nº 2 versus nº 8) : cela fait augmenter son score de *precision* pour la classe 1 de 13% et réduit de 15% l'écart entre ses deux scores de *precision*. Ses scores de *recall* subissent toutefois un déséquilibre, l'écart augmentant de 18% à 47%. Pour les deux meilleures implémentations de l'algorithme TSF, celle sans mise à l'échelle (nº 6) a des scores de *precision* plus équilibrés (écart de 9%) par rapport à celle avec mise à l'échelle (nº 7) (écart de 23%). En revanche, la situation s'inverse pour les scores de *recall* : l'écart est de 59% pour le premier contre 37% pour le second.

Les approches basées sur les **distances** (LogReg et KNN-DTW), quant à elles, ne semblent pas particulièrement adaptées à notre jeu de données. Après exclusion des trois implémentations de KNN-DTW (nº 16, nº 17, nº 18), il reste LogReg (nº 13), qui a obtenu des scores corrects, mais pas exceptionnels en comparaison aux autres algorithmes que nous présentons ici. Ses scores de *precision* présentent d'ailleurs un déséquilibre (écart de 28%).

Enfin, pour compléter cette description des résultats, nous abordons notre algorithme basé sur les **réseaux de neurones** (DNN) (nº 4). Comme LogReg, Il affiche des bons scores dans l'ensemble, sans en proposer les meilleurs. Nous avons tenté d'apporter de nombreuses améliorations à l'architecture du réseau (nombre de perceptrons), aux fonctions de perte (personnalisation) et aux autres mécanismes de contrôle (*callbacks*), mais les scores plafonnent et présentent au passage un déséquilibre au niveau du *recall* (écart de 40%).

4.3. Discussion

4.3.1. Interprétation des résultats

Comme indiqué dans la section précédente, nos meilleurs scores viennent des **méthodes d'ensemble.**

Pour rappel, notre implémentation du Voting réunit et fait voter les quatre modèles optimisés suivants :

- LogReg
- DT
- RF
- KNN

Notre implémentation du Stacking, qui affiche également de belles performances, est construite sur ces mêmes modèles de base, suivie de la désignation de LogReg comme estimateur final.

Le **pouvoir** de ces approches résident en leur capacité à combiner des modèles très différents : elles sont en mesure de tirer profit des forces individuelles de ces derniers, tout en maîtrisant leurs faiblesses respectives. Cette diversité, que nous avons essayé de respecter au mieux dans nos implémentations (en choisissant des modèles aussi variés que possible), est un atout face notre jeu de données. En effet, la complexité, le déséquilibre, les dimensions et l'hétérogénéité de ce dernier mettent à mal les modèles classiques, qui ne parviennent pas à en capturer les caractéristiques plurielles et la structure complexe sous-jacente. [15]

Cependant, malgré leurs bons scores, ces modèles comportent aussi des **inconvénients.** Les deux principaux sont les suivants :

- **Ils sont coûteux en ressources de calcul.** [16] À titre d'exemple, notre implémentation du Voting revient à faire quatre modélisations, ce qui représente une opération conséquente compte tenu des dimensions de notre jeu de données.
- **Leur complexité se traduit par une interprétabilité plus faible.** [17] Dans le domaine de l'apprentissage automatique, la question de l'interprétabilité des modèles est loin d'être anodine. En fonction du secteur d'activité, des parties prenantes et d'autres éléments de contexte, elle peut même être centrale et contraindre le choix de modèle (souvent au détriment de la performance, laquelle augmente généralement avec la sophistication d'un modèle). Le plus souvent, le besoin ou le désir de développer la confiance des usagers dans un modèle et ses prédictions incite à en préférer un avec une bonne interprétabilité, quitte à faire des concessions sur la performance. [18] L'environnement réglementaire influe également sur les parties prenantes : le RGPD donne, aux personnes dont les données sont traitées, le droit à une explication des décisions algorithmiques et le droit d'être informées. [19] Enfin, le fait de proposer des interprétations et des explications des algorithmes souvent perçus ou caractérisés comme des « boîtes noires » ont, en soi, une valeur sociale et éthique. [20]

Il n'est donc pas évident de préférer les méthodes d'ensemble aux autres approches que nous avons également testées, notamment celles basées sur les **arbres de décision,** car ces dernières proposent des scores qui restent corrects, tout en étant plus facilement interprétables.

Afin de nous assurer d'avoir les meilleurs scores possibles en main, nous avons effectué diverses **optimisations** (mise à l'échelle, rééquilibrage, optimisation du seuil). Ces dernières fonctionnent très bien sur la plupart des algorithmes que nous avons implémentés. En revanche, elles semblent produire l'**effet contraire** sur ceux issus de la **classification des séries temporelles.** Cela est peut-être dû à notre emploi d'**approches naïves** au sein des étapes suivantes de prétraitement des données :

- **Mise à l'échelle.** Il existe de nombreuses méthodes de normalisation des séries temporelles, chacune adaptée à des problématiques différentes. [21] Par manque de temps et d'expérience, nous nous sommes limités aux méthodes proposées par `scikit-learn` (`StandardScaler`, `MinMaxScaler`). Par ailleurs, bien que la normalisation soit généralement nécessaire dans l'étude des séries temporelles, elle peut aussi détruire des informations et contribuer à la dégradation des analyses. [22]
- **Rééquilibrage.** Notre approche ne tient pas compte de la difficulté inhérente à la classification des séries temporelles déséquilibrées, une thématique de recherche qui se révèle assez riche et complexe. [23] En effet, la dimensionnalité élevée et la forte corrélation inter-variable qui caractérisent ces ensembles de données ne sont pas prises en compte par les méthodes classiques proposées par `imbalanced-learn` (`SMOTE`, `RandomUnderSampler`). [24]

Compte tenu de tous les éléments exposés ci-dessus, nous avons décidé de retenir **l'algorithme nº 2 (RF avec `StandardScaler` + `SMOTE` + ROC)** comme modèle d'apprentissage automatique pour notre projet de classification binaire. Même s'il se classe légèrement en dessous de l'algorithme nº 1 (Hard Voting optimisé avec `StandardScaler` + `SMOTE`) au niveau des scores, il est plus facile à interpréter et à expliquer, ce qui en fait un choix global plus équilibré, et meilleur, selon nous.

4.3.2. Remise en contexte

La ***numerical weather prediction*** / **NWP** (« prévision numérique du temps » / PNT en français) utilise des modèles mathématiques de l'atmosphère et des océans pour prédire le temps en fonction des conditions météorologiques actuelles. [25] Son histoire remonte à 1922, mais ce n'est qu'à partir de 1950, avec l'apparition des premiers ordinateurs, qu'elle a réellement commencé à se développer. [26]

Bien que la NWP reste encore aujourd'hui l'approche standard en prédiction météorologique, les météorologues s'intéressent de plus en plus à l'apprentissage machine / profond, certains ayant commencé à intégrer ces approches dans leurs flux de travail. [27] D'autres spécialistes restent sceptiques et réticents, reprochant à l'apprentissage automatique /

profond un manque d'explicabilité (notamment dans le cas des réseaux de neurones profonds), ainsi que l'absence de prise en compte des contraintes physiques. [28] Toutefois, cela n'empêche pas la croissance de l'appétit des météorologues pour les deux approches [29], avec une accélération depuis février 2022. [27]

Parmi les deux, l'**apprentissage profond** est de plus en plus plébiscité (certains travaux estiment que l'apprentissage automatique est moins adapté aux données géospatiales [30]). En novembre 2023, Google DeepMind a publié un article dans la revue *Science* qui compare les performances de GraphCast, son modèle de prédiction météorologique basé sur les *graph neural networks*, à celles de l'Integrated Forecasting System (IFS), le modèle NWP du European Centre for Medium-Range Weather Forecasts (ECMWF) qui était considéré jusqu'à présent comme l'étalon-or de la discipline. [31] GraphCast a distancé de loin l'IFS en matière d'*accuracy* et de vitesse. L'ECMWF a reconnu que les modèles d'apprentissage profond sont devenus tellement performants qu'ils sont désormais incontournables en météorologie, et l'organisme a même déjà commencé à utiliser GraphCast. [32] Ils expérimentent également avec d'autres modèles de pointe, notamment Pangu-Weather de Huawei [33] (basé sur les *3D neural networks*) et FourCastNet de NVIDIA [34] (basé sur les *Fourier-based neural networks*). [27]

Nous avons également tenté d'appliquer l'apprentissage profond à notre problématique de projet avec notre modèle DNN, mais les résultats ne sont pas à la hauteur de nos espérances. Cela ne veut pas dire que la méthode n'est pas adaptée (d'autres chercheurs ayant implémenté des réseaux de neurones simples ont formulé cette conclusion hâtivement [28]), mais qu'il faut trouver la bonne architecture et les bons paramètres, ce qui correspond au véritable défi à la maîtrise du domaine complexe qu'est l'apprentissage profond.

5. Conclusion et perspectives

Au final, nous choisissons les forêts aléatoires (Random Forest) comme meilleur modèle. Ses performances en termes de prédiction nous semblent acceptables dans l'optique d'un déploiement aux particuliers souhaitant connaître le temps qu'il fera le lendemain, avec une bonne prédiction 6 jours sur 7. Son interprétabilité est meilleure que les autres méthodes (réseau de neurones, méthodes d'ensemble).

Il reste un déséquilibre entre la précision des deux classes (87% de précision pour la classe négative, 80% pour la classe positive), mais ce choix de modèle nous permet d'avoir un meilleur rappel sur la classe positive (qui reste malgré tout en deçà de nos attentes, à 50%).

Nous tombons toujours sur le même problème, quel que soit notre modèle : le déséquilibre de prédiction entre les deux classes, la classe positive étant souvent la moins bonne des deux, même en rééquilibrant le jeu initial avec `SMOTE`, ou en jouant sur le seuil de classification, ou encore en personnalisant la fonction de pertes dans le cas d'un réseau de neurones.

Cela nous évoque deux hypothèses :

- Les phénomènes météorologiques sont par essence de nature chaotique, et il ne peut y avoir que des corrélations limitées entre les caractéristiques physiques de l'atmosphère un jour donné et celles du jour suivant. Cela semble peu probable au vu des récentes avancées en *deep learning* évoquées en termes de prédiction météo.
- La qualité de nos données, notamment en ce qui concerne les choix que nous avons fait en termes de *preprocessing*.

En effet, l'approche consistant à remplacer les valeurs manquantes pour les grandeurs qui ne sont jamais mesurée par certaines stations par la valeur moyenne de ces grandeurs sur le reste du tableau nous paraît quelque peu brutale.
Nous aimerions pouvoir gérer ça plus finement, et plusieurs pistes s'offrent à nous :

- La première concerne des grandeurs dont la mesure s'interrompt plus ou moins aléatoirement au cours de la période étudiée. Pour ces valeurs manquantes, il pourrait être envisagé de les remplacer en faisant de l'interpolation temporelle, à partir des données déjà présentes.
- La deuxième concerne les grandeurs non mesurées. Nous avons réalisé quelques modèles simples de régression linéaire sur les variables numériques, comme la pression, l'humidité où la température. Pour certaines d'entre elles, les résultats sont très bons, et on pourrait envisager entraîner un modèle par grandeur, et les utiliser pour prévoir les valeurs manquantes, afin d'avoir un tableau avec des variables explicatives de meilleure qualité. Ce travail semble très long, puisqu'il implique de rechercher un modèle par grandeur, et de ce que les optimiser. Une piste serait de travailler sur un tableau où nous supprimerions brutalement toutes les valeurs manquantes pour entrainer les modèles de régression linéaire, pour ensuite les appliquer au tableau compléter.
- Supprimer des données, voire des variables.

Le modèle XGBoost semble prometteur : une très rapide optimisation donne déjà de très bons résultats, mais le temps nous a manqué pour nous plonger pleinement dans la complexité de ce modèle.

L'analyse des features importances sur le modèle des forêts aléatoire montre que les grandeurs physiques mesurées à 15h sont les plus déterminantes pour prévoir la pluie le lendemain. Les stations mesurent ces grandeurs en continu tout au long de la journée, il pourrait être intéressant de suggérer d'enregistrer ces grandeurs à 18h ou 21h et d'effectuer nos prévisions dessus.

Finalement, nous pourrions chercher dans la littérature s'il n'existe pas d'indicateurs plus pertinents que nous pourrions construire à partir des grandeurs déjà disponibles.

6. Références

[1] « Welcome to sktime — sktime documentation ». Consulté le : 22 novembre 2023. [En ligne]. Disponible sur : https://www.sktime.net/en/stable/

[2] A. Amidon, « sktime: a Unified Python Library for Time Series Machine Learning », Medium. Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://towardsdatascience.com/sktime-a-unified-python-library-for-time-series-machine-learning-3c103c139a55

[3] T. R. Dinger, Y. Chang, R. Pavuluri, S. Subramanian, « What is time series classification? », IBM Developer. Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://developer.ibm.com/what-is-time-series-classification

[4] « Dynamic time warping », *Wikipedia*. 19 novembre 2023. Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://en.wikipedia.org/w/index.php?title=Dynamic_time_warping&oldid=1185828343

[5] R. Tavenard, « An introduction to Dynamic Time Warping ». 2021. [En ligne]. Disponible sur : https://rtavenar.github.io/blog/dtw.html

[6] A. Amidon, « A Brief Survey of Time Series Classification Algorithms », Medium. Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://towardsdatascience.com/a-brief-introduction-to-time-series-classification-algorithms-7b4284d31b97

[7] H. Deng, G. Runger, E. Tuv, et M. Vladimir, « A time series forest for classification and feature extraction », *Information Sciences*, vol. 239, p. 142-153, août 2013, doi : 10.1016/j.ins.2013.02.030.

[8] A. Amidon, « ROCKET: Fast and Accurate Time Series Classification », Medium. Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://pub.towardsai.net/rocket-fast-and-accurate-time-series-classification-f54923ad0ac9

[9] A. Dempster, F. Petitjean, et G. I. Webb, « ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels », *Data Min Knowl Disc*, vol. 34, nº 5, p. 1454-1495, sept. 2020, doi : 10.1007/s10618-020-00701-z.

[10] « 3.1. Cross-validation: evaluating estimator performance », scikit-learn. Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://scikit-learn/stable/modules/cross_validation.html

[11] « In-memory data representations and data loading — sktime documentation ». Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://www.sktime.net/en/stable/examples/AA_datatypes_and_datasets.html

[12] « tslearn's documentation — tslearn 0.6.2 documentation ». Consulté le : 30

novembre 2023. [En ligne]. Disponible sur : https://tslearn.readthedocs.io/en/stable/

[13] « Demo of ROCKET transform — sktime documentation ». Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://www.sktime.net/en/latest/examples/transformation/rocket.html

[14] « Converting images into time series for data mining ». Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://izbicki.me/blog/converting-images-into-time-series-for-data-mining.html

[15] X. Dong, Z. Yu, W. Cao, Y. Shi, et Q. Ma, « A survey on ensemble learning », *Front. Comput. Sci.*, vol. 14, nº 2, p. 241-258, avr. 2020, doi : 10.1007/s11704-019-8208-z.

[16] A. Nair, « Combine Your Machine Learning Models With Voting », Medium. Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://towardsdatascience.com/combine-your-machine-learning-models-with-voting-fa1b42790d84

[17] S. Kumar, « Use Voting Classifier to improve the performance of your ML model », Medium. Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://towardsdatascience.com/use-voting-classifier-to-improve-the-performance-of-your-ml-model-805345f9de0e

[18] R. Marcinkevičs et J. E. Vogt, « Interpretable and explainable machine learning: A methods-centric overview with concrete examples », *WIREs Data Mining and Knowledge Discovery*, vol. 13, nº 3, p. e1493, 2023, doi : 10.1002/widm.1493.

[19] P. Voigt et A. Von Dem Bussche, *The EU General Data Protection Regulation (GDPR)*. Cham: Springer International Publishing, 2017. doi : 10.1007/978-3-319-57959-7.

[20] S. Wachter, B. Mittelstadt, et C. Russell, « Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR », *Harv. J. L. & Tech.*, vol. 31, p. 841, 2018 2017.

[21] F. T. Lima et V. M. A. Souza, « A Large Comparison of Normalization Methods on Time Series », *Big Data Research*, vol. 34, p. 100407, nov. 2023, doi : 10.1016/j.bdr.2023.100407.

[22] M. Łuczak, « Combining raw and normalized data in multivariate time series classification with dynamic time warping », *J. Intell. Fuzzy Syst.*, vol. 34, nº 1, p. 373-380, janv. 2018, doi : 10.3233/JIFS-171393.

[23] H. Cao, X.-L. Li, D. Y.-K. Woon, et S.-K. Ng, « Integrated Oversampling for Imbalanced Time Series Classification », *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, nº 12, p. 2809-2822, déc. 2013, doi : 10.1109/TKDE.2013.37.

[24] T. Zhu, C. Luo, Z. Zhang, J. Li, S. Ren, et Y. Zeng, « Minority oversampling for imbalanced time series classification », *Knowledge-Based Systems*, vol. 247, p.

108764, juill. 2022, doi : 10.1016/j.knosys.2022.108764.

[25] « Numerical weather prediction », *Wikipedia*. 15 novembre 2023. Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://en.wikipedia.org/w/index.php?title=Numerical_weather_prediction&oldid=1185267885

[26] P. Lynch, « The origins of computer weather prediction and climate modeling », *Journal of Computational Physics*, vol. 227, nº 7, p. 3431-3444, mars 2008, doi : 10.1016/j.jcp.2007.02.034.

[27] K. Maskell, « The rise of machine learning in weather forecasting », ECMWF. Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://www.ecmwf.int/en/about/media-centre/science-blog/2023/rise-machine-learning-weather-forecasting

[28] M. G. Schultz *et al.*, « Can deep learning beat numerical weather prediction? », *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, nº 2194, p. 20200097, févr. 2021, doi : 10.1098/rsta.2020.0097.

[29] G. Lentze, « How AI models are transforming weather forecasting: a showcase of data-driven systems », ECMWF. Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://www.ecmwf.int/en/about/media-centre/news/2023/how-ai-models-are-transforming-weather-forecasting-showcase-data

[30] M. Reichstein *et al.*, « Deep learning and process understanding for data-driven Earth system science », *Nature*, vol. 566, nº 7743, Art. nº 7743, févr. 2019, doi : 10.1038/s41586-019-0912-1.

[31] « Learning skillful medium-range global weather forecasting | Science ». Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://www.science.org/doi/10.1126/science.adi2336

[32] « Google DeepMind’s weather AI can forecast extreme weather faster and more accurately », MIT Technology Review. Consulté le : 30 novembre 2023. [En ligne]. Disponible sur : https://www.technologyreview.com/2023/11/14/1083366/google-deepminds-weather-ai-can-forecast-extreme-weather-quicker-and-more-accurately/

[33] K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, et Q. Tian, « Accurate medium-range global weather forecasting with 3D neural networks », *Nature*, vol. 619, nº 7970, Art. nº 7970, juill. 2023, doi : 10.1038/s41586-023-06185-3.

[34] J. Pathak *et al.*, « FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators ». arXiv, 22 février 2022. doi : 10.48550/arXiv.2202.11214.