

# Rapport de modélisation sur les données météorologiques Australiennes

Formation : Data Scientist (05/2023 – 04/2024)

Encadrant : Francesco MADRISOTTI

Réaliseurs : Sophie BERTHIER

Luciano LANGHI

Quyen THIEU MARCAUD

Le 08/12/2023 – Définitif

## Table des matières

1	Introduction.....	3
1.1	Méthodologie .....	3
1.2	Approches .....	3
2	Prédiction de la variable <i>RainTomorrow</i> .....	6
2.1	Rappel sur déséquilibre .....	6
2.2	Métriques .....	6
2.3	Résultats de la classification par approches « classiques » via scikit-learn.....	7
2.3.1	Modèles étudiés .....	7
2.3.2	Optimisation de métriques.....	7
2.3.3	Impact du feature enginerring.....	14
2.3.4	Seuil de probabilité.....	15
2.3.5	Interprétabilité des modèles.....	20
2.4	Deep Learning avec Keras et TensorFlow.....	28
2.4.1	DNN.....	28
2.4.2	RNN .....	39
3	Prédiction de la pluie à un horizon de temps.....	39
3.1	Objectif et méthodologie.....	39
3.2	Limite théorique.....	40
3.3	Comportement détaillé.....	41
3.4	Analyse par zone climatique .....	42
3.5	Prédictions.....	44
3.6	Interprétabilité.....	45
3.7	Conclusions .....	47
4	MaxTemp.....	47
4.1	Présentation .....	47
4.2	Résultats de la régression par approches « classiques » via scikit-learn .....	47
4.3	Séries Temporelles par SARIMAX .....	48
4.4	Deep Learning.....	50
4.4.1	RNN .....	50
5	Autres variables cibles .....	53
6	Conclusion .....	53

# 1 Introduction

## 1.1 Méthodologie

L'objectif principal est ici de prédire le mieux possible la variable '*RainTomorrow*', qui indique s'il pleuvra ou non le lendemain d'une observation donnée, pour l'une des 49 stations météo (*Location*) du jeu de données. Nous utiliserons pour cela les données issues du feature engineering que nous avons effectué lors de la première partie du projet, et allons appliquer des modèles de classification tels que Logistic Regression, Decision Tree, Random Forest ou XGBoost, mais également des modèles de Deep Learning, avec des Réseaux Neuronaux Denses (DNN) et des Réseaux Neuronaux Récurrents (RNN).

Les hyperparamètres de chaque modèle seront optimisés via des tests manuels, des GridSearch, mais aussi à l'aide de la bibliothèque Hyperopt, en cherchant à maximiser diverses métriques telles que l'accuracy, la précision, le recall, le score F1 et le ROC AUC. Nous expliquerons les enjeux portant sur le choix d'une métrique adaptée.

Nous avons également inclus des éléments visuels tels que des matrices de confusion, des graphiques de courbes ROC AUC, et des analyses des caractéristiques les plus importantes pour chaque modèle. Ces éléments permettent une compréhension approfondie de la performance de chaque modèle.

L'utilisation de SHAP (Shapley Additive exPlanations) pour évaluer la contribution de chaque variable explicative a été cruciale.

Dans les parties suivantes, nous tenterons de prédire non seulement s'il pleuvra le lendemain, mais également de voir s'il est possible de prévoir la pluie plusieurs jours à l'avance. Nous essaierons également de prédire d'autres variables, comme la température, avec une problématique de régression et de séries temporelles.

## 1.2 Approches

La phase de feature engineering nous a donné plusieurs pistes sur l'ajout de variables. De premières approches rapides de modélisation n'avaient pas mis en évidence de changements importants de performances selon variables ajoutées. Pour autant, nous ne pouvions à ce stade pas encore arbitrer sur les variables à conserver ou ajouter. Nous avons donc fait le choix d'effectuer les travaux de modélisation au sein de notre groupe de travail non pas avec les mêmes variables explicatives pour chaque personne, mais de nous laisser à chacun la liberté d'explorer différentes pistes afin de pouvoir dans un second temps mieux identifier l'impact des différentes approches. Ce choix d'explorer des pistes différentes explique que, selon les captures d'écran, les variables explicatives peuvent diverger, tout particulièrement pour le vent, ou nous avons fait d'une part un encodage OneHot à partir d'une variable qualitative, mais également d'autre part un encodage trigonométrique pour réduire le nombre de features et transformer ces variables qualitatives en variables quantitatives.

Nous avons également abordé nos problématiques de modélisation selon trois niveaux de finesse :

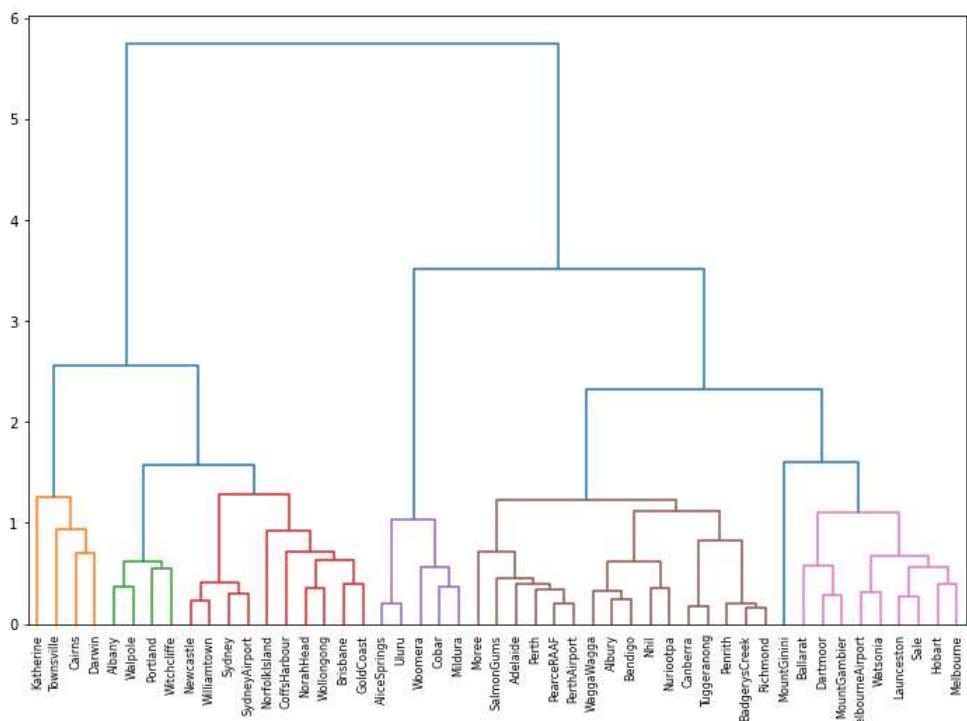
- Un niveau macro, avec des modèles portant sur l'ensemble des données australiennes du jeu de données
- Un niveau micro, où nous générerons des modèles spécifiques pour chaque *Location*
- Un niveau intermédiaire, dans lequel nous aurons clusterisé l'Australie en plusieurs zones climatiques

Le niveau macro permettra de pouvoir réaliser toutes nos prédictions avec un seul modèle.

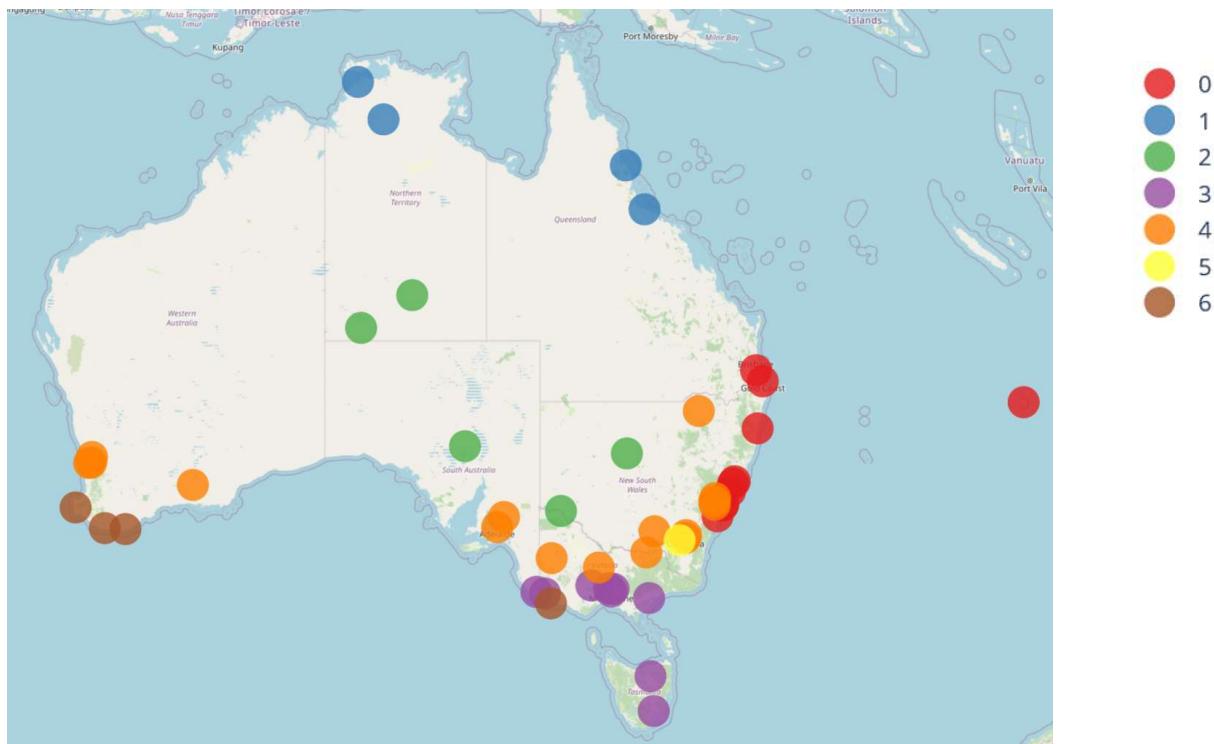
Le niveau micro rendra plus complexe l'analyse des performances puisqu'il y aura 49 modèles, mais permettra potentiellement une meilleure adaptation aux spécificités de chaque *Location*.

Le niveau intermédiaire sera un compromis entre les deux approches, puisque nous aurons 6 modèles, qui seront associés aux *Location* en fonction des spécificités climatiques communes détectées lors de la clusterisation.

La clusterisation climatique a été réalisée en reprenant la moyenne et l'écart-type de chaque feature, à l'exception des variables du vent. Les coordonnées géographiques ne sont pas utilisées. Il aurait bien sûr été possible de laisser les features du vent, mais nous avons choisi de les enlever, car la clusterisation obtenue nous amenait à pas moins de 10 clusters. De plus, elle distinguait alors des villes géographiquement très proches, telles Perth et PerthAirport, car le vent souffle beaucoup plus sur l'aéroport que sur la ville alors que les précipitations sont très similaires. Malgré tout, nous aurions bien entendu utilisé cette clusterisation utilisant le vent, mais nous avons choisi de les retirer afin d'obtenir les sept zones représentées dans les Figure 1 et Figure 2.



**Figure 1 : Diagramme de la clusterisation**



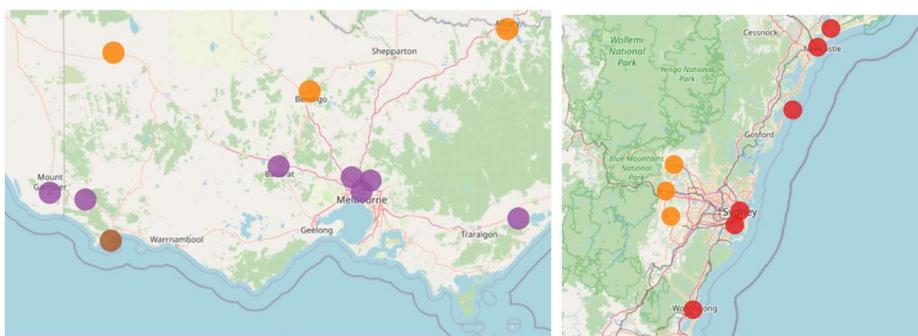
**Figure 2 : Les 7 zones climatiques**

Il saute aux yeux que malgré l'absence de coordonnées géographiques dans les features initiaux – nous les ajouterons ensuite pour nos modèles – il existe une véritable cohérence spatiale de la répartition des *Location*.

Cette cohérence nous permet de nommer chaque zone :

- Climat 0 (rouge) : Côte Est
- Climat 1 (bleu) : Nord
- Climat 2 (vert) : Centre
- Climat 3 (violet) : Sud-Est
- Climat 4 (orange) : Intermédiaire
- Climat 5 (jaune) : Mount Ginini
- Climat 6 (marron) : Côte Sud

Un point marron semble se trouver par erreur au milieu de la zone climatique violette. Il n'en est rien : Portland est bel et bien une ville côtière du sud de l'Australie, ce qui n'est pas le cas des villes alentour, qui sont plus retranchées dans les terres. De même, nous voyons à proximité de Sydney trois point orange pour les stations de Richmond, Penrith et BadgerysCreek : ces trois villes sont à l'intérieur des terres alors que Sydney et les autres villes de la zone 0 sont des villes côtières de l'est.



**Figure 3 : Cas de Portland (en marron) – Cas de Richmond/Penrith/BadgerysCreek (orange)**

Remarquons que Mount Ginini constitue un cluster à elle seule : cette station météorologique est située à 1762m d'altitude, au sommet du mont éponyme. Cette *Location* est bien plus froide que les autres et présente des caractéristiques particulières qui font d'elle un outlier climatique. Nous la conservons dans notre dataset portant sur l'ensemble de l'Australie, mais, dans un objectif de simplification de restitution des travaux, nous l'écartons des modélisations par zone climatique.

Dans un souci de synthèse et de lisibilité, nous ne déclinerons pas systématiquement par la suite les modélisations selon chacune des approches évoquées ci-dessus et nous attacherons plutôt à restituer une représentation assez variée des types d'analyses, en reprenant les résultats les plus intéressants.

Une force de notre groupe a donc été d'avoir des idées très complémentaires sur les pistes à explorer. Nous nous efforcerons dans ce rapport de conclure sur les intérêts et limites de chaque approche.

## 2 Prédiction de la variable *RainTomorrow*

### 2.1 Rappel sur déséquilibre

Il est primordial de garder à l'esprit que les deux classes de *RainTomorrow* sont déséquilibrées, puisque, sur l'ensemble du dataset, seules 22,4% des observations sont positives (= « il pleuvra demain ») et donc 77,6% sont négatives (= « il ne pleuvra pas demain »).

Ce constat nous a amené dans un premier temps à effectuer un rééquilibrage des données par oversampling. Cependant, le Tableau 1 montre que les résultats obtenus n'ont pas indiqué de différence significative sur les performances obtenues. Nous avons donc finalement conservé les données sans oversampling.

Comparaison des modèles selon l'oversampling					
Modèle	accuracy	recall	precision	f1	auc
sans oversampling	0.8642	0.5558	0.7642	0.6436	0.9003
RandomOverSampler	0.8224	0.7997	0.5694	0.6652	0.8988
SMOTE	0.8501	0.6361	0.6684	0.6518	0.8895

Tableau 1 : Comparaison des modèles selon l'oversampling

Notons que ce déséquilibre implique qu'une prédiction systématiquement négative entraînerait une accuracy de 0,776. Par conséquent, nous attendrons de nos modèles qu'ils proposent une accuracy supérieure à ce chiffre. La question de la pertinence de l'accuracy comme critère peut donc se poser.

Comme nous l'avons vu précédemment, le taux de journée pluvieuses diffère de façon significative selon les lieux. Ce ratio de 0,776 vaut donc pour uniquement pour les modèles entraînés sur l'ensemble de l'Australie.

### 2.2 Métriques

Si l'accuracy permet une compréhension simple des résultats, elle n'est pas nécessairement pertinente pour notre jeu de données, du fait du déséquilibre de *RainTomorrow*. Nous la conserverons comme indicateur global de la qualité, tout en gardant à l'esprit qu'une accuracy inférieure à 0,776 sera mauvaise.

Un faible recall correspond pour *RainTomorrow* à un nombre élevé de jours de prédictions d'absence de pluie le lendemain alors qu'il pleuvra (faux négatifs). Un modèle naïf prédisant qu'il ne pleuvra jamais aura comme vu précédemment une accuracy de 0,776, mais un recall de 0. C'est une métrique qu'il sera intéressant d'optimiser.

Une faible précision de *RainTomorrow* indiquerait que nous nous trompons souvent lorsque nous prédisons qu'il pleuvra le lendemain (faux positifs).

L'AUC-ROC, que nous noterons simplement AUC par la suite, est une métrique particulièrement intéressante dans le cadre de classe binaire déséquilibrée.

Le choix de la métrique dépendrait normalement de l'objectif à atteindre. Par exemple, pour anticiper des annulations de réservations, un hôtel touristique pourrait vouloir savoir s'il y a un risque de pluie, quitte à avoir beaucoup de faux positifs, alors qu'un agriculteur doit être certain qu'il pleuvra, même si nous devons pour cela prédire parfois à tort qu'il pleuvra.

Dans le premier cas, nous optimiserions le recall pour avoir un modèle très sensible. Dans le second cas, nous optimiserions plutôt la précision pour avoir un modèle avec une grande spécificité.

Voyons maintenant en pratique les différences obtenues selon les métriques optimisées pour chaque modèle.

## 2.3 Résultats de la classification par approches « classiques » via scikit-learn

### 2.3.1 Modèles étudiés

Nous regardons ici des modèles générés avec une approche de machine learning qui n'est pas du deep learning : nous exploitons simplement la variable *RainTomorrow* de chaque observation, indépendamment de la date de l'observation.

(Ajouter résultats du KNN, GradientBoosting, MLP? Bof, déjà bien chargé)

### 2.3.2 Optimisation de métriques

Les hyperparamètres optimaux ont été déterminés en optimisant plusieurs métriques.

Attardons-nous sur une problématique incontournable dans le machine learning dans le machine learning, à savoir l'overfitting. Une attention permanente a été apportée dans tous les modèles sur ce point, d'une part en nous assurant que l'accuracy de l'ensemble d'entraînement était proche de l'accuracy de l'ensemble de test (inférieure à 2%), mais également via une méthodologie plus complexe via la librairie HyperOpt. Nous avons utilisé la validation croisée *StratifiedKFold(n\_splits=5, random\_state=42, shuffle=True)*. Pour le modèle XGBoost en particulier, nous avons ajouté l'option *early\_stopping\_rounds: int = 50* qui est recommandée pour éviter le problème d'overfitting pour ce modèle. En vérifiant les métriques calculées sur l'ensemble d'entraînement et l'ensemble de test, nous n'avons pas identifié de problème d'overfitting.

#### 2.3.2.1 Maximisation de l'accuracy

Les résultats présentés dans le Tableau 2 montrent les performances de quatre modèles différents en termes de cinq métriques d'évaluation, avec l'optimisation basée sur l'accuracy.

	accuracy	recall	precision	f1	auc
LogisticRegression	0.8461	0.5004	0.7278	0.593	0.8694
TreeDecision	0.8389	0.4841	0.7049	0.574	0.8456
RandomForest	0.8465	0.4761	0.7477	0.5818	0.8653
<b>XGBoost</b>	<b>0.8526</b>	0.5407	0.7315	0.6218	0.8844

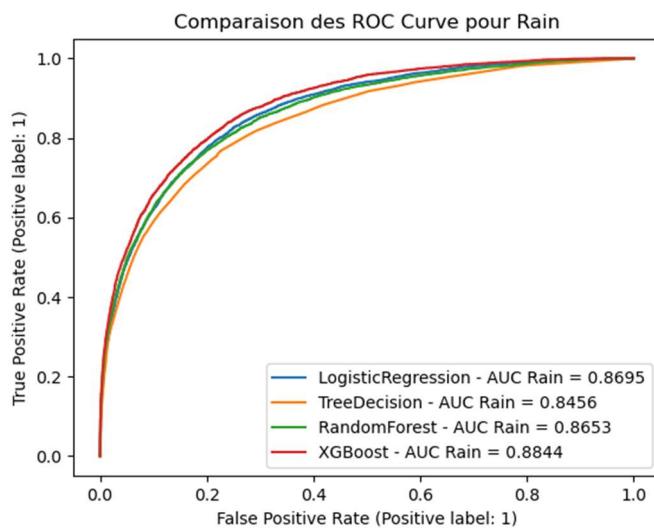
Tableau 2 : Scores des modèles ayant meilleur accuracy

- Accuracy : XGBoost a la meilleure accuracy parmi les quatre modèles, avec une valeur de 0.8526
- Recall : XGBoost a également le recall le plus élevé, ce qui suggère qu'il a une capacité supérieure à identifier les exemples positifs par rapport aux autres modèles.

- Precision : Random Forest a la precision la plus élevée, indiquant qu'il a moins de faux positifs par rapport aux autres modèles.
- F1-score : XGBoost a le F1-score le plus élevé, ce qui est une combinaison équilibrée de recall et precision.
- AUC : XGBoost a également la meilleure AUC, indiquant de bonnes performances globales pour la classification binaire.

Si l'objectif principal est d'optimiser l'accuracy, le modèle XGBoost semble être le meilleur choix

On peut observer dans la Figure 4e que la ROC courbe du modèle XGBoost est au-dessus des ROC courbes des autres modèles. Cela suggère que le modèle XGBoost a une meilleure capacité à maintenir un taux élevé de vrais positifs tout en limitant le taux de faux positifs, même à différents seuils de classification.



**Figure 4 : Les ROC courbes des 4 modèles ayant meilleur accuracy**

### 2.3.2.2 Maximisation de la précision

Les résultats présentés dans le Tableau 3 montrent les performances de quatre modèles différents en termes de cinq métriques d'évaluation, avec l'optimisation basée sur la précision.

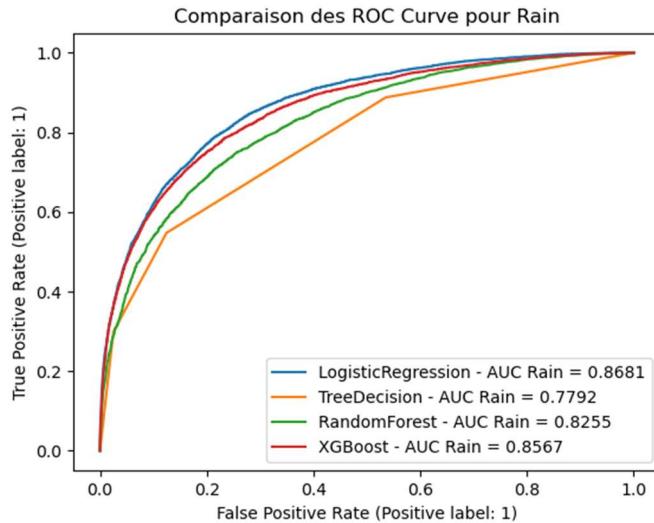
Scores des modèles ayant meilleur precision					
	accuracy	recall	precision	f1	auc
LogisticRegression	0.8455	0.4894	0.7327	0.5869	0.8681
TreeDecision	0.8233	0.2987	0.7746	0.4311	0.7792
<b>RandomForest</b>	0.783	0.0367	<b>0.8897</b>	0.0705	0.8315
XGBoost	0.841	0.4144	0.7698	0.5388	0.8567

**Tableau 3 : Scores des modèles ayant meilleure précision**

- Précision : La précision mesure la proportion d'exemples positifs parmi ceux que le modèle a identifiés comme positifs. Dans ce cas, Random Forest a la précision la plus élevée (0.8897), indiquant qu'il a une capacité à minimiser les faux positifs par rapport aux autres modèles.
- Accuracy : La précision seule ne donne pas une image complète de la performance du modèle, car elle ne prend pas en compte les faux négatifs. En termes d'accuracy, Linear Regression a la valeur la plus élevée (0.8455)

- Recall : Random Forest a le recall le plus faible (0.0367), ce qui signifie qu'il identifie très peu d'exemples positifs parmi tous les exemples réellement positifs.
- F1-score : XGBoost a un F1-score relativement équilibré, combinant recall et precision de manière équilibrée.
- AUC : Random Forest a la plus haute AUC dans ce cas (0.8315).

Si la précision est la métrique la plus importante, Random Forest pourrait être le choix préféré en se basant sur ces résultats spécifiques.



**Figure 5: Les ROC courbes des 4 modèles ayant meilleure précision**

### 2.3.2.3 Maximisation du recall

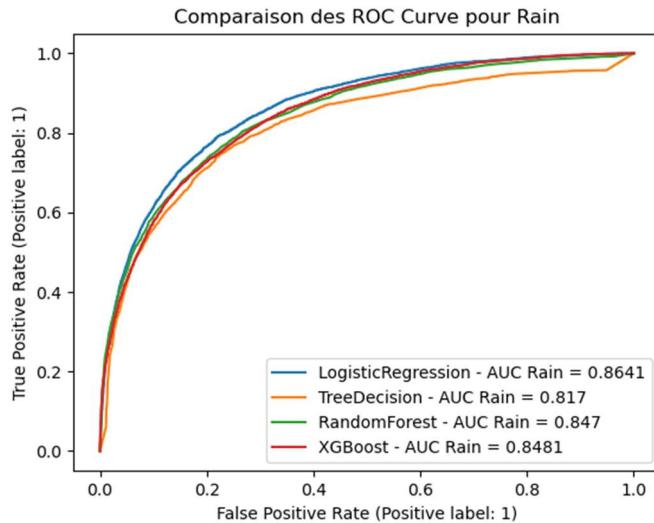
Les résultats présentés dans le Tableau 4 montrent les performances de quatre modèles différents en termes de cinq métriques d'évaluation, avec l'optimisation basée sur le recall.

Scores des modèles ayant meilleur recall					
	accuracy	recall	precision	f1	auc
<b>LogisticRegression</b>	0.7366	<b>0.8464</b>	0.4531	0.5902	0.864
TreeDecision	0.8313	0.4944	0.6669	0.5679	0.818
RandomForest	0.8397	0.5062	0.6956	0.586	0.8469
XGBoost	0.8292	0.5547	0.6365	0.5928	0.8481

**Tableau 4: Scores des modèles ayant meilleur recall**

- Recall : Recall mesure la proportion d'exemples positifs réellement identifiés par le modèle. Linear Regression a le recall le plus élevé (0.8464), indiquant qu'il a la meilleure capacité parmi les modèles à capturer la majorité des exemples positifs. Cependant, il est important de noter que cette performance peut être associée à une baisse de précision.
- Precision : Linear Regression a la precision la plus faible (0.4531), indiquant qu'il a tendance à identifier un grand nombre de faux positifs.
- Accuracy : Random Forest a la valeur la plus élevée en termes d'accuracy (0.8397), mais cela peut être dû à une balance entre les performances en termes de faux positifs et de faux négatifs.
- F1-score : Le F1-score de Linear Regression est relativement équilibré, étant donné son recall élevé et sa precision basse.

- AUC : AUC mesure la capacité du modèle à discriminer entre les classes positives et négatives. XGBoost a la plus haute AUC dans ce cas (0.8481).



**Figure 6: : Les ROC courbes des 4 modèles ayant meilleur recall**

#### 2.3.2.4 Maximisation du F1-score

Les résultats présentés dans le Tableau 5 montrent les performances de quatre modèles différents en termes de cinq métriques d'évaluation, avec l'optimisation basée sur le F1-score

Scores des modèles ayant meilleur f1					
	accuracy	recall	precision	f1	auc
LogisticRegression	0.846	0.5002	0.7277	0.5929	0.8695
TreeDecision	0.8409	0.4883	0.7114	0.5791	0.8498
RandomForest	0.8494	0.5049	0.7409	0.6006	0.8699
<b>XGBoost</b>	<b>0.8554</b>	<b>0.5475</b>	<b>0.7397</b>	<b>0.6292</b>	<b>0.8852</b>

**Tableau 5: Scores des modèles ayant meilleur F1-score**

- F1-score : Le modèle XGBoost affiche le F1-score le plus élevé (0.6292) parmi tous les modèles. Le F1-score est une métrique qui prend en compte à la fois la precision et le recall. Cela suggère que XGBoost atteint un équilibre optimal entre l'identification des vrais positifs (recall) et la minimisation des faux positifs (precision).
- Accuracy : XGBoost a également une accuracy élevée (0.8554), indiquant une classification correcte d'une grande proportion des exemples.
- Recall et Precision : XGBoost a des valeurs de recall et de precision compétitives par rapport aux autres modèles, ce qui renforce l'idée d'un équilibre entre la sensibilité aux vrais positifs et la limitation des faux positifs.
- AUC : XGBoost présente également la plus haute AUC (0.8852), ce qui confirme sa capacité à bien discriminer entre les classes positives et négatives.

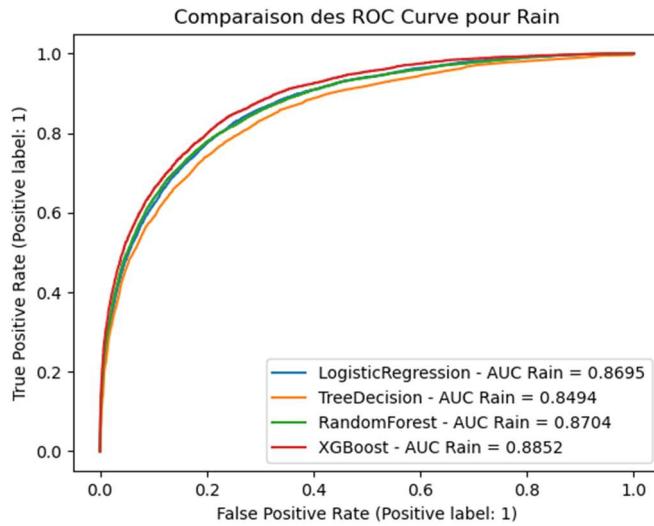


Figure 7: Les ROC courbes des 4 modèles ayant meilleur F1-score

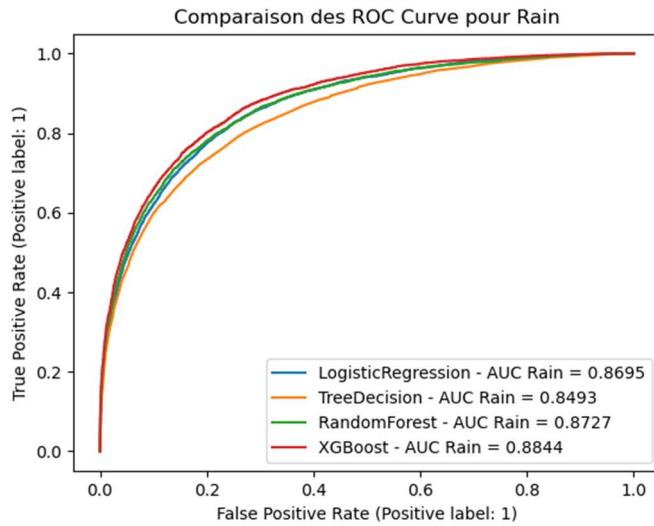
#### 2.3.2.5 Maximisation de l'AUC

Les résultats présentés dans le Tableau 6 montrent les performances de quatre modèles différents en termes de cinq métriques d'évaluation, avec l'optimisation basée sur l'AUC.

Scores des modèles ayant meilleur auc					
	accuracy	recall	precision	f1	auc
LogisticRegression	0.846	0.5004	0.7273	0.5929	0.8695
TreeDecision	0.8399	0.4478	0.7341	0.5563	0.8493
RandomForest	0.8497	0.498	0.7474	0.5978	0.8716
<b>XGBoost</b>	<b>0.8538</b>	<b>0.5358</b>	<b>0.74</b>	<b>0.6216</b>	<b>0.8844</b>

Tableau 6: Scores des modèles ayant meilleur AUC

- AUC : L'AUC (Area Under the Curve) est souvent utilisée comme métrique pour évaluer les modèles de classification binaire. Dans ce cas, XGBoost a la valeur d'AUC la plus élevée (0.8844), ce qui indique une performance globale solide pour la classification binaire. Cela suggère que le modèle XGBoost a une bonne capacité à discriminer entre les classes positives et négatives.
- Accuracy : XGBoost a également la meilleure accuracy (0.8538), ce qui signifie qu'il a correctement classé une grande proportion des exemples
- Recall : XGBoost a le recall le plus élevé, indiquant qu'il a une capacité supérieure à identifier les exemples positifs par rapport aux autres modèles.
- Precision : Random Forest a la precision la plus élevée, ce qui suggère qu'il a moins de faux positifs par rapport aux autres modèles.
- F1-score : XGBoost a le F1-score le plus élevé, ce qui est une combinaison équilibrée de recall et precision



**Figure 8: Les ROC courbes des 4 modèles ayant meilleur AUC**

#### 2.3.2.6 Comparaison avec des modélisations par lieu et par zone climatique

Les résultats dans les Sections 7-11 suggèrent que XGBoost est le modèle qui offre la meilleure performance globale, en particulier en termes d'AUC, accuracy, recall, precision et F1-score. Ces modélisations ont été réalisées avec l'ensemble des données du dataset. Comparons maintenant les performances d'un XGBoost entraîné en optimisant l'AUC-ROC sur l'ensemble du jeu de données avec des modélisations ciblant chaque station météo d'une part (filtrage par la variable *Location*), et chaque zone climatique d'autre part (filtrage via la variable *Climat* issue de la clusterisation). Le Tableau 7 indiquera donc les performances moyennes de chaque approche ainsi que le modèle offrant les scores les plus intéressants pour une *Location* donnée et pour une zone climatique donnée.

Comparaison des XGBoost selon le périmètre de modélisation					
	accuracy	recall	precision	f1	auc
Macro	0.8642	0.5558	0.7642	0.6436	0.9003
Micro (moyenne)	0,8560	0,4563	0,7757	0,5690	0,8716
Micro (meilleur : PearceRAAF)	0,9167	0,5647	0,8727	0,6857	0,9584
Climat (moyenne)	0,8646	0,5592	0,7667	0,6442	0,8993
Climat (meilleur : Centre)	0,9378	0,4743	0,7742	0,5882	0,9379

**Tableau 7 : Comparaison des XGBoost selon le périmètre de modélisation**

A titre indicatif, les hyperparamètres du modèle global sont un learning rate de 0,23, une profondeur max de 6 et 50 estimateurs. Pour les modèles locaux le learning rate est 0,2, la profondeur max 3 avec 4 estimateurs. Enfin, les modèles climatiques ont également un learning rate de 0,2, mais une profondeur max de 4 et comportent 30 estimateurs.

A première vue, les performances moyennes des modèles par climat semblent très proches du modèle global. Les performances moyennes des modèles locaux ont l'air d'être légèrement inférieures. Toutefois, la réalité est un peu plus complexe. On voit notamment que le modèle de climat le plus performant (recouvrant le centre de l'Australie) performe nettement mieux que le modèle global. De même, le meilleur modèle local, PearceRAAF, est celui qui présente les meilleures performances de tous les modèles confondus au regard de l'AUC.

Afin que les données soient comparables, regardons les performances des trois niveaux de modélisation appliqués sur trois *Location*, représentées dans le Tableau 8.

Comparaison des modèles pour PearceRAAF					
	accuracy	recall	precision	f1	auc
Micro	0,9167	0,5647	0,8727	0,6857	<b>0,9584</b>
Global	0,9053	0,6526	0,7470	0,6966	0,9486
Climat – 4, Intermédiaire	0,9170	0,6818	0,8108	0,7407	0,9463

Comparaison des modèles pour NorfolkIsland					
	accuracy	recall	precision	f1	auc
Micro	0,7555	0,3352	0,7262	0,4586	0,7655
Global	0,7883	0,4837	0,7355	0,5836	<b>0,8094</b>
Climat – 0, Côte Est	0,7783	0,4776	0,7619	0,5872	0,8037

Comparaison des modèles pour Penrith					
	accuracy	recall	precision	f1	auc
Micro	0,8593	0,3879	0,8036	0,5233	0,8284
Global	0,8761	0,5517	0,7901	0,6497	0,8677
Climat – 4, Intermédiaire	0,8693	0,5468	0,8352	0,6609	<b>0,8922</b>

**Tableau 8 : Les performances des trois niveaux de modélisation pour PearceRAAF, NorfolkIsland et Penrith**

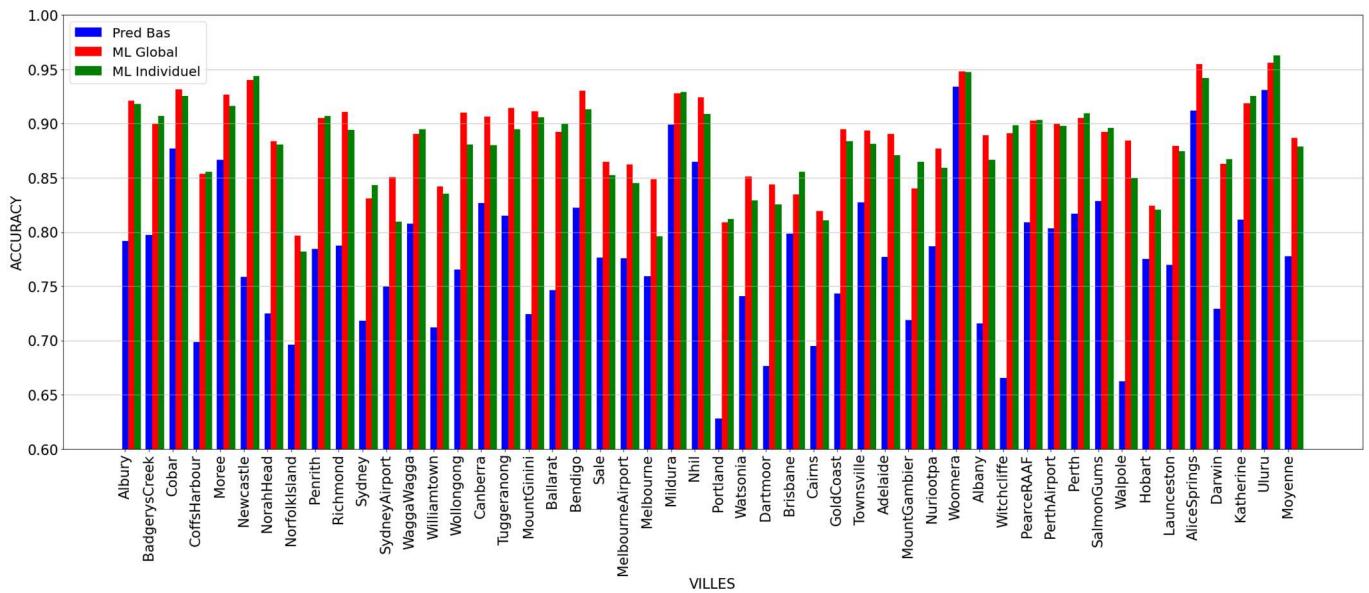
Précisons en premier lieu que le taux de journée pluvieuses est très différent pour ces trois lieux et est à rapprocher directement de la valeur de l'accuracy : 18% pour PearceRAAF, 31% pour NorfolkIsland et 20% pour Penrith, soit, avec un modèle prédisant systématiquement qu'il ne pleuvra pas, des accuracy « naïves » respectivement de 0,82, 0,69 et 0,80.

Les résultats dans Tableau 8 illustrent qu'aucune des trois approches, globale, locale ou climatique, n'est meilleure de façon systématique, quelle que soit la métrique. Si nous nous basons sur l'AUC, c'est pour PearceRAAF un modèle entraîné spécifiquement sur les données de la station qui offre les meilleures performances. Pour NorfolkIsland, le modèle global performe mieux que les deux autres. Enfin, on préférera une modélisation par climat pour Penrith.

Ces trois niveaux de granularité semblent donc complémentaires.

De façon plus détaillée, observons l'accuracy pour chacun des 49 lieux en comparant :

- Un modèle global, entraîné sur l'ensemble du dataset (« ML Global »)
- Un modèle local, entraîné spécifiquement sur les données du lieu concerné (« ML Individuel »)
- Un modèle naïf se contentant de prédire qu'il ne pleuvra jamais, qui nous permettra de relativiser les scores obtenus par les deux premiers modèles (« Pred Bas »)



La moyenne de l'ensemble des *Location* donne un score de 0,879 pour les ML Individuels, 0,886 pour les ML Global et 0,777 pour le modèle « Pred Bas ». L'approche macro fournit donc un score légèrement meilleur en moyenne. La différence est cependant faible, et nous pouvons voir sur le graphique ci-dessus que même en observant chaque *Location*, il n'y a que peu de différences de performances.

### 2.3.3 Impact du feature enginerring

Maintenant que nous avons pu déterminer que le XGBoost performe mieux que les autres modèles de machine learning et que nous avons identifié des hyperparamètres adaptés, regardons quel impact ont eu les travaux de feature engineering.

Le modèle dit « avec features d'origine » effectue des transformations élémentaires sur le dataset *WeatherAUS.csv*, tel le remplacement des ‘Yes’ par des True pour les variables binaires ainsi qu'un encodage OneHot des variables catégorielles. Les valeurs manquantes (NA) sont traitées par suppression pure et simple, sans substitution. Les quatre variables ayant un taux très important de NA (*Sunshine* : 48%, *Evaporation* : 43%, *Cloud3pm* : 41%, *Cloud9am* : 38%) sont supprimées, faute de quoi un *dropna* global entraîne la suppression de plus de la moitié du dataset.

Le modèle dit « avec nouvelles features » est issu des travaux menés dans la première partie du projet. Les valeurs nulles sont gérées par KNN Imputation, les variables catégorielles de direction du vent sont remplacées par des variables quantitatives trigonométriques, les *Location* sont remplacées par la latitude et la longitude, une variable Climat indique le résultat de la clusterisation par zone climatique, l'amplitude thermique est ajoutée (=TempMax-TempMin), et afin deux variables temporelles respectivement égales au cosinus de  $2\pi \times (\text{numéro du jour dans l'année}/365)$  et cosinus  $4\pi \times (\text{numéro du jour dans l'année}/365)$ .

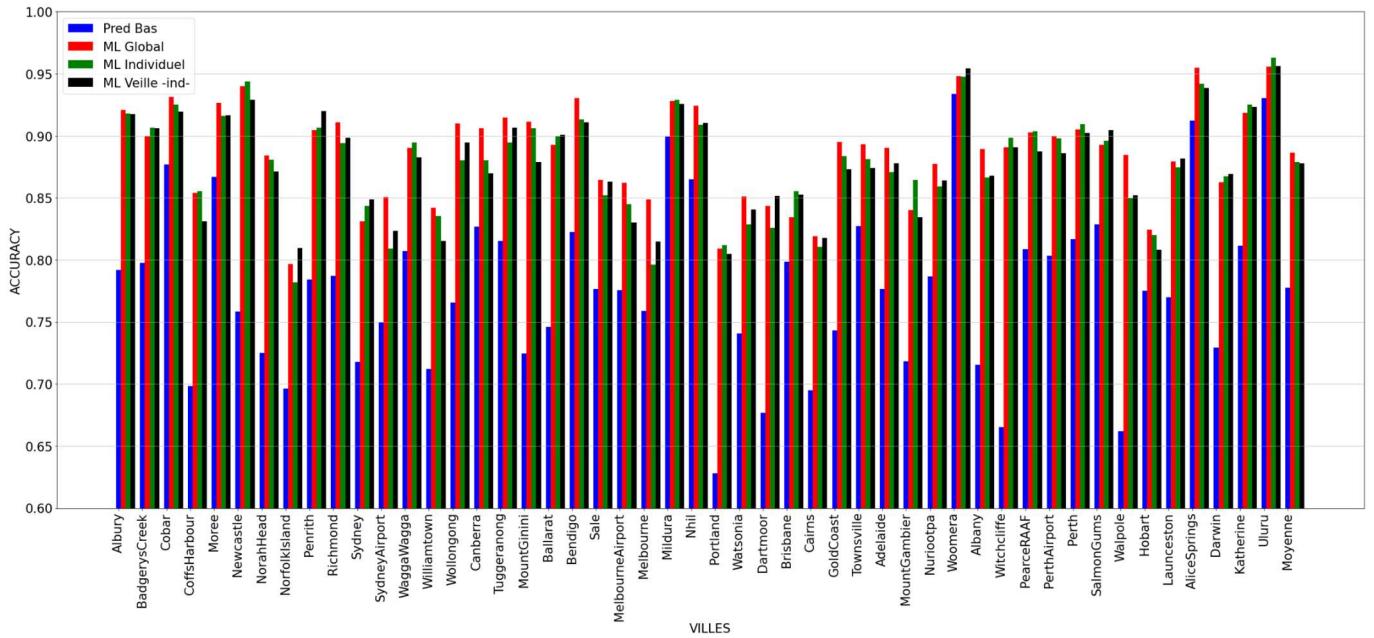
Nous comparons dans le Tableau 9 les performances d'un XGBoost avec les mêmes hyperparamètres (learning rate 0,23, max\_depth 6, n\_estimators 50, obtenu par optimisation de l'AUC-ROC) sur ces deux dataset portant sur toute l'Australie dans les deux cas :

Performances d'un XGBoost en fonction du feature engineering					
	accuracy	recall	precision	f1	auc
Avec nouvelles features	0.8642	0.5558	0.7642	0.6436	0.9003
Avec features d'origine	0.8566	0.5299	0.7414	0.618	0.8847

Tableau 9 : Performances d'un XGBoost en fonction du feature engineering

Certes, le modèle bénéficiant des nouvelles features propose de meilleures performances quelle que soit la métrique observée, mais le gain est très faible, tout particulièrement au regard du temps passé pour effectuer le feature engineering.

Nous avons également voulu enrichir les variables pour chaque journée en reprenant également les valeurs de la veille. Ainsi, pour chaque observation, pour prédire *RainTomorrow*, nous disposons des relevés pour le jour J mais également J-1. En plus des trois modélisations décrites plus haut, nous avons donc ici 49 modèles « ML Veille-ind » entraînés avec ces nouvelles features.



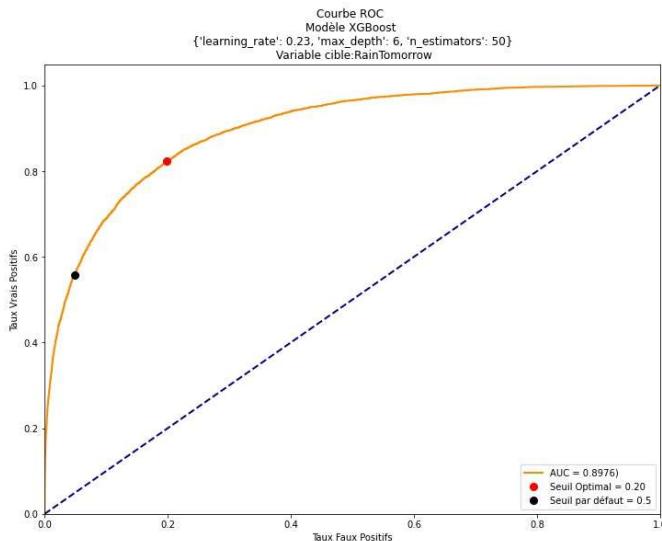
On constate qu'avec les données de la veille, le score du modèle ne s'améliore pas par rapport à un modèle individuel (0.886 contre 0.877). Là aussi, nous obtenons que les différences soient faibles.

La phase de feature engineering, bien que particulièrement chronophage, ne semble donc pas être déterminante dans l'échelle de qualité fournie par les modèles.

### 2.3.4 Seuil de probabilité

Importance des features sur un modèle XGBoost optimisant l'AUC-ROC.

Jusqu'ici, nous avons exploité directement les prédictions de nos différents modèles XGBoost. Nous allons maintenant profiter du fait que ce modèle nous permet de connaître la probabilité de chaque prédition pour l'affiner.



**Figure 9 : Courbe ROC issue d'un XGBoost entraîné sur toute l'Australie**

Nous l'avons vu dans les tableaux de métriques précédent : un point faible est les performances sur le recall, qui indique globalement que, lorsqu'il pleut, nous ne sommes capables environ qu'une fois sur deux de prévoir la précipitation. Rappelons quand dans le un modèle probabiliste de classification, le seuil est par défaut de 0,5. Ce seuil est représenté par le point noir dans la Figure 9. Quelques remarques :

- Nous confirmons immédiatement visuellement que le taux de vrai positif n'est que d'environ la moitié
- Nous voyons que le taux de faux positif est en revanche particulièrement faible
- Nous disposons donc de modèles ayant une spécificité plutôt bonne, mais une sensibilité assez mauvaise

En d'autres termes, nous sommes assez bons pour garantir qu'il ne pleuvra pas mais assez mauvais pour prédire de façon assez fiable qu'il pleuvra.

Ce constat ne permet pas de juger de la qualité du modèle : pour cela, il faudrait connaître l'objectif du client sur ce modèle. A-t-il besoin d'une garantie absolue d'absence de pluie ? Préfèrerait-il savoir avec certitude lorsqu'il pleuvra ? Souhaite-t-il un compris entre les deux approches ? Encore plus pragmatiquement, quel serait le coût d'une erreur, et donc quel serait le taux de faux positifs maximal ou de vrais positifs minimal pour que le modèle soit économiquement viable ?

La problématique posée ne nous permet malheureusement pas de répondre à ces questions, et donc il ne nous est pas possible de trancher pour savoir s'il fait faire varier le seuil de probabilité.

Nous allons cependant proposer une approche « de compromis », que nous avons nommé pompeusement « Seuil Optimal » qui est le seuil permettant simultanément de maximiser le taux de vrais positifs et de minimiser celui de faux positifs. Dans nos différentes courbes ROC ce point sera représenté par le point rouge et sera généralement sur le coude de la courbe ROC.

Ce changement réduit logiquement légèrement l'accuracy, mais nous offre un gain souvent significatif sur le recall. Ici, nous voyons que les jours de pluie sont d'environ 0,8 et les faux positifs de 0,2, c'est-à-dire que lorsqu'il pleut, nous pouvons le prévoir environ 4 fois sur 5, de même que lorsqu'il ne pleut pas. Dans le cadre d'un bulletin météo dans la presse, par exemple, nous pouvons imaginer qu'il s'agit là de performances plus acceptables pour le grand public que la situation précédente dans laquelle nous ne pouvions prédire que la moitié des jours de pluie, quand bien même l'accuracy était plus élevée de 6%.

Observons plus en détail les performances sur les différentes métriques, présentées dans le Tableau 10, et les matrices de confusions présentées dans le Tableau 11. Notons d'ailleurs qu'au-delà de la légère baisse

d'accuracy, nous avons une baisse importante de la précision, c'est-à-dire que, désormais, nous nous trompons presque une fois sur deux lorsque nous prédisons qu'il pleuvra.

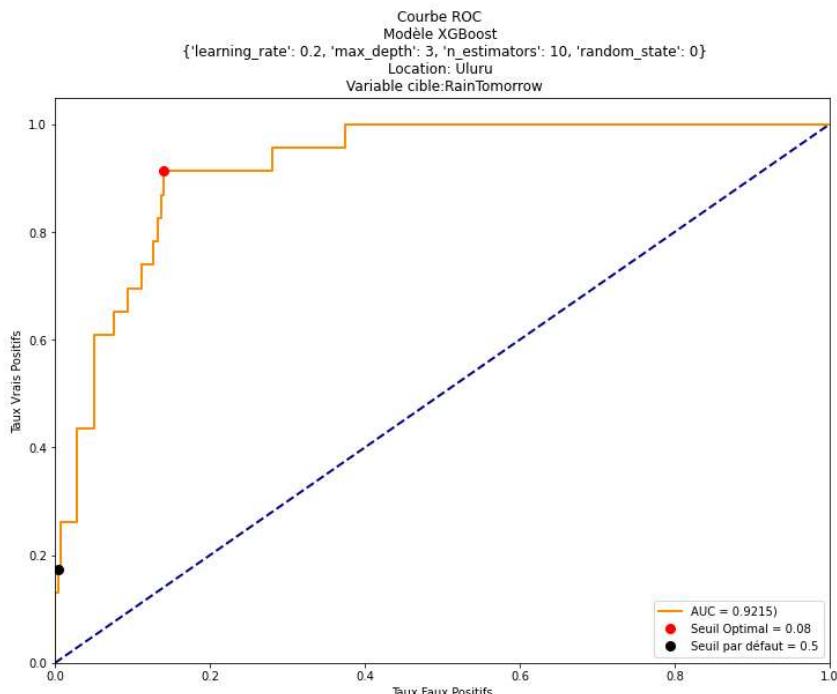
Impact du seuil de probabilité					
	accuracy	recall	precision	f1	auc
Seuil par défaut (0,50)	0.8642	0.5558	0.7642	0.6436	0.9003
Seuil Optimal (0,22)	0.8196	0.814	0.563	0.6656	0.9003

**Tableau 10 : Impact du seuil de probabilité**

		Classe prédictive		Classe prédictive			
		False	True	False	True		
Classe réelle	False	74,2%	3,8%	Classe réelle	False	64,0%	13,9%
	True	9,8%	12,3%		True	4,1%	18,0%
Seuil par défaut (0,5)				Seuil Optimal (0,2)			

### Tableau 11 : Matrices de confusion

Regardons le cas tout à fait extrême de la station d'Uluru, située en plein désert (zone Centre) :



**Figure 10 : Courbe ROC des prédictions d'Uluru**

		Classe prédictive		Classe prédictive	
		False	True	False	True
Classe réelle	False	92,0%	0,3%	False	79,4%
	True	6,3%	1,3%	True	13,0%
Seuil par défaut (0,5)			Seuil optimal (0,08)		

En passant du seuil par défaut au seuil optimal (0,08 seulement !), nous passons de prédictions qui nous permettaient de prévoir avec une grande fiabilité lorsqu'il ne pleuvra pas à un modèle dans lequel nous arrivons à capter quasiment l'intégralité des jours où il pleuvra ! Dans une zone aussi aride, nous pouvons imaginer qu'il est beaucoup plus intéressant de pouvoir anticiper les jours où il pleuvra potentiellement plutôt que d'affirmer avec certitude lorsqu'il ne pleuvra pas, mais, là encore, sans contexte économique sur les objectifs à atteindre, il est impossible de trancher sur l'approche la plus pertinente. Il reste très intéressant de voir que les courbes ROC nous permettent de visualiser l'impact de la modification du seuil.

La Figure 11 les 6 courbes ROC des zones climatiques avec les points de seuil par défaut et de seuil optimal : nous voyons que l'impact varie considérablement selon les zones climatiques :

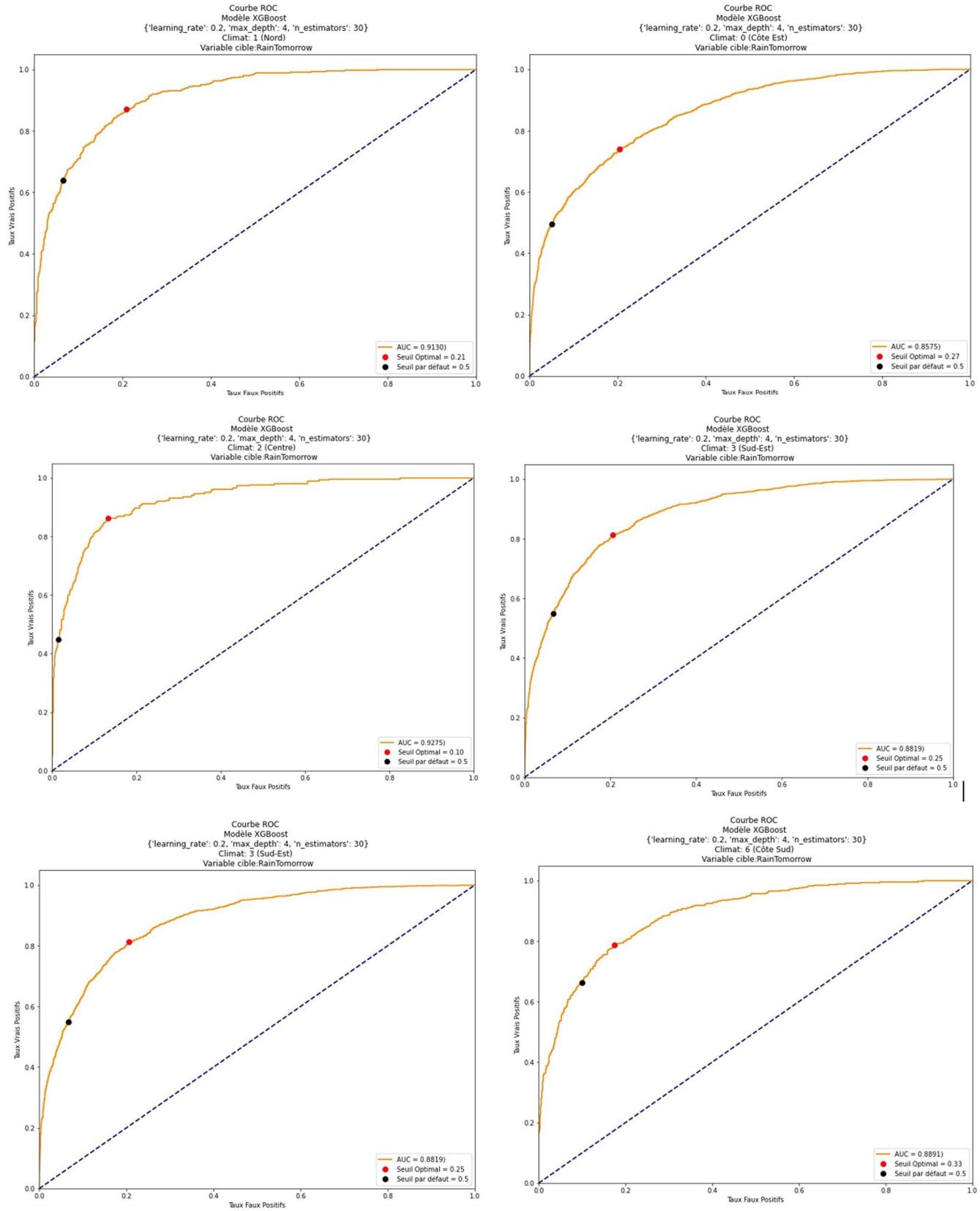


Figure 11: Les courbes ROC des 6 zones climatiques

## 2.3.5 Interprétabilité des modèles

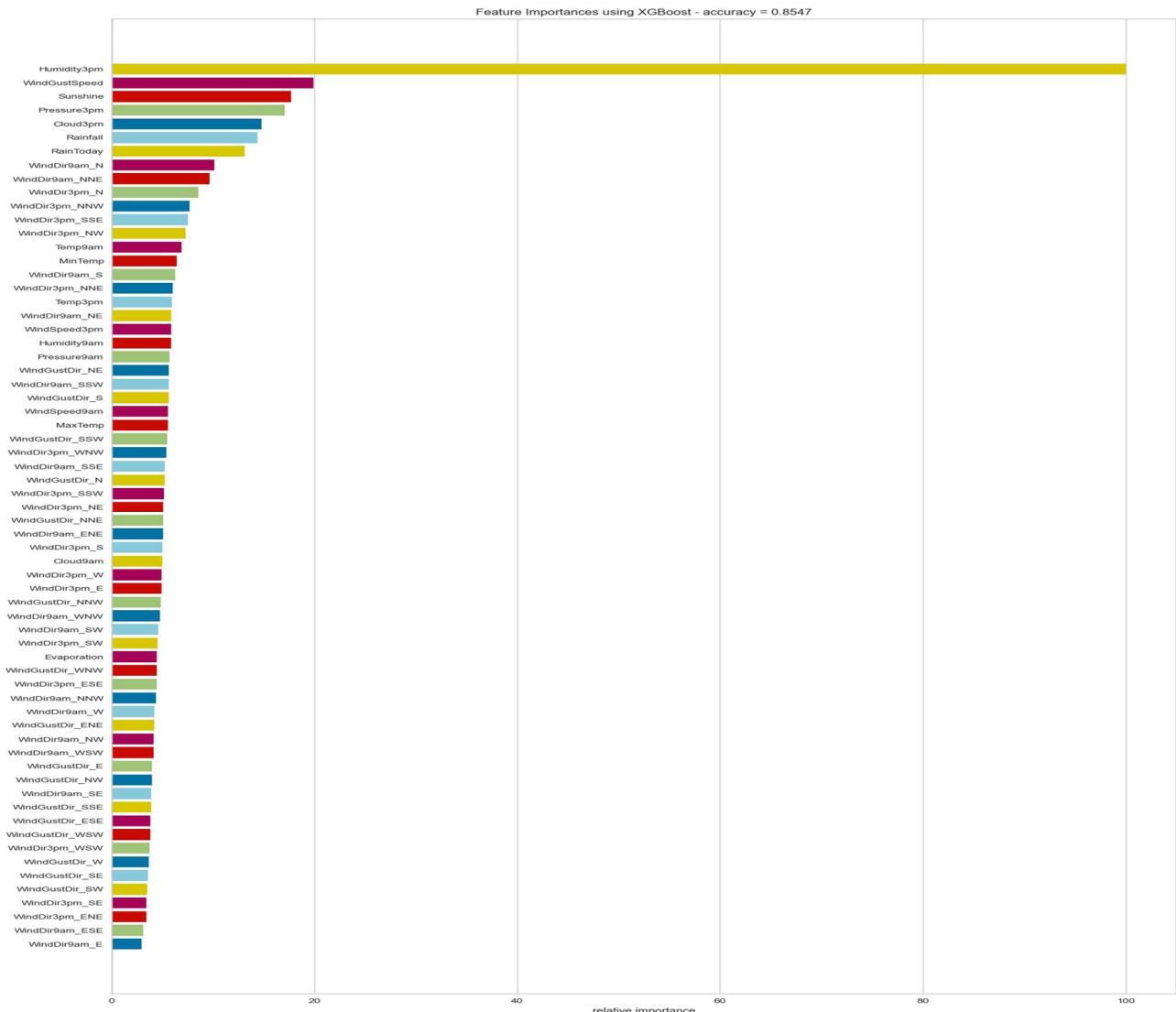
### 2.3.5.1 Importance des features

Sur la quasi-totalité des modèles entraînés, la variable la plus importante semble être de loin *Humidity3pm*, indiquant le taux d'humidité à 15h00. Contre toute attente, *RainToday* semble finalement peu importante dans la prédiction de chute de pluie le lendemain.

Nous avons toutefois pu constater des divergences entre les modèles sur l'importance de chaque variable. Ainsi, si *XGBoost* conserve une importance pour l'ensemble des features, c'est loin d'être le cas des autres modèles. Les *RandomForest* et *TreeClassifier* écartent en particulier la plupart des features pour se concentrer sur quelques-unes. Il est également intéressant d'observer que le poids accordé à chaque variable dépend de la métrique à optimiser. Le cas le plus frappant concerne les *RandomForest* entraînés pour optimiser la précision : dans un contexte de modélisation globale, il s'agit du seul cas où la variable la plus importante devient alors *Cloud3pm*.

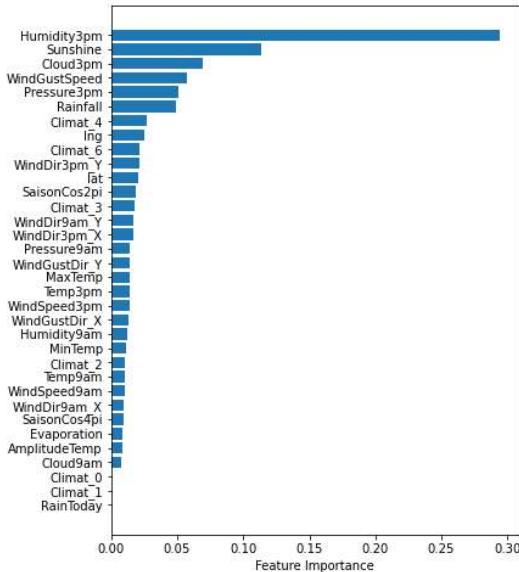
Il est intéressant de relever que *Humidity3pm* était effectivement la variable la plus corrélée avec *RainTomorrow* 0,45, d'une façon symétrique avec *Sunshine* (-0,45). Cette dernière a pourtant un poids nettement inférieur dans les features importances.

Comme nous l'avons indiqué plus haut, nous avons testé des modélisations selon des feature engineering différents, afin de tester plusieurs approches. Les Figure 12 et Figure 13 représentent les features importances sur un modèle entraîné sur un dataset au sein duquel en particulier les variables de vent ont été encodées en OneHot et les features importances d'un second modèle entraîné cette fois avec des variables venteuses trigonométriques et l'ajout de différentes features (zone climatique, coordonnées, variables temporelles).



**Figure 12 : Features importances du modèle XGBoost au niveau macro**

(dataset avec encodage OneHot)



**Figure 13 : Features importances du modèle XGBoost au niveau macro**

(dataset avec features de zones climatiques et variables venteuses converties via sin/cos)

Sur ce second modèle, entraîné sur toute l’Australie, les variables ajoutées de la longitude, la latitude et l’appartenance aux zones climatiques Intermédiaire et Sud-Est sont identifiées parmi le quart des features les plus significatives.

Observons, dans la Figure 14, maintenant les divergences sur des modèles entraînés sur chacune des zones climatiques. Pour toutes les zones climatiques, les variables *Humidity3pm* et *Sunshine* sont très importantes. Il y a par contre des divergences importantes sur le poids de certaines autres variables. Par exemple, si *Cloud3pm* est la feature la plus importante pour la zone Centre, elle se retrouve dans le dernier tiers des variables pour la zone Sud. Nous pouvons voir que la latitude et la longitude se retrouvent généralement dans la moitié des variables les plus importances, avec toutefois une distinction notable pour la latitude dans la zone Nord qui est carrément mise de côté par le modèle. L’amplitude thermique est assez peu exploitée. SaisonCos2pi est se retrouve dans le premier quart des variables pour la moitié des zones climatiques (Nord, Sud-Est, Sud).

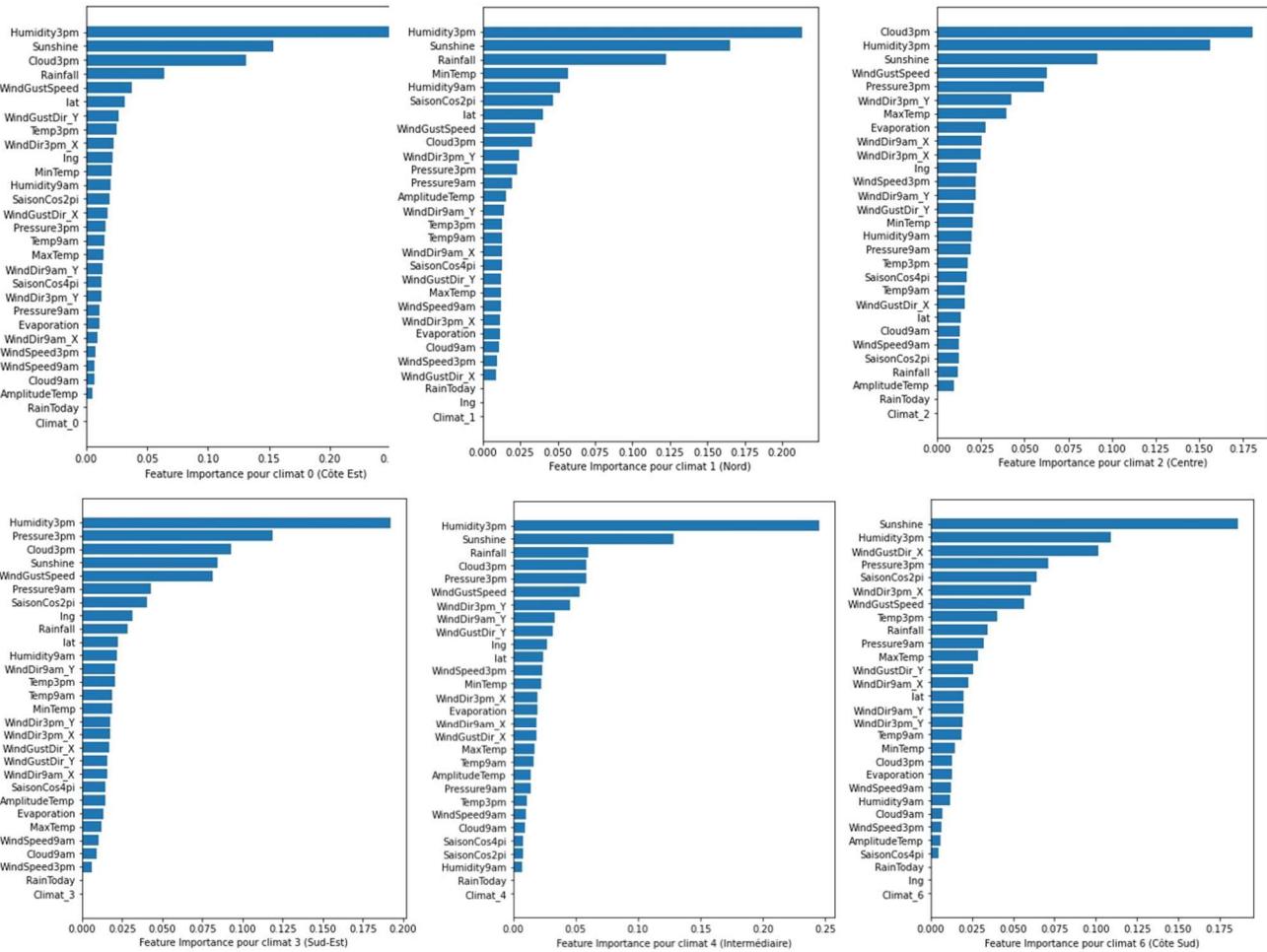


Figure 14: Features importances des 6 zones climatiques

### 2.3.5.2 Valeurs de Shapley

Pour déterminer quelles caractéristiques sont généralement les plus importantes pour les prédictions de notre modèle, nous pouvons utiliser un diagramme à barres des valeurs moyennes de SHAP pour toutes les observations. Prendre la moyenne des valeurs absolues garantit que les valeurs positives et négatives ne s’annulent pas.

On observe dans la Figure 15 que la variable avec la valeur SHAP moyenne la plus élevée est *Humidity3pm*, ce qui indique qu’elle a l’impact le plus important sur les prédictions de notre modèle. Ces informations peuvent nous aider à comprendre quelles variables sont essentielles au processus décisionnel du modèle.

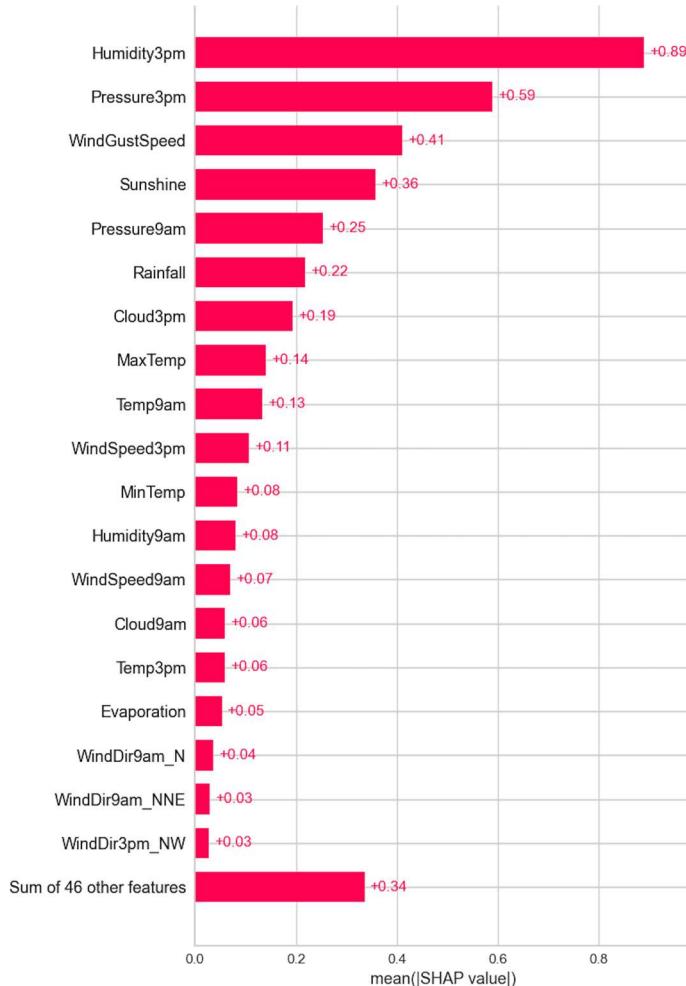


Figure 15: Valeurs de Shapley

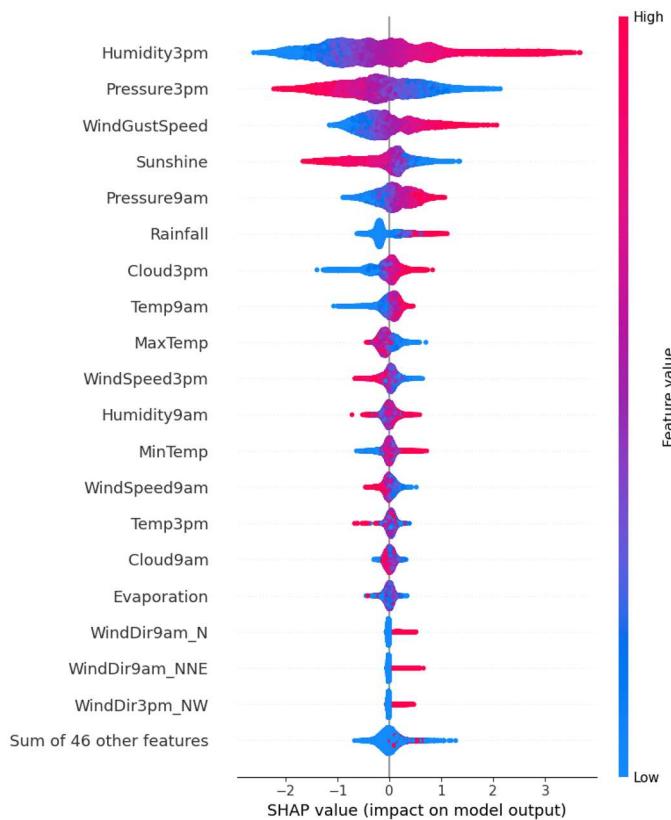
### 2.3.5.3 Beeswarm Plot

Le graphique Beeswarm est une visualisation utile pour examiner toutes les valeurs SHAP pour chaque entité. L'axe Y regroupe les valeurs SHAP par caractéristique, la couleur des points indiquant la valeur de la caractéristique correspondante. En règle générale, les points les plus rouges représentent des valeurs de caractéristiques plus élevées.

Le beeswarm plot peut aider à identifier les relations importantes entre les caractéristiques et les prédictions du modèle. Dans ce graphique, les caractéristiques sont classées selon leurs valeurs SHAP moyennes.

En examinant les valeurs SHAP dans la Figure 16, nous pouvons commencer à comprendre la nature des relations entre les variables et la pluie du lendemain. Par exemple, pour *Humidity3pm* et *WindGustSpeed*, nous observons que les valeurs SHAP augmentent à mesure que la valeur de la fonctionnalité augmente. Cela suggère que des valeurs plus élevées de *Humidity3pm* et *WindGustSpeed* contribuent à la probabilité qu'il va pleuvoir demain plus élevée.

En revanche, pour la *Pressure3pm* et la *Sunshine*, nous remarquons la tendance inverse, où des valeurs de caractéristiques plus élevées conduisent à des valeurs SHAP plus faibles. Cette observation implique que des valeurs de *Pressure3pm* et de *Sunshine* plus élevées sont associées à la probabilité qu'il va pleuvoir demain plus faible.



**Figure 16: Beeswarm Plot**

#### 2.3.5.4 Dependence Plots

Pour mieux comprendre les relations entre les caractéristiques individuelles et leurs valeurs SHAP correspondantes, nous pouvons créer des tracés de dépendance. Un diagramme de dépendance est un nuage de points qui montre la relation entre la valeur SHAP et la valeur de caractéristique pour une seule caractéristique.

En analysant les diagrammes de dépendance, la Figure 17, nous pouvons confirmer les observations faites dans le beeswarm plot. Par exemple, lorsque nous créons un diagramme de dépendance pour *Humidity3pm* et *WindGustSpeed*, nous observons une relation positive entre les valeurs de ces variables et les valeurs SHAP. En d'autres termes, des valeurs de ces deux variables plus élevées entraînent des prévisions de la pluie demain plus élevées.

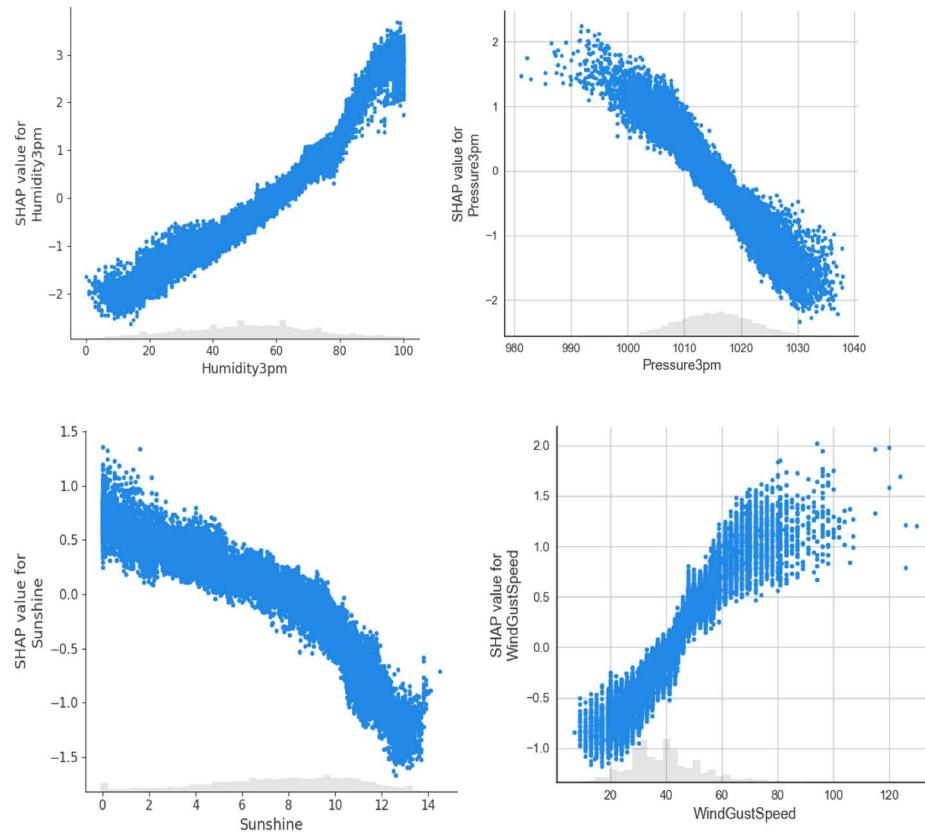
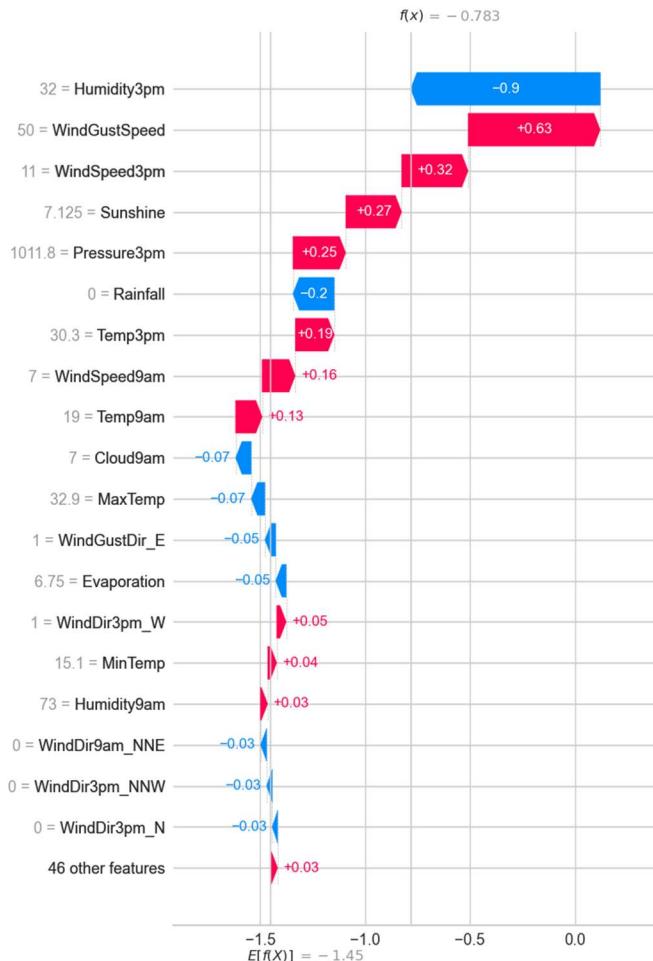


Figure 17: Dépendences plots

#### 2.3.5.5 Waterfall Plot

Ce graphique nous aide à visualiser les valeurs SHAP de chaque échantillon dans nos données individuellement. La Figure 18 visualise les valeurs SHAP du premier échantillon de test.

En ignorant les signes, l'ampleur de la valeur SHAP pour la *Humidity3pm*, 0.9, est supérieure à celle des autres variables. Cela impliquait que *Humidity3pm* avait l'impact le plus significatif sur cette prédiction particulière.



**Figure 18: Waterfall plot, exemple 1**

Tout comme nous avons visualisé les valeurs SHAP du premier échantillon, nous pouvons également visualiser les valeurs SHAP du deuxième échantillon de test, comme dans la Figure 19.

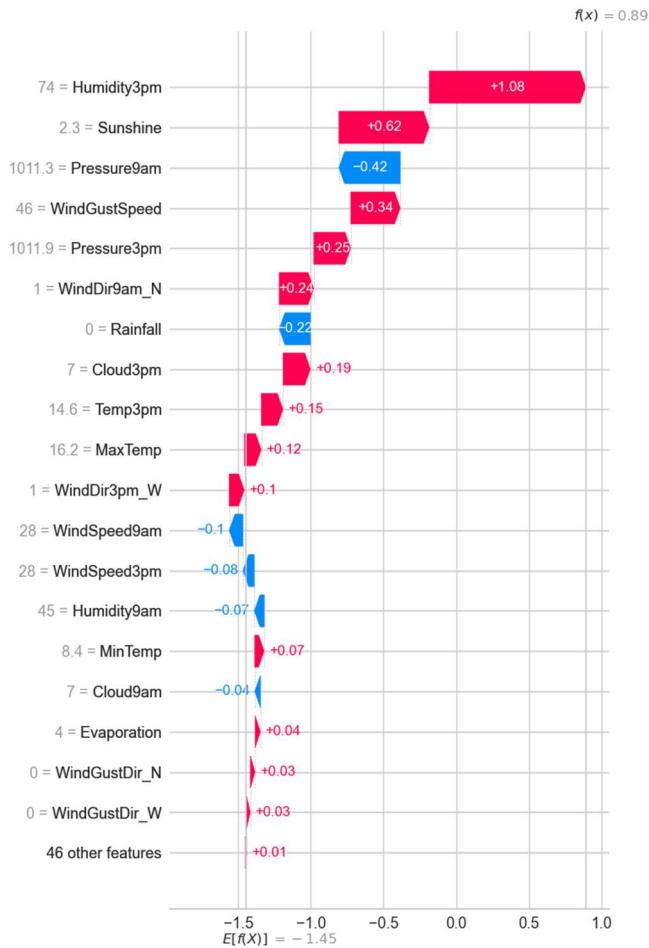


Figure 19: Waterfall plot, exemple 2

## 2.4 Deep Learning avec Keras et TensorFlow

### 2.4.1 DNN

#### 2.4.1.1 Nombre de couches et nombre de neurones

Notre première problématique va consister à dimensionner correctement notre réseau de neurones dense, tant en nombre de couches cachées qu'en nombre de neurones par couches.

Pour cela, nous allons procéder itérativement par couche. Nous allons débuter par un « réseau » sans couche cachée, constitué du seul neurone de la couche de sortie.

Nous allons ensuite ajouter une première couche cachée, avec un faible nombre de neurones (5), un grand (200) et un nombre intermédiaire (50). Nous avons en réalité tester beaucoup d'autres combinaisons : nous allons retenir ici les cas extrêmes pour synthétiser les résultats obtenus.

Nous allons ensuite ajouter une deuxième couche en testant plusieurs combinaisons, puis nous regarderons l'apport d'une troisième couche.

Les modèles ont tous été entraînés avec un *learning rate* de 0.001, un *batch\_size* de 512, des fonctions d'activation *tanh* et sur 300 époques. Ces paramètres seront affinés par la suite.

Notre fonction loss est une *binary\_crossentropy*, et notre métrique une *binary\_accuracy*. Enfin, le neurone de la couche de sortie sera activé par une fonction *sigmoïde*.

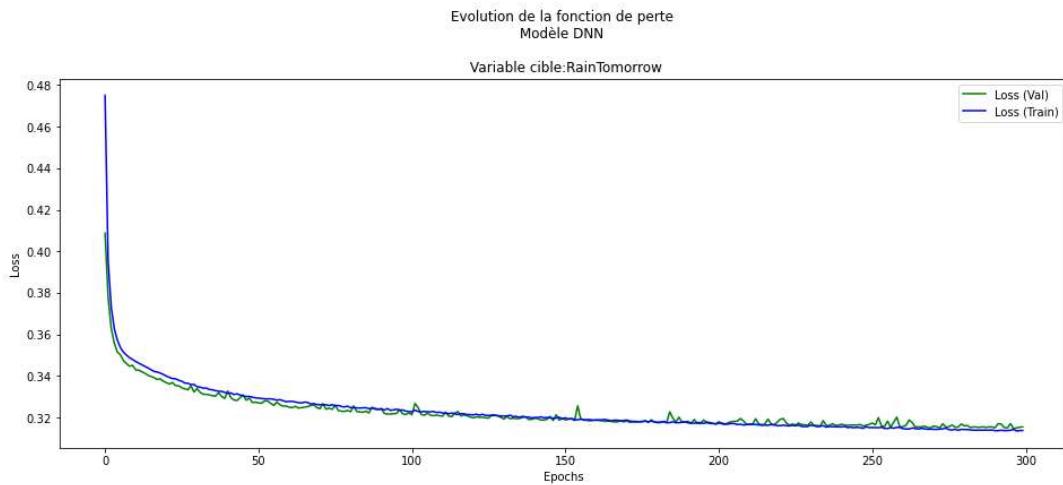
Comparaison des réseaux de neurones					
	accuracy	recall	precision	f1	auc
Aucune couche cachée	0,8533	0,5198	0,7375	0,6098	0,8773
Couche cachée (CC) de 5 Neurones (n)	0,8612	0,5484	0,7554	0,6355	0,8925
CC 200 neurones	0,8637	0,5626	0,7574	0,6456	0,8982
CC 50 neurones	0,8638	0,5798	0,7461	0,6525	0,8986
CC 200 n, CC 200 n	0,8561	0,6257	0,6924	0,6574	0,8922
CC 200 n, CC 50 n	0,8660	0,5391	0,7863	0,6396	0,9018
CC 50 n, CC 200 n	0,8655	0,5621	0,7657	0,6483	0,9003
CC 50 n, CC 50 n	0,8656	0,6211	0,7293	0,6709	0,9020
CC 10 n, CC 10 n	0,8644	0,5988	0,7370	0,6607	0,8976
CC 50 n, CC 50 n, CC 50 n	0,8598	0,5772	0,7305	0,6449	0,8939
CC 50 n, CC 50 n, CC 5 n	0,8637	0,5922	0,7381	0,6571	0,9011

Tableau 12 : Les scores des différents modèles de réseau de neurones

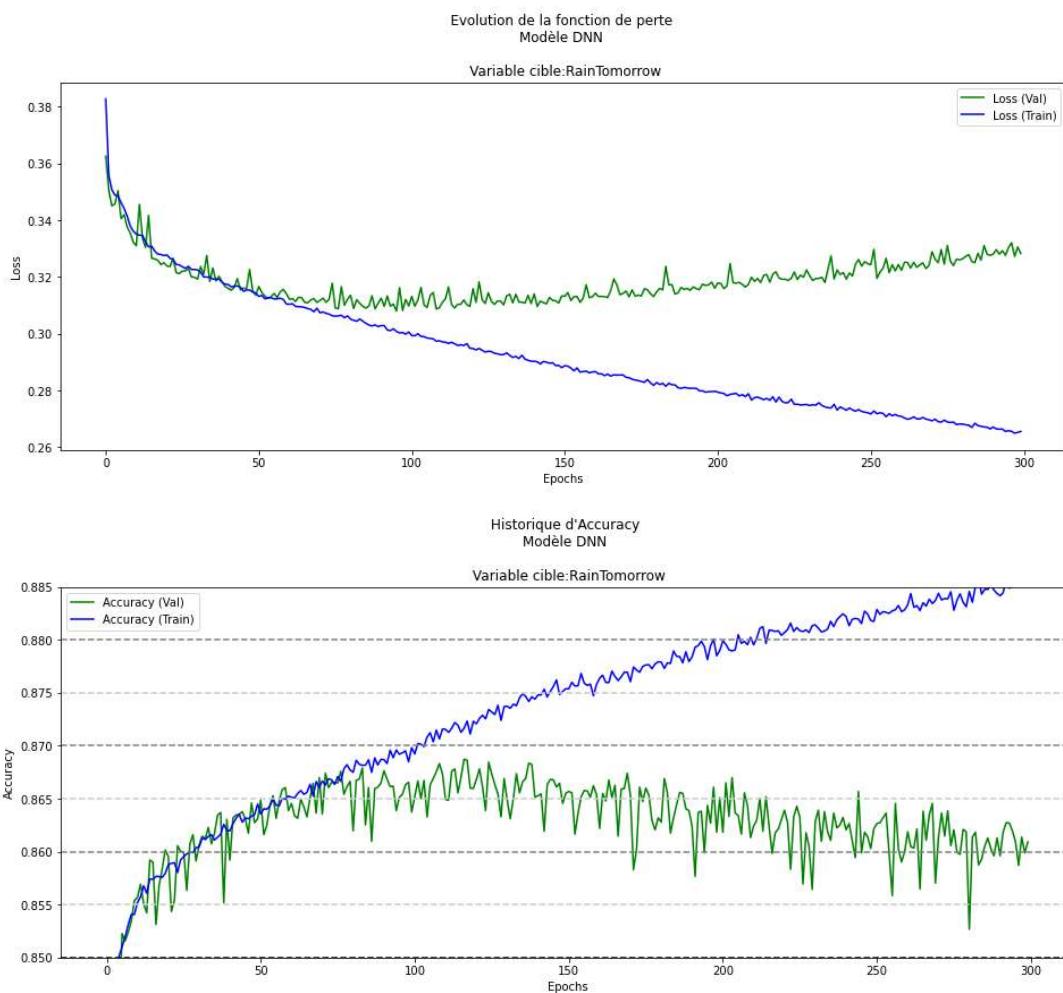
Toutes ces métriques sont souvent très proches. Malgré des réseaux très différents, allant d'un perceptron à un réseau dense de 3 couches de 50 neurones par couche, les performances ne se distinguent parfois que par la troisième décimale de la métrique observée. Il semble de prime abord difficile de trancher sur une topologie de réseaux en se basant uniquement sur ces métriques. Et pour cause : le comportement de ces différents réseaux évolue différemment tout au long de ces 300 époques. Certains d'entre eux divergent même après quelques dizaines d'époques seulement, indiquant qu'il faudrait stopper l'apprentissage et ne justement surtout pas le laisser perdurer sur 300 époques. En particulier, de façon systématique, un trop grand nombre de neurones entraîne rapidement un accroissement de la fonction de perte en validation, voire une baisse de l'accuracy sur l'échantillon de validation. Il en va de même lorsque nous introduisons la troisième couche. Les réseaux à une seule couche cachée ne semblent pas souffrir de ces défauts, mais apprennent moins bien que ceux possédant une seconde couche cachée.

La conclusion de l'observation de ces courbes d'apprentissage ne nous permet donc que d'écartier timidement certains modèles. En effet, les courbes nous montrent surtout que la présence d'un grand nombre de neurones ou celle d'une troisième couche cachée rend le réseau beaucoup plus sensible au surapprentissage, et nous poussera donc à redoubler de vigilance sur le nombre d'époques pour ce type de réseau. On peut voir sur la courbe d'accuracy du réseau (200,200) (Figure 21 : Evolution de la fonction de perte pour (200,200) – l'apprentissage diverge au-delà de l'époque 60 ) que l'accuracy maximale atteinte lors de l'apprentissage vers l'époque 60 est très légèrement plus faible que celle du réseau (50,50). On pourrait donc privilégier ce dernier. Toutefois, étant donné qu'aucun hyperparamètre n'a été encore optimisé à ce stade, il est prématûré de tirer des conclusions définitives sur le réseau optimal.

Pour la suite, nous faisons le choix de conserver un réseau de deux couches cachées de 50 neurones chacune car ce modèle offre des performances intéressantes tout en étant assez stable et moins sensible au surapprentissage que des réseaux plus complexes. De plus, nous avons également réalisé les travaux de recherche de paramètres optimaux décrits ci-après sur des réseaux plus complexes sans obtenir *in fine* de meilleures performances y compris avec des entraînement sur plusieurs milliers d'époques.



**Figure 20 : Evolution de la fonction de perte pour (50) – bon apprentissage**



**Figure 21 : Evolution de la fonction de perte pour (200,200) – l'apprentissage diverge au-delà de l'époque 60**

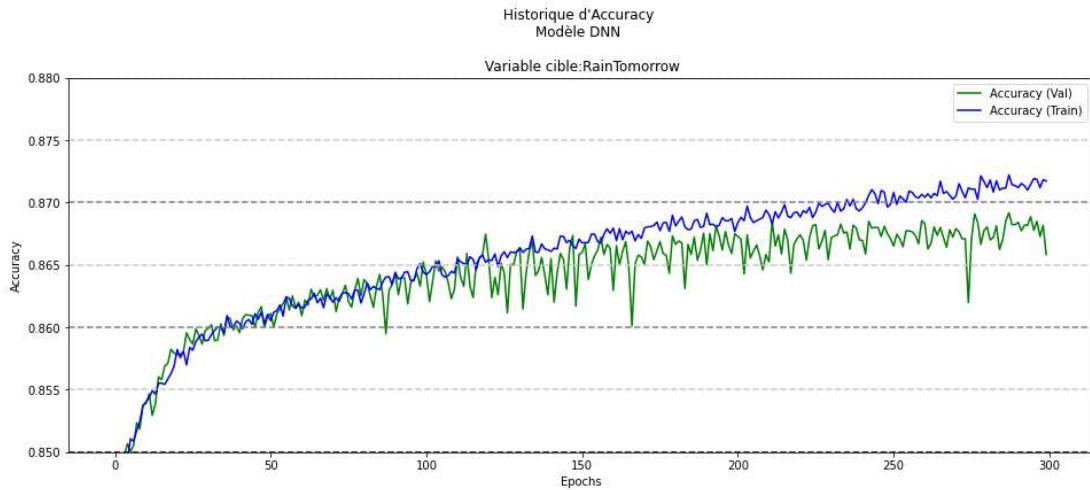


Figure 22 : Evolution de l'accuracy pour (50,50) – overfitting croissant à partir de l'époque 150

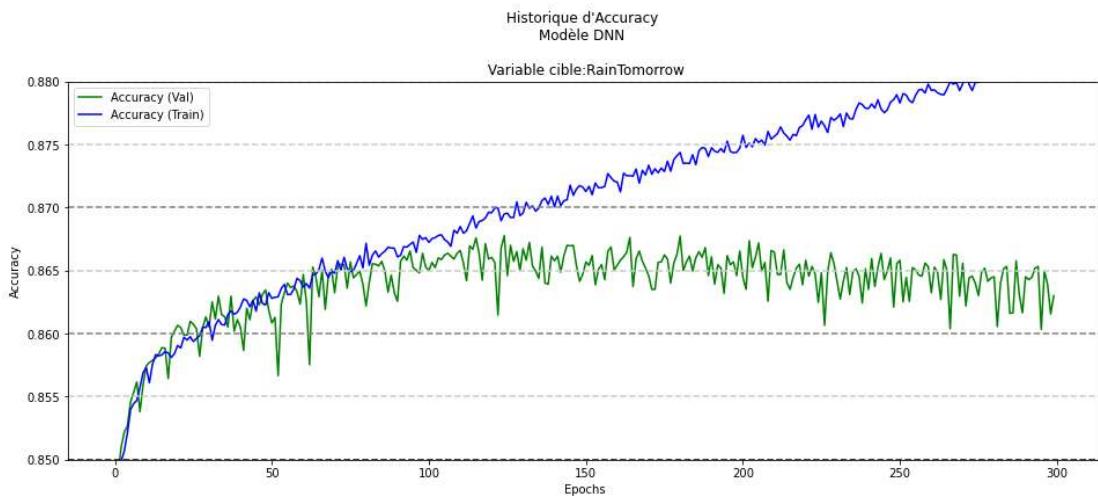
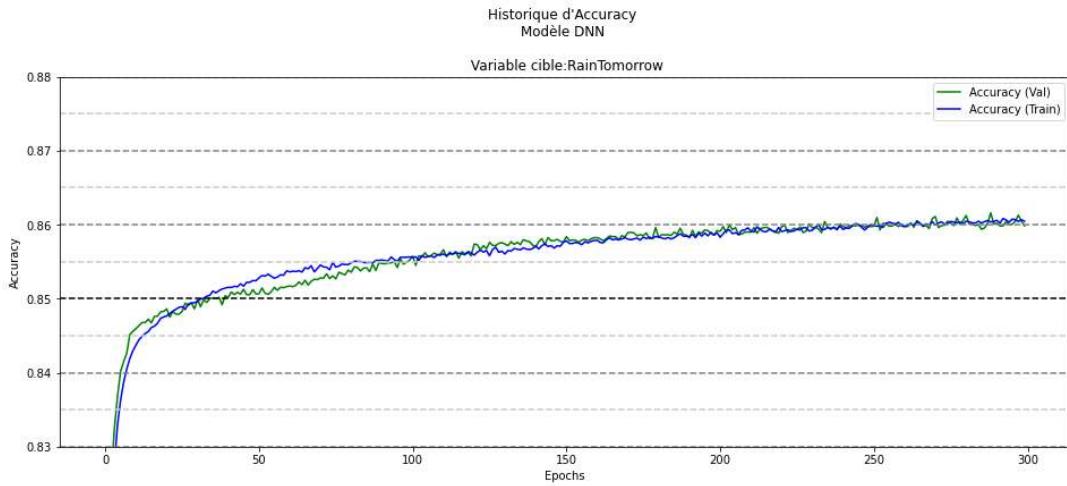


Figure 23 : Evolution de l'accuracy pour (50,50,50) – overfitting entraînant une dégradation de la validation dès l'époque 110

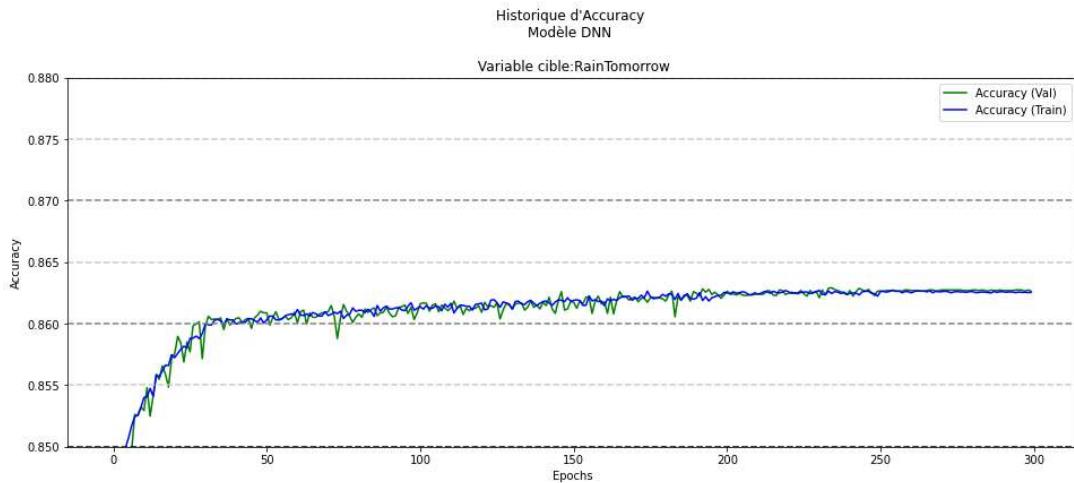
#### 2.4.1.2 Learning Rate

Le taux d'apprentissage de  $10^{-3}$  utilisé ci-dessus n'est pas assez fin : nous voyons des sauts assez brusques d'une époque à l'autre. L'apprentissage avec un taux plus faible ( $10^{-4}$ ) ne présente plus ces défauts. En revanche, nous voyons qu'il faudrait davantage d'époques pour que le modèle converge. Un taux de  $10^{-5}$  ne permet pas même d'atteindre la limite de 0,850 à l'issue des 300 époques.

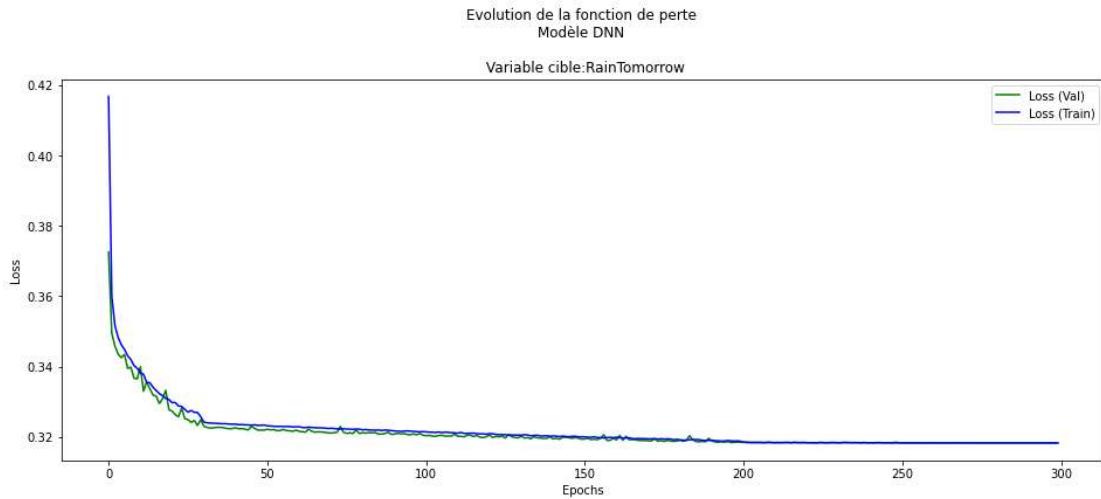


**Figure 24 : Learning Rate de  $10^{-4}$**

Nous allons ajouter une callback via `LearningRateScheduler` qui adaptera le learning rate en fonction de l'époque. Nous allons débuter par un learning rate de  $10^{-3}$  pendant les 30 premières époques afin d'atteindre rapidement une accuracy de 0,86. Ensuite, nous allons passer sur une accuracy de  $10^{-4}$  (Figure 24 : Learning Rate de  $10^{-4}$ ) pour poursuivre l'apprentissage de façon plus précise jusqu'à l'époque 200, au-delà de laquelle nous passons à  $10^{-5}$ . Enfin, nous finaliserons sur les 50 dernières époques sur un learning rate de  $10^{-6}$  seulement. La courbe d'apprentissage ainsi obtenue converge de façon harmonieuse (Figure 25 : Learning rate dynamique avec callback). A dire vrai, il serait même possible d'arrêter l'apprentissage dès l'époque 200.



**Figure 25 : Learning rate dynamique avec callback**



**Figure 26 : Fonction de perte avec Learning rate dynamique**

La fonction de perte évolue cette fois sans pics, en convergeant rapidement vers son minimum.

Ce learning rate dynamique ne permet pas directement d'apporter un gain sur les résultats, car il suffirait d'entraîner sur beaucoup plus d'époques un modèle avec un faible taux d'apprentissage. Mais il nous permet d'obtenir les résultats optimums des modèles avec beaucoup moins d'époques, et donc de temps de calcul lors de l'apprentissage.

#### 2.4.1.3 Fonctions d'activation

Comparaison des fonctions d'activation					
	accuracy	recall	precision	f1	auc
Tanh / Tanh	0,8631	0,5617	0,7551	0,6442	0,8957
ReLU / ReLU	0,8621	0,5594	0,7518	0,6415	0,8970
ReLU / Tanh	0,8629	0,5664	0,7511	0,6458	0,8958
Tanh / ReLU	0,8637	0,5588	0,7598	0,6440	0,8967

**Tableau 13 : Comparaison des fonctions d'activation**

Les historiques nous montrent que l'apprentissage d'un réseau ReLU/ReLU (la Figure 27) est plus dispersé qu'un apprentissage Tanh / Tanh (Figure 27-Figure 28), permettant potentiellement de sortir de minimum local. C'est un constat intéressant qui nous a ensuite incité à tester des combinaisons ReLU / Tanh et Tanh / ReLU. Bien nous en a pris : il semble que cette dernière configuration nous permette un léger gain. Nous conserverons donc ensuite un réseau avec une première couche de 50 neurones activés par Tanh et une seconde couche de 50 neurones activés par ReLU.

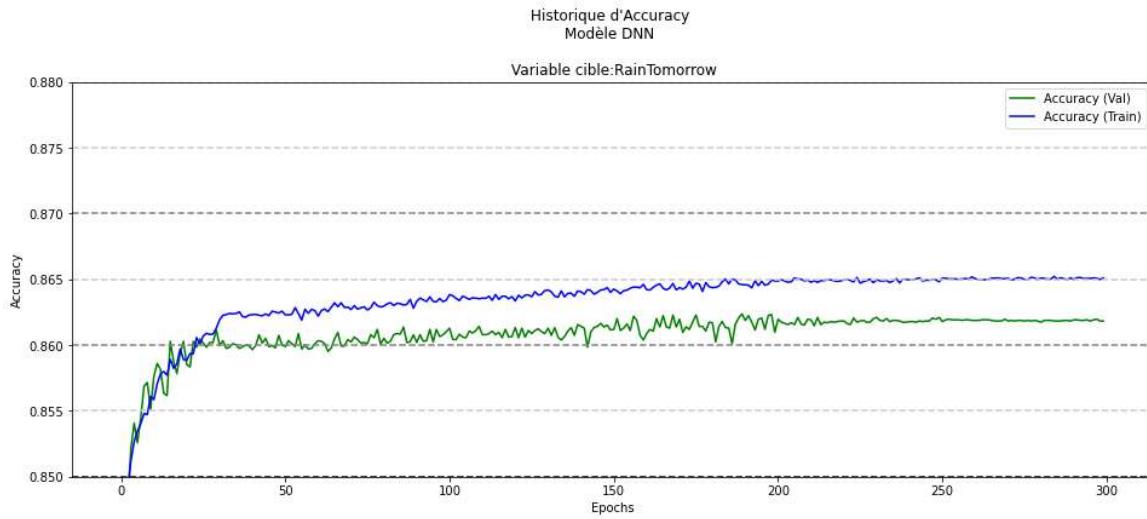


Figure 27 : Accuracy du réseau ReLU / ReLU

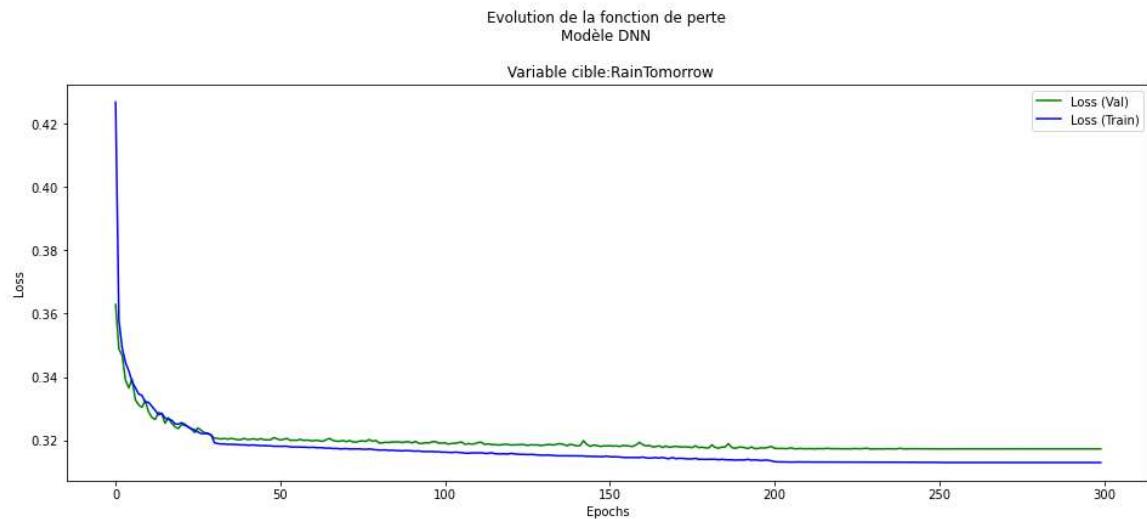


Figure 28 : Loss du réseau ReLU / ReLU

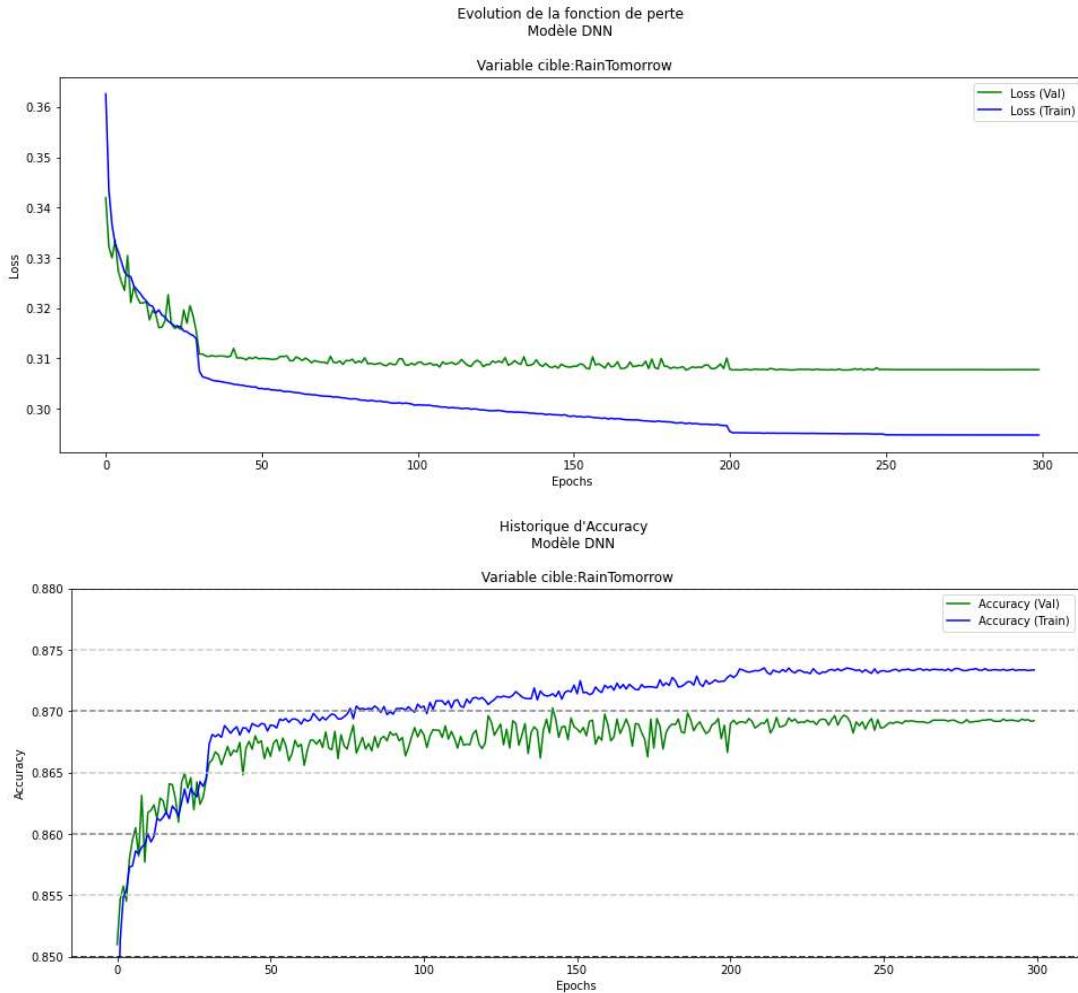
#### 2.4.1.4 Batch size

Nous avons pris un batch size relativement grossière de 512 échantillons. Le Tableau 14 représente l'impact d'une variation de cette quantité d'une part sur les métriques, mais également sur les temps de calculs.

Variation du batch size						
Batch_size	accuracy	recall	precision	f1	auc	Temps
2048	0,8597	0,5567	0,7430	0,6365	0,8914	0 mn 33s
512	0,8637	0,5588	0,7598	0,6440	0,8967	1 mn 23s
128	0,8660	0,5655	0,7656	0,6505	0,9006	3 mn 48
16	0,8666	0,5767	0,7609	0,6561	0,9024	13 mn 25s

Tableau 14 : Variation du batch\_size

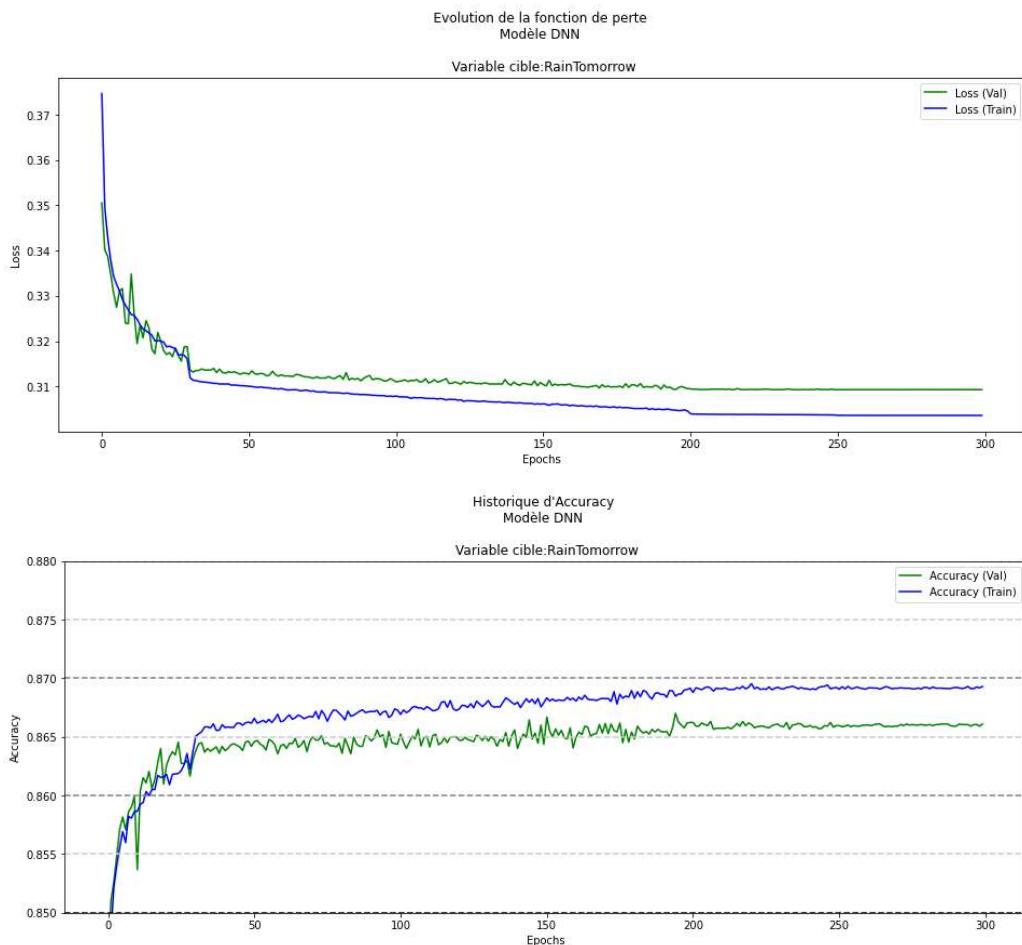
Le modèle avec *batch\_size* de 16 est le plus performant, mais également de loin le plus lent. On remarque une différence relativement marquée entre les échantillons de train et de validation, tant pour l'évolution de la *loss* que de l'*accuracy*. Il est amusant de remarquer qu'on peut constater une baisse de la *loss* à chaque changement de learning rate par notre callback aux époques 30, 200 et 250. Au final, étant donné le faible écart de performances, le temps bien plus important pour entraîner le modèle de *batchsize*=16 et l'écart entre l'échantillon de train et de test, notre préférence se portera sur un *batch\_size* de 128.



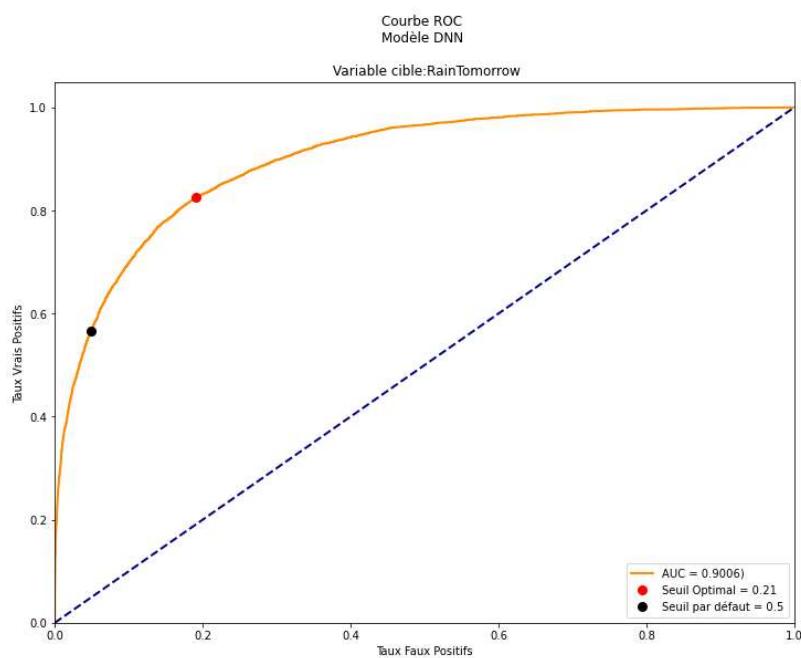
**Figure 29 : Evolutions de l'apprentissage pour modèle avec batch\_size = 16**

#### 2.4.1.5 Comparaison des performances

Maintenant que nous avons obtenu notre meilleur modèle de DNN (learning rate dynamique via notre callback personnalisée, batch size=128, 1<sup>ère</sup> couchée cachée de 50 neurones avec fonction d'activation tanh, 2<sup>ème</sup> couche cachée de 50 neurones avec fonction d'activation ReLU, 300 époques), observons ses courbes d'apprentissage présentées dans la Figure 30, puis comparons ses performances avec notre meilleur XGBoost. Nous allons profiter du fait que notre neurone de sortie applique une fonction continue pour afficher sa courbe ROC. Comme nous l'avons fait précédemment, nous regarderons l'impact de la variation du seuil de classification dans les métriques.



**Figure 30 : Evolutions de l'apprentissage de notre modèle final**



**Figure 31 : Courbe ROC du modèle DNN final**

<b>Comparaison du DNN avec XGBoost</b>					
	<b>accuracy</b>	<b>recall</b>	<b>precision</b>	<b>f1</b>	<b>auc</b>
XGBoost - Seuil par défaut (0,50)	0.8642	0.5558	0.7642	0.6436	0.9003
XGBoost - Seuil Optimal (0,22)	0.8196	0.8140	0.5630	0.6656	0.9003
DNN – Seuil par défaut	0,8660	0,5655	0,7656	0,6505	0,9006
DNN – Seuil Optimal	0,8131	0,8263	0,5510	0,6611	0,9006

Avec le seuil par défaut, le DNN est plus performant que le XGBoost sur toutes les métriques, mais dans des proportions infimes. Avec le seuil optimal, le DNN est meilleur sur le recall et l'AUC, là aussi de façon peu significative.

#### 2.4.1.6 Interprétabilité

Si *Humidity3pm* reste une feature importante, remarquons que *Pressure3pm* revêt aux yeux du DNN une importance bien plus élevée qu'avec nos modèles XGBoost. L'appartenance aux zones climatiques, la latitude et la longitude ont une importance notable.

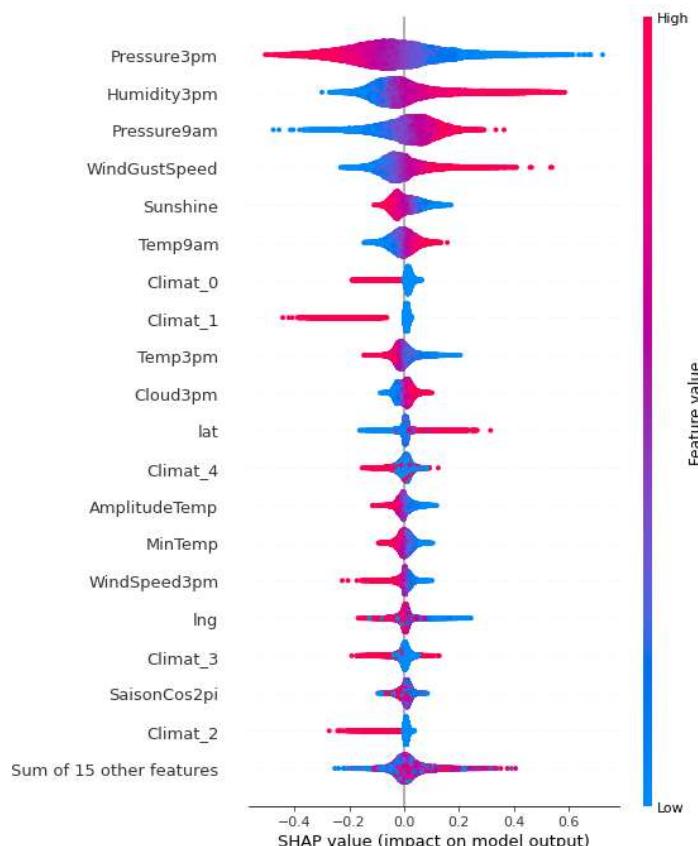


Figure 32 : beeswarm des valeurs de Shapley de notre réseau dense

Regardons le poids des variables explicatives pour quatre prédictions par notre DNN représentés dans la Figure 33

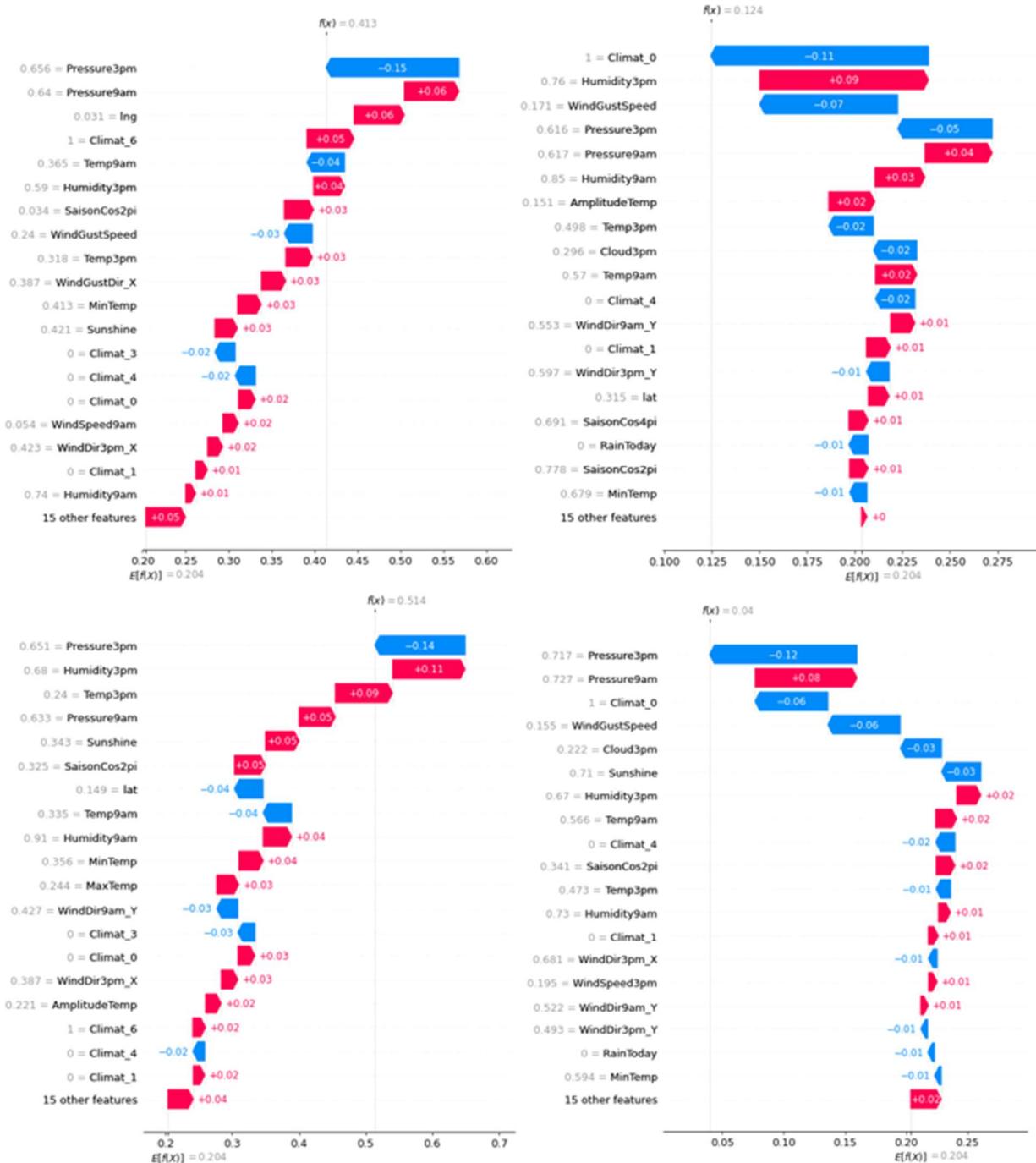


Figure 33: Exemples de quatre prédictions par le modèle DNN

Dans la première prédition, *Pressure3pm* pèse beaucoup pour indiquer qu'il ne pleuvra pas, contrairement à *Pressure9am*. La latitude et l'appartenance à la zone climatique de la côte sud jouent en la faveur de la pluie également.

C'est l'appartenance à la zone climatique de la côte Est qui a le plus de poids dans notre seconde prédition en haut à droite, devant l'*Humidity3pm* qui, elle, incite à prévoir la pluie.

Les variables de saisonnalité *SaisonCos2pi* et *SaisonCos4pi* n'influent que modérément les décisions. Bien que le poids des features explicatives varie beaucoup d'une prédition à l'autre, nous remarquerons que *Pressure3pm* est en tête de trois d'entre elle dans l'importance de la prise de décision.

#### 2.4.1.7 Conclusion

L'entraînement de nos DNN a été parfois très long lors de nos tâtonnements. A de nombreuses reprises, nous avons entraîné des modèles prenant plusieurs dizaines de minutes, voire plus d'une heure. Les XGBoost, quant à eux, ne prennent chaque fois que quelques secondes à être entraînés. De plus, il y a un nombre bien plus important de paramètres à définir sur un DNN que sur un XGBoost, ce qui implique une recherche également plus importante en temps humain. Certes, notre DNN offre légèrement de meilleures performances que notre XGBoost, mais, dans le cadre d'un projet avec un budget défini, cette expérience nous montre qu'il faudra être particulièrement attentif sur le temps total que nous serons prêts à consacrer à un DNN et qu'il faudra bien peser le rapport du bénéfice de performances par rapport au coût de développement.

### 2.4.2 RNN

#### 2.4.2.1 Prédiction monovariée

Jusque-là, pour effectuer la prédiction à  $J+1$ , nous ne disposions que des relevés météo à  $J$ . Les réseaux récurrents vont nous permettre de bénéficier également de l'apport des relevés météorologiques de plusieurs jours précédents la prédiction. Une des difficultés consistera d'ailleurs à déterminer le nombre de journées optimum à reprendre.

Contrairement aux DNN et aux modèles de machine learning classiques, l'entraînement d'un RNN nécessite que les données d'entraînement soient triées chronologiquement et qu'il n'y ait que peu de trous dans la chronologie.

Cette chronologie implique que notre RNN ne pourra être entraîné qu'au niveau micro, et non sur des zones climatiques ou sur l'ensemble de l'Australie, puisqu'une seule observation ne sera utilisée par jour.

Les modèles monovariés de prédiction de RainTomorrow donnent des résultats particulièrement mauvais : quels que soient les propriétés de notre RNN, les résultats sont certes meilleurs qu'un tirage aléatoire mais sans commune mesure avec les résultats obtenus précédemment avec nos modèles de machine learning classique. Et pour cause : n'oublions pas que contrairement aux modèles précédents, nous n'exploitons que la seule feature RainTomorrow. Nous ne bénéficions donc plus de l'apport des autres features. Or, cette variable ne présentant que peu de régularité, nous comprenons ces résultats médiocres.

Il semble indispensable de disposer d'un modèle multivarié pour prédire RainTomorrow avec un RNN.

#### 2.4.2.2 Prédiction multivariée

Dans le rapport final, nous développerons une modélisation RNN multivariée. Nous espérons que cette approche nous permettra non seulement de proposer des résultats satisfaisants, contrairement à l'approche monovariée, mais aussi qu'elle nous fera bénéficier d'un gain significatif par rapport au DNN.

## 3 Prédition de la pluie à un horizon de temps

### 3.1 Objectif et méthodologie

Nous avons tenté jusqu'ici de prédire s'il pleuvra à  $J+1$ . Voyons voir maintenant s'il est possible de prévoir la pluie sur davantage de jours dans le futur.

Pour cela, nous allons créer de nouvelles variables cibles, nommées  $Rain\_J\_h$ , où  $h$  est le nombre de jours dans le futur et dont  $Rain\_J\_h$  vaudra 1 s'il pleuvra dans  $h$  jours et 0 sinon.

Afin d'obtenir ces variables, nous partons de *RainToday*, à laquelle nous allons appliquer des shifts successifs à partir de notre Dataframe trié chronologiquement, après avoir rempli les dates manquantes

avec des données vierges. Ce décalage est appliqué pour chaque *Location*, et non sur le dataframe global pour que le shift n'impute pas la valeur de la dernière date d'une *Location* sur la valeur de la première date d'une autre *Location*.

*Rain\_J\_1* est donc égale à *RainTomorrow*, *Rain\_J\_2* indique s'il pleuvra après-demain, etc. Les taux de répartition entre les classes restent donc identiques.

Notre approche va consister à entraîner spécifiquement un modèle pour chaque variable cible *Rain\_J\_h*. Dans les graphes qui suivront, lorsque nous regardons l'évolution des qualités de prédiction suivant le nombre de jours dans le futur prédit, il s'agira donc d'autant de modèles qu'il y a de jours distincts prédits. Nous utiliserons des XGBoost avec les mêmes hyperparamètres que vus précédemment, en fonction de la granularité micro, macro ou climatique.

### 3.2 Limite théorique

Les sites de prévision météorologique proposent des prédictions maximales sur 2 semaines. Nous nous attendons donc que la qualité des prédictions se réduisent progressivement jusqu'à ne plus pouvoir se distinguer du hasard avant cette limite théorique de 15 jours. Observons ici l'accuracy, le recall et l'AUC-ROC pour des modèles entraînés sur l'ensemble de l'Australie. Dans la Figure 34, nous avons conservé le seuil par défaut de 0,5. Dans la Figure 35, nous avons pris le seuil optimal au regard de la courbe ROC. Regardons comment les métriques évoluent en fonction du nombre de journées dans le futur de prédictions :

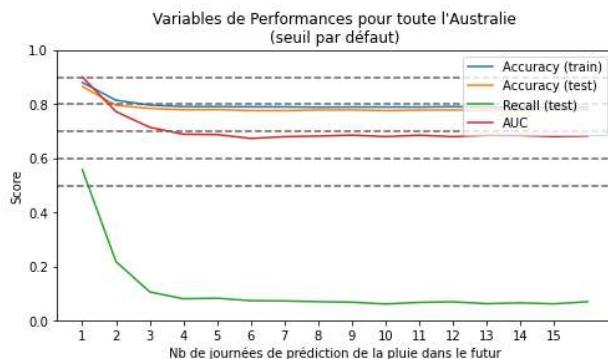


Figure 34: Scores du modèle macro, seuil par défaut

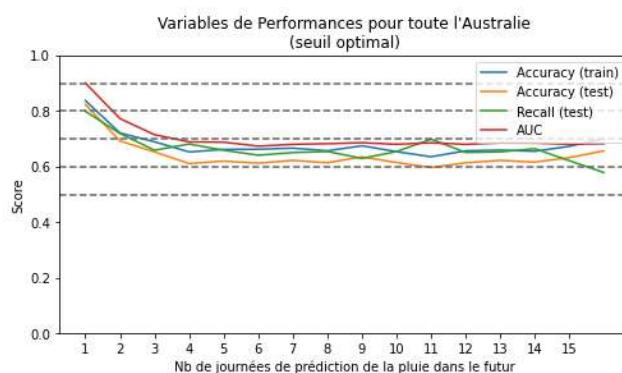


Figure 35 : Scores du modèle macro, seuil optimal

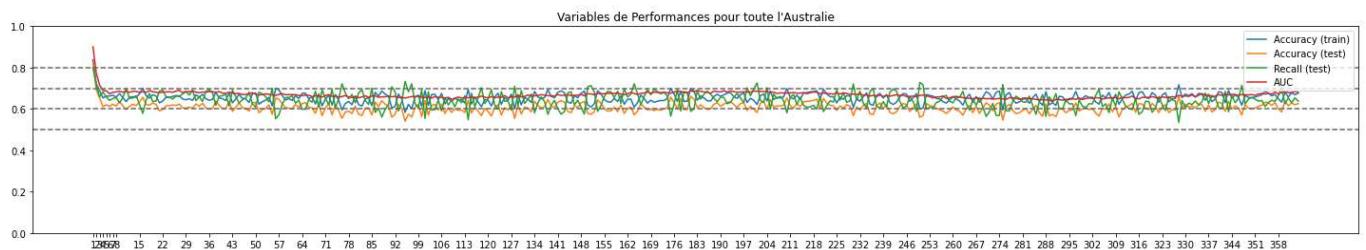
Plusieurs constats s'imposent :

- L'accuracy faiblit rapidement dans les deux cas les quatre premiers jours
- Elle converge vers une valeur légèrement inférieure à 0,8 avec le seuil par défaut (rappelons qu'il y a un taux de journées non pluvieuses de 0,77), et un peu supérieure à 0,6 avec le seuil optimal

- Le recall s'effondre à moins de 0,1 là aussi avec le seuil par défaut, ce qui fait supposer que les modèles ne doivent que rarement prédire de la pluie à partir de J+4
- Avec le seuil optimal, le recall reste supérieur à 0,6 et montre qu'on arrive encore à capter plus de 60% des journées pluvieuses à J+15 dans nos prédictions positives
- Enfin, et surtout, l'AUC ne converge pas vers 0,5, mais plutôt vers une valeur proche de 0,7 !

Ce dernier constat est particulièrement surprenant : si, comme nous l'avions supposé, la qualité des prédictions s'étaient dégradée jusqu'à devenir impossible, nous aurions dû avoir une AUC-ROC qui aurait dû se rapprocher de 0,5, de sorte à être comparable avec des prédictions aléatoires.

Plus surprenant encore : même en élargissant nos prédictions sur une année, l'AUC reste très au-dessus de 0,5. Nous représentons ci-après les métriques utilisant le seuil optimal de chaque modèle :



**Figure 36 : Métriques de performance de 360 modèles entraînés chacun à prédire la pluie à J+n**

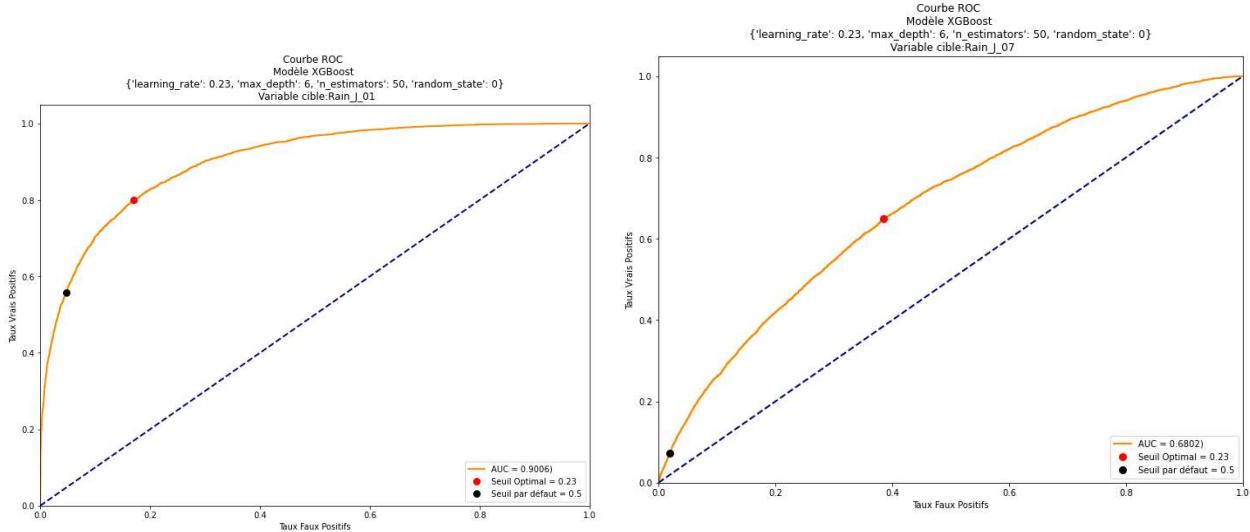
Bien entendu, l'accuracy étant d'environ 0,6 pour des prévisions dès J+4, nous avons bien conscience de la faible qualité de ces prédictions. Il n'empêche qu'il reste étonnant de constater que les seuls relevés météorologiques d'un jour donné permettent de prédire mieux que le hasard s'il pleuvra un certain nombre de journées dans le futur.

### 3.3 Comportement détaillé

Essayons de comprendre nos modèles. La courbe ROC de gauche illustre les résultats à J+1 (ce qui correspond donc à *RainTomorrow*). La courbe de droite trace les résultats à J+7. Les différences entre ces deux courbes sont représentatives de l'évolution dans le temps :

- la courbe ROC s'aplatit rapidement les 4 premiers jours
- le point noir, représentant le seuil par défaut, se rapproche de l'origine
- le point rouge, représentant le seuil optimal, reste au centre de la courbe

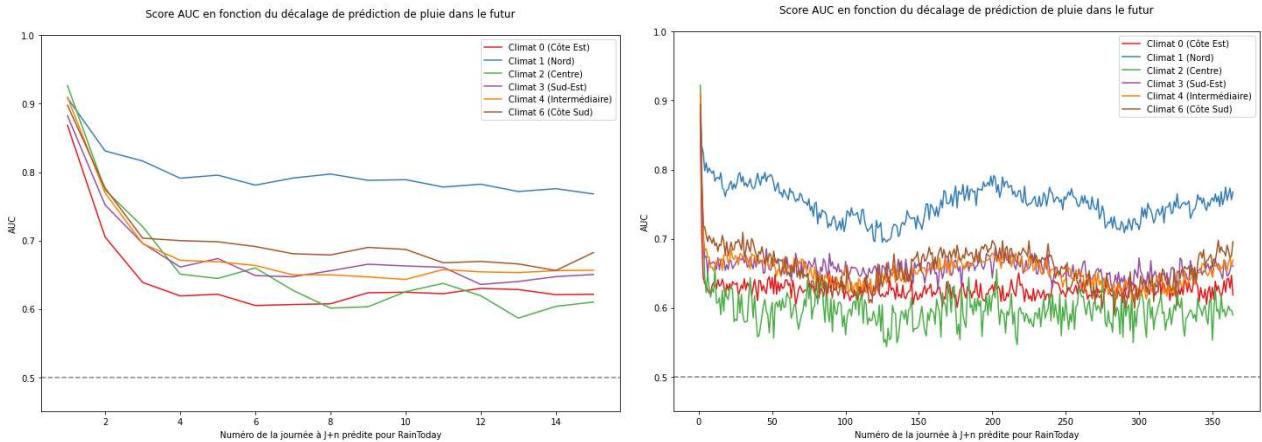
Ces simples visualisations corroborent les métriques vues plus haut : en conservant le seuil par défaut, nous allons très vite prédire presque systématiquement qu'il ne pleuvra pas. En revanche, le seuil optimal nous permet d'améliorer d'une façon frappante le taux de vrais positifs



### 3.4 Analyse par zone climatique

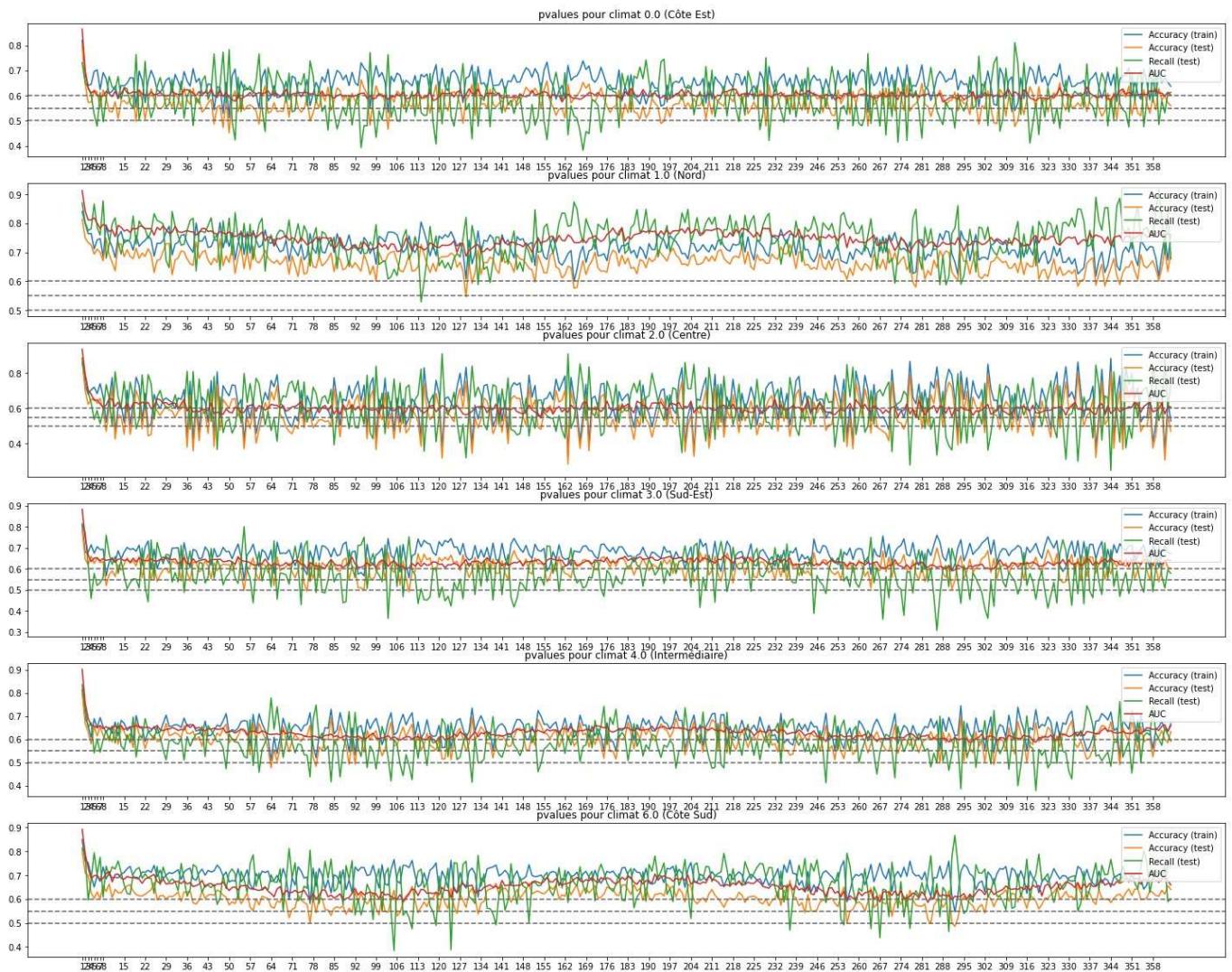
Le graphique de gauche ci-dessous nous présente l'évolution de l'AUC-ROC pour chaque zone climatique pour les 15 premiers jours de prédiction. Le graphique de droite les représente sur une année.

Sur les deux premières semaines, nous pouvons voir des différences dans l'évolution de l'AUC selon les zones climatiques. La zone septentrionale conserve un AUC proche de 0,8 après deux semaines, alors que la zone du désert central baisse plus rapidement vers 0,6.



Lorsqu'on observe les métriques sur une plage de prédiction d'un an, les différences de prévisibilité à long terme se confirment selon les zones climatiques. Notez qu'on remarque une saisonnalité de l'AUC. Il s'agit d'un phénomène que nous avons remarqué de façon encore plus marquée pour la prévision de la température maximale dans le futur. Cette tendance saisonnière semble suivre une courbe ayant deux cycles au cours de l'année, c'est pourquoi nous avons ajouté une variable temporelle SaisonCos4pi.

Eclatons cette représentation sur une année pour chaque zone climatique, et enrichissons-là de l'accuracy (avec les données d'entraînement et de test, afin de repérer d'éventuels overfitting) et du recall :

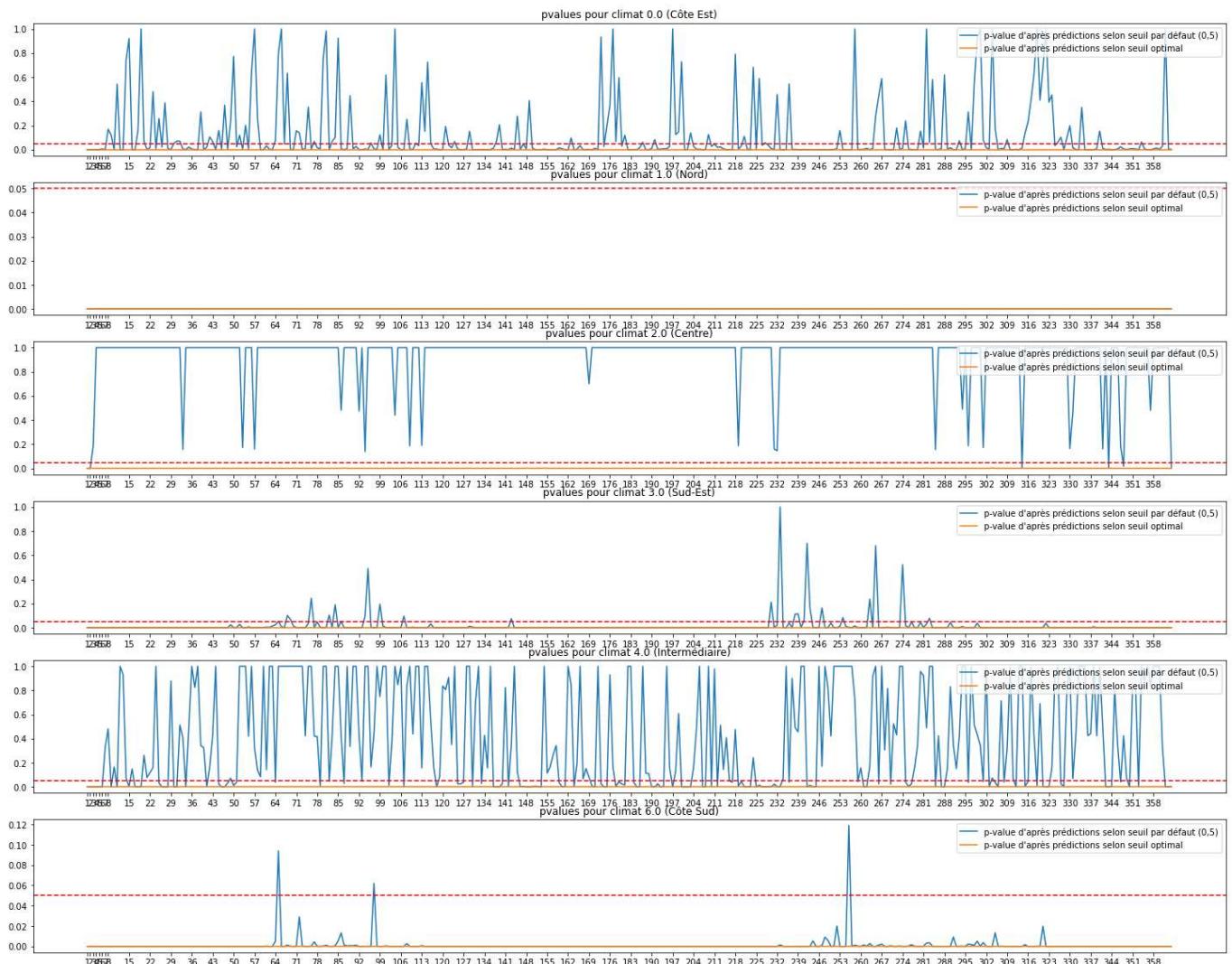


**Figure 37 : Métriques de 360 modèles prédisant la pluie à J+n pour chaque zone climatique**

Nous pouvons légitimement nous demander si ces modèles sont finalement plus performants que le hasard de façon systématique ou non. On effet, nous savons qu'un score AUC de 0,5 correspond à des prédictions aléatoires, mais est ce que nos scores souvent proches de 0,6 sont réellement pertinents ?

Pour éclairer ce point, nous allons réaliser des tests Chi<sup>2</sup> entre les valeurs observées et les valeurs prédites de la variable cible, pour chaque journée de prédiction dans le futur, et nous allons afficher la pvalue obtenue. Notre hypothèse nulle est que nos prédictions ne sont pas corrélées avec la réalité.

Les courbes bleues dans la Figure 38 issues des prédictions avec le seuil par défaut, montrent que H0 ne peut souvent pas être rejetée, car étant supérieur au seuil de 0,05. En revanche, les courbes orange de prédictions tenant compte du seuil optimal sont systématiquement inférieures à 0,05. Les prédictions prenant en compte le seuil optimal prédisent donc véritablement mieux que le hasard d'une façon significative.

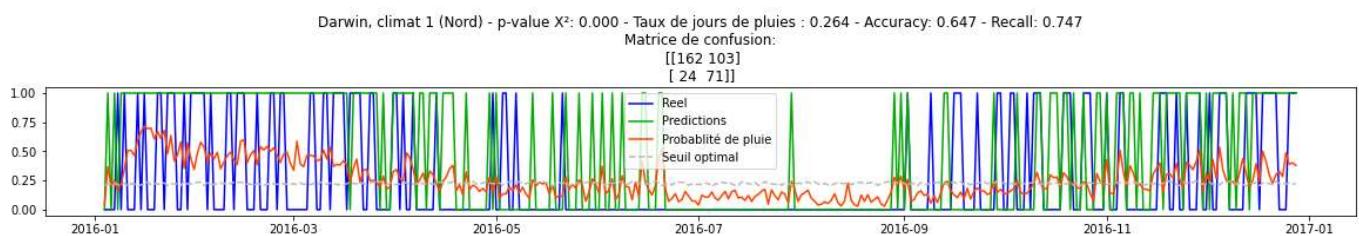


**Figure 38 : pvalue issues du test  $\chi^2$  de corrélation entre les prédictions de 360 modèles prédisant la pluie à  $J+n$  et les observations à ces dates, pour chaque zone climatique**

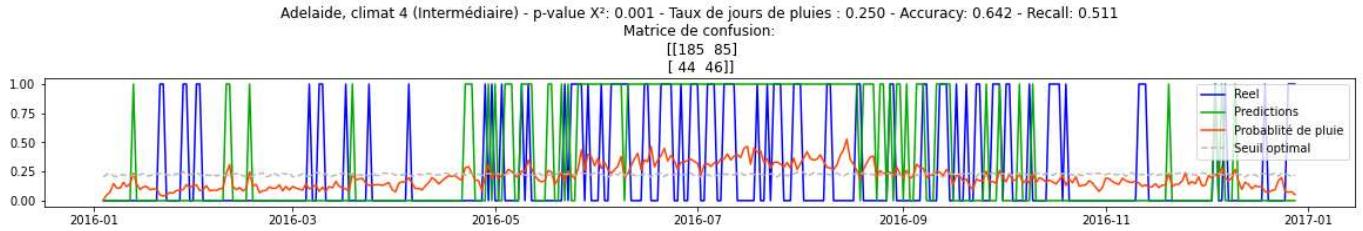
### 3.5 Prédictions

Puisqu'il semble possible de prédire s'il pleuvra ou non un an à l'avance, jetons-nous à l'eau sur quelques cas concrets.

Les graphiques ci-après montrent les résultats de la prédiction d'une journée pluvieuse ou non sur l'ensemble de l'année 2016 avec comme unique observation les features du 1<sup>er</sup> janvier 2016 ! Les 360 modèles entraînés pour effectuer cette prédiction l'ont bien entendu été uniquement avec des données antérieures à 2016.



**Figure 39 : Prédictions de la pluie sur l'année 2016 pour Darwin à partir de la seule observation du 1<sup>er</sup> janvier 2016**



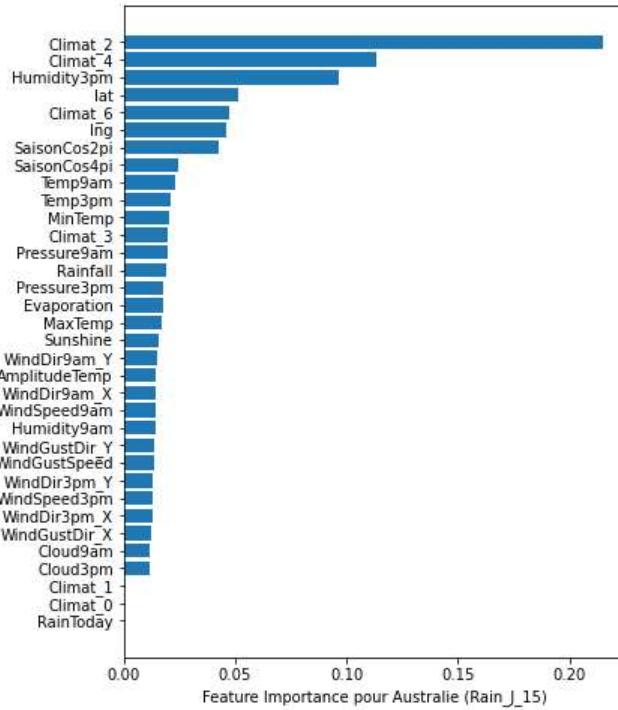
**Figure 40: Prédiction de la pluie sur l'année 2016 pour Adelaide à partir de la seule observation du 1<sup>er</sup> janvier 2016**

Nous avons retenu ici deux villes, Darwin et Adélaïde, situées dans des zones climatiques différentes. Dans les deux cas, on voit que le modèle prévoit qu'il pleuvra sur une vaste période au cours de laquelle les épisodes réellement pluvieux ont effectivement été nombreux (de janvier à avril pour Darwin, de juin à septembre pour Adélaïde). Les modèles semblent aussi avoir capté la saison sèche de Darwin de juillet à septembre. Bien sûr, ces modèles comprennent nos deux features de temporalité, SaisonCos2pi et SaisonCos4pi, mais même sans ces informations temporelles les modèles sont capables d'identifier ces tendances à partir des celles observations du 1<sup>er</sup> janvier.

Les courbes de pluie réelle et de prédiction sont cependant loin de se confondre, et de nombreuses erreurs sont visibles, ainsi que l'attestent les métriques indiquées dans le titre de ces deux graphiques.

### 3.6 Interprétabilité

Nous avons vu que pour RainTomorrow, les features les plus importantes étaient *Humidity3pm*, *Sunshine*, *Cloud3pm*, *WindGustSpeed*. Les features que nous avions ajoutées étaient exploitées de façon modérée par les modèles. La situation change considérablement lorsque nous regardons les variables exploitées par XGBoost pour prédire s'il pleuvra plusieurs jours dans le futur. Regardons ci-dessous les features importance sur un modèle entraîné sur toute l'Australie pour déterminer s'il pleuvra à J+15 :



**Figure 41: Feature Importance de la prévision de la pluie à J+15 avec un modèle macro**

L'appartenance aux zones climatiques 2 (Centre), 4 (Intermédiaire) et dans une moindre mesure 6 (Côte Sud) est une information importante. Nous retrouvons d'autres features que nous avons créées parmi les plus importantes, telles la latitude et la longitude, ainsi que nos deux variables de temporalité SaisonCos2pi et SaisonCos4pi. L'importance du feature engineering, qui était relativement modérée pour prédire RainTomorrow, semble bien plus marquée pour des prévisions plus lointaines.

De plus, si nous regardons cette fois les features exploitées sur un modèle micro (la Figure 42) nous verrons que des spécificités géographiques qui n'étaient pas déterminantes pour RainTomorrow deviennent particulièrement intéressantes ici. Sur ce modèle entraîné sur les données de Darwin uniquement pour prédire s'il pleuvra dans 100 jours, nous voyons par exemple que c'est WindDir3pm\_Y qui est la variable la plus importante. Or, Darwin est située tout au Nord de l'Australie, dans une zone touchée par la mousson : on comprend ici que connaître les vents venant du nord permet au modèle d'anticiper sur 100 jours si la mousson touchera la ville.

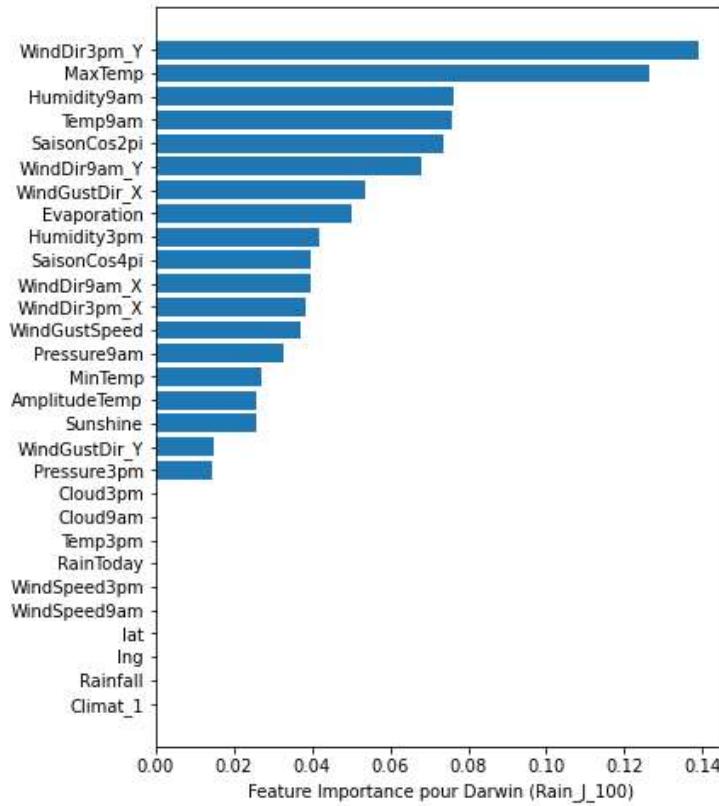


Figure 42 : Feature Importance de la prévision de la pluie à J+100 avec un modèle micro sur Darwin

### 3.7 Conclusions

Du fait des faibles scores sur l'accuracy, la prédiction plusieurs jours à l'avance avec des XGBoost n'est pas directement exploitable. Seules les prédictions jusqu'à J+3 sont exploitables en pratique. Cependant, savoir que les observations d'une journée donnée peuvent permettre de prédire s'il pleuvra un an à l'avance, même avec un taux d'erreur important, reste intéressant.

## 4 Prédiction de la variable *MaxTemp*

### 4.1 Présentation

Après nous être attelés à prévoir s'il pleuvra le lendemain, voire dans plusieurs jours, nous avons tenté de prévoir la température maximale du lendemain.

Nous avons choisi d'effectuer cette prédiction car il s'agit là d'une variable classiquement présentée dans les bulletins météorologiques, et elle présente l'intérêt d'être une variable non seulement continue mais saisonnière. Nous pourrons donc utiliser des approches de régressions et de séries temporelles.

### 4.2 Résultats de la régression par approches « classiques » via scikit-learn

Tout comme nous avions créé des variables *Rain\_J\_h* pour tenter de prédire s'il pleuvra dans h jours, nous avons créé dans notre dataset des variables *MaxTemp\_J\_h*. Regardons ici les performances obtenues par un XGBoost de regression pour prédire *MaxTemp*, c'est-à-dire prédire la température maximale de la journée à partir des autres features, mais aussi *MaxTemp\_J\_01*, c'est-à-dire la température maximale du lendemain à partir des features de la journée ainsi que la température dans 2 jours :

Prédiction des températures avec XGBoost		
	RMSE	MAE
MaxTemp	0,71	0,51
MaxTemp_J_01	2,65	1,93
MaxTemp_J_02	3,35	2,50

On voit qu'il y a beaucoup plus d'erreurs dans les prédictions de température du lendemain que dans celles de la journée même, et davantage encore dans celle du surlendemain. Comparons avec d'autres approches.

### 4.3 Séries Temporelles par SARIMAX

La courbe des températures présente une saisonnalité apparente. Nous pouvons donc tenter une modélisation monovariée avec SARIMAX.

Nous effectuerons là des analyses micro afin de tenter de prédire l'évolution des températures maximales pour une *Location* donnée. Prenons l'exemple de Mildura, qui a l'avantage de présenter une amplitude thermique assez importante au cours de l'année :

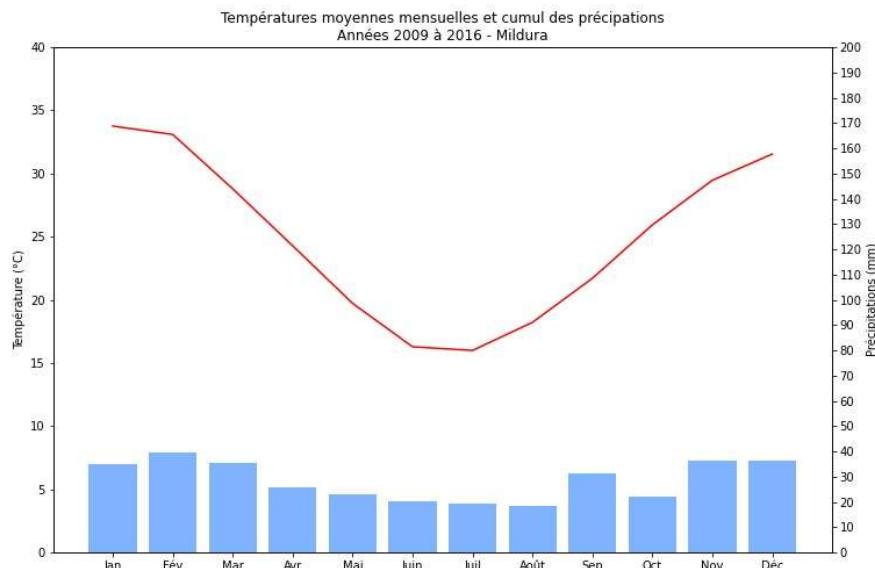
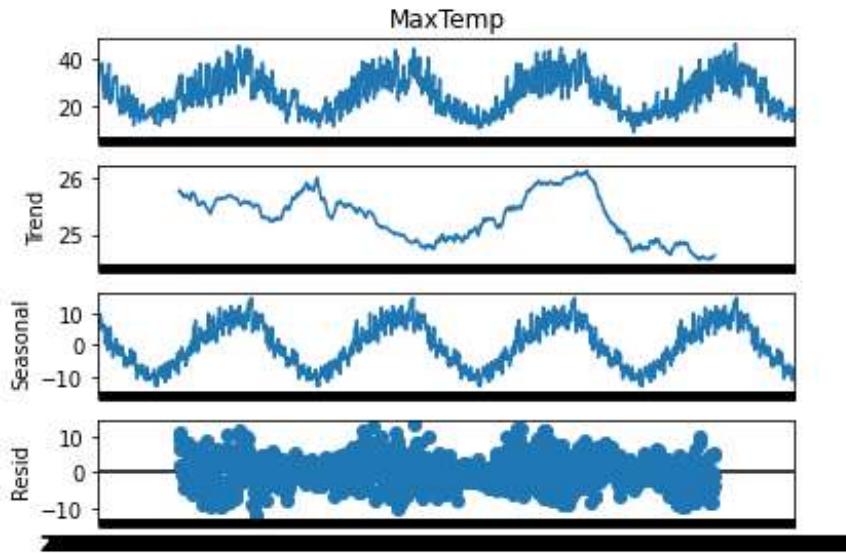


Figure 43 : Températures moyennes mensuelles et cumul des précipitations, années 2009 à 2016 - Mildura

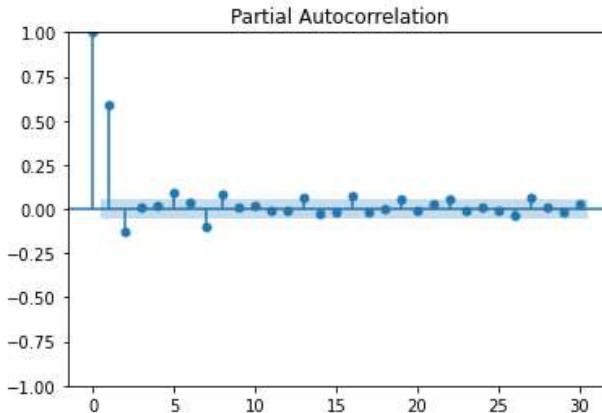
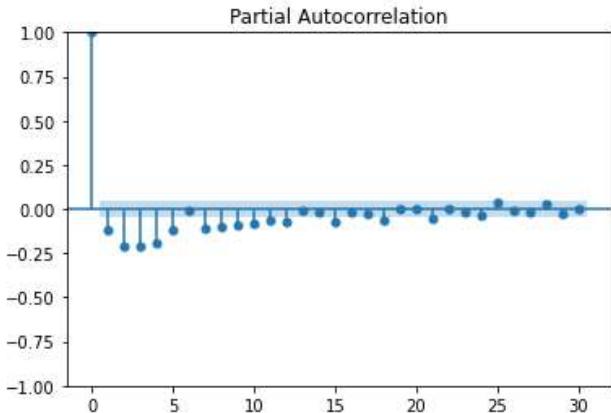
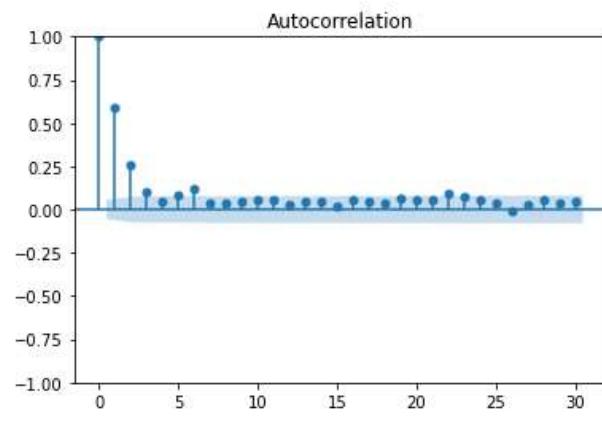
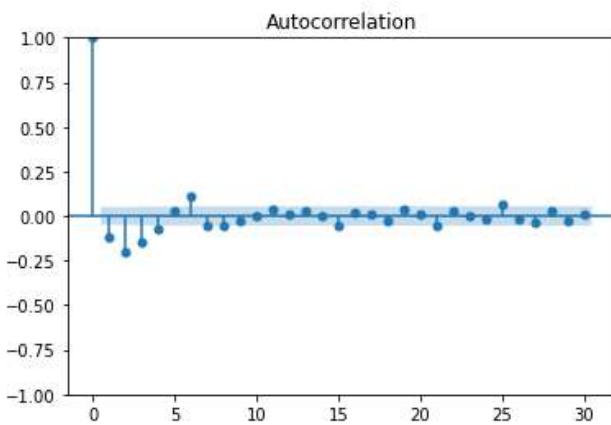
La décomposition de la variable *MaxTemp* nous apprend plusieurs choses :

- Il existe bien, comme nous nous en doutions, une saisonnalité, qui explique des variations d'une amplitude de 20°
- La tendance reste dans une fourchette de variation d'un peu plus de 1° seulement
- Les résidus sont étalés sur une amplitude de 20°



Si les deux premiers constats sont encourageants, le dernier l'est nettement moins : l'existence d'un bruit aussi important implique souvent des prévisions de qualité réduite.

Passons aux décompositions ACF et PACF, sur la base d'une différenciation d'une journée d'une part et de 365 jours (=un an) d'autre part :



**Figure 44 : ACF et PACF – différenciation d'un journée**

**Figure 45 : ACF et PACF – différenciation d'une année (=365 journées)**

La série est stationnaire après une différenciation d'une journée (graphes de gauche), de même que sur 365 journées (graphes de droite). Les décompositions ACF et PACF permettent de déterminer les valeurs max de  $(p,d,q)(P,D,Q)$  comme étant  $(5,1,4)(2,1,3)(365)$ .

Malheureusement, il nous a été impossible d'obtenir des résultats avec cette approche. En effet, quelle que soit la Location sélectionnée pour modéliser la variable *TempMax*, la modélisation n'était pas terminée après 12h de calculs, y compris lorsque nous avions restreint les données sur 4 ans seulement au lieu de 10.

## 4.4 Deep Learning

### 4.4.1 RNN

#### 4.4.1.1 Prédiction monovariée sur une journée

Nous avons développé ici un réseau récurrent afin de tenter de prédire la température d'une journée à l'aide de la température des jours précédents. L'analyse est ici monovariée.

Parmi les paramètres, le plus simple à déterminer a été le `batch_size` : dès lors qu'il était supérieur à 1, les performances étaient réduites. Nous avons donc retenu un `batch_size` de 1, malgré les temps plus importants.

La topologie du réseau de neurone est la suivante :

- Une première couche cachée de 30 neurones LSTM, avec une fonction d'activation ReLU
- Une seconde couche cachée de 10 neurones LSTM, avec une fonction d'activation ReLU
- Une couche dense de sortie de 1 neurone, sans fonction d'activation

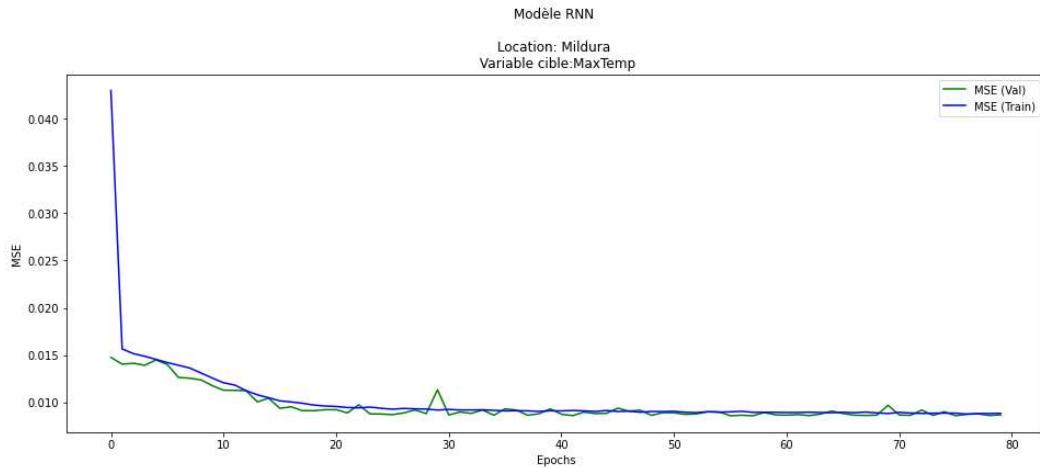
La fonction de perte est une MSE.

Pour savoir combien de jours dans le passé il convient de reprendre, regardons les performances obtenues en faisant varier cette valeur. Il semble qu'une fenêtre de 15 jours soit optimale.

**Impact du nombre de journées dans le passé**

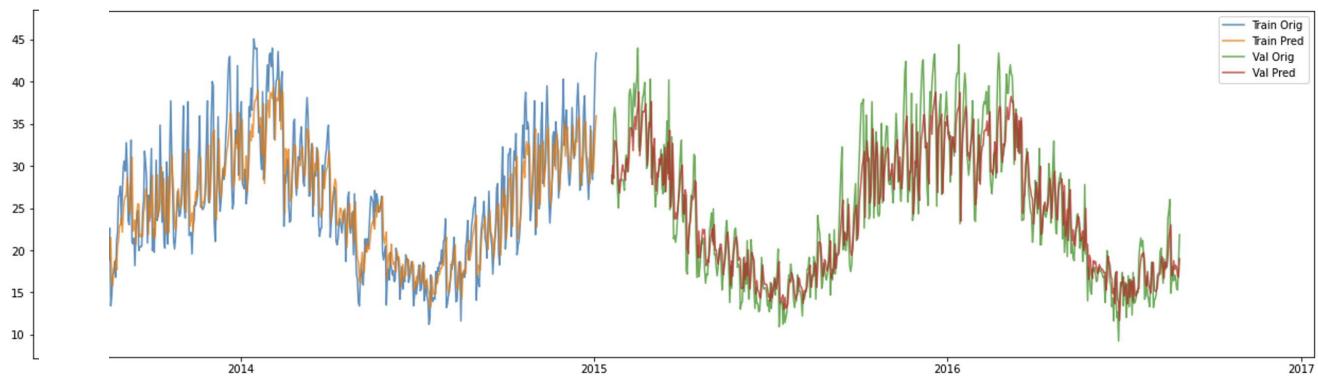
	RMSE (train)	RMSE (valid)	MAE (train)	MAE (valid)
30 jours	3,568	3,556	2,732	2,723
15 jours	3,507	3,500	2,679	2,669
7 jours	3,576	3,572	2,726	2,743
3 jours	3,841	3,894	2,948	3,019

L'évolution de la fonction de perte se déroule correctement au fil des 80 époques d'entraînement :



**Figure 46 : Evolution de la loss (MSE) lors de l'apprentissage**

Les prédictions des températures sont assez proches des données réelles, bien sûr sur les dates d'entraînement (bleu) mais aussi de validation (vert). En revanche, on constate que le modèle reste chaque fois dans une fourchette plus réduite que la réalité. Il n'arrive jamais à prédire des pics de chaleurs ou des vagues de froid.

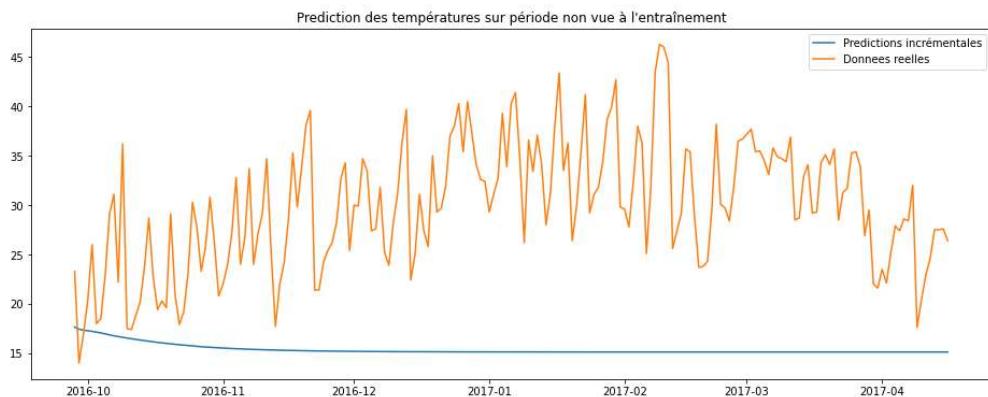


**Figure 47 : Comparaison des prédictions (sur train et validation) avec les données réelles**

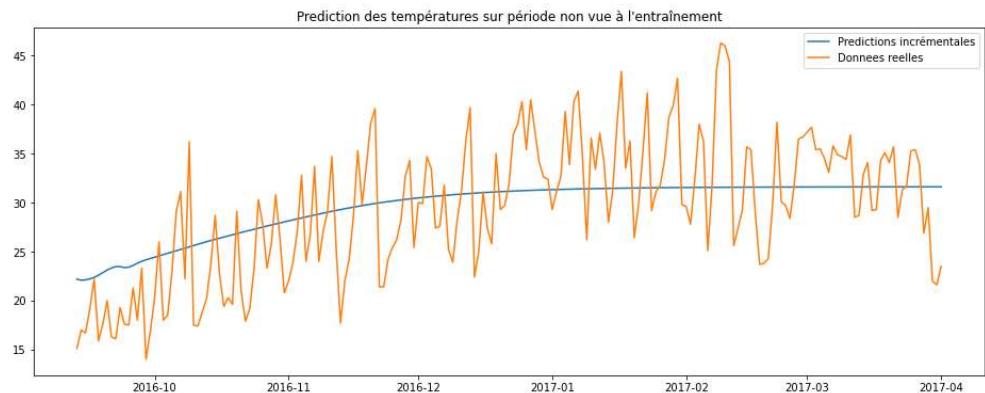
#### 4.4.1.2 Prédiction monovariée sur plusieurs jours

A partir du meilleur modèle entraîné, essayons maintenant de prédire de façon itérative la température maximale sur plusieurs jours de suite. Notre modèle a été entraîné sur des données de train et de validation. Nous allons ici effectuer les prédictions sur des données jamais vues par le modèle.

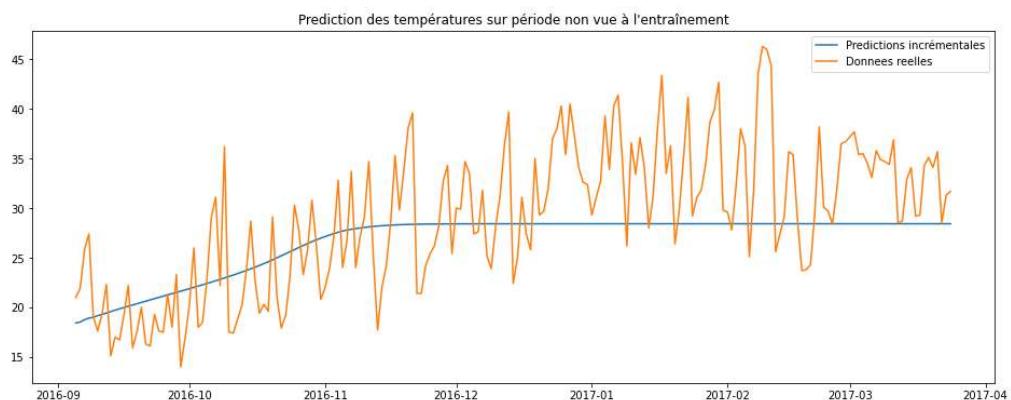
Nous avons vu plus haut qu'une fenêtre de 15 jours était optimale au regard des métriques observées. Comparons maintenant visuellement les conséquences sur le forecast des températures pour ces 4 plages de fenêtres :



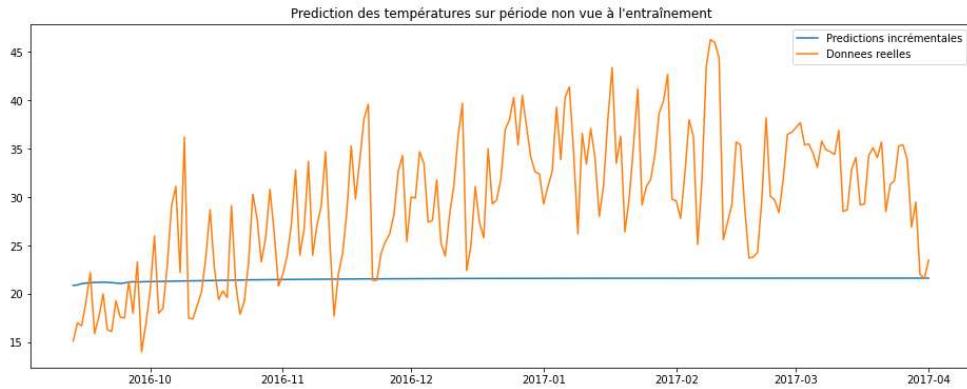
**Figure 48 : Fenêtre de 30 jours**



**Figure 49 : Fenêtre de 15 jours**



**Figure 50 : Fenêtre de 7 jours**



**Figure 51 : Fenêtre de 3 jours**

Le modèle avec une fenêtre de 30 jours ne part pas dans la bonne direction pour prédire les températures. A l'instar du modèle entraîné avec une fenêtre de 3 jours, il converge rapidement vers une valeur fixe dont il ne s'écarte plus. Les deux autres modèles finissent également par converger, mais sur une durée plus longue : environ 2 mois pour le modèle avec fenêtre de 7 jours et 3 mois pour le modèle d'une fenêtre de 15 jours.

Les prédictions présentent globalement assez peu d'intérêt : si la tendance générale est respectée par le modèle avec fenêtre de 15 jours, les variations soudaines d'une journée à l'autre ne sont en rien prédictes par le modèle. L'utilité d'un forecast des températures basé sur un modèle univarié semble donc avoir un intérêt limité.

## 5 Autres variables cibles

Il aurait été intéressant de pouvoir prédire *Rainfall*, laquelle nous aurait permis dans un second temps de déduire *RainTomorrow*. Cependant, la dispersion et l'irrégularité de cette variable font qu'il semble particulièrement complexe de prédire cette donnée. Par ailleurs, souvenons-nous que le site du Bureau of Meteorology indiquait que le niveau des pluviomètres n'était parfois pas relevé pendant quelques jours, et que le cumul des précipitations de ces journées était de ce fait rattaché à la prochaine journée de relève. Nous cumulons donc ici une problématique de distribution et un problème de récolte de données.

## 6 Conclusion

### 6.1 Constats

Nos modélisations nous ont réservé plusieurs surprises lors de nos explorations. La première d'entre elle a été la robustesse et l'intérêt du XGBoost : pour notre problématique, ce modèle, très rapide à entraîner, rivalise y compris avec des réseaux de neurones bien plus complexes.

La seconde surprise a été que malgré le fort déséquilibre de nos classes sur la variable cible, la plupart des modèles entraînés nous ont offert de très belles performances.

La troisième surprise est l'apport limité du feature engineering dans notre problématique. Même s'il présente bel et bien un intérêt, en particulier pour effectuer des prédictions plus éloignées dans le futur, nous avons été particulièrement surpris par le faible écart de performances une modélisation effectuée avec les données initiales et celles obtenues après plusieurs mois de feature engineering. C'est une vraie leçon en termes de gestion de projet sur le temps à budgéter sur cet aspect. Il est en effet assez aisé de se perdre dans d'innombrables conjectures pour un intérêt potentiellement infime.

La quatrième surprise a été la possibilité de prédire la pluie sur une année à partir de l'observation d'une seule journée ! Encore une fois, les performances sont faibles, mais cette possibilité est particulièrement intrigante. Il s'agit là d'un aspect qu'il serait intéressant d'approfondir avec le regard d'un météorologue australien, qui pourrait potentiellement comprendre ce phénomène à partir de son expertise métier.

## 6.2 Limites et perspectives

Pour autant, nous restons avec une satisfaction mitigée sur les performances finales de nos notre meilleur modèle. Nous espérions en effet obtenir des prédictions quasiment fiables à 100%. Or, nous en sommes très loin avec notre accuracy de 86,6%, en particulier au regard du taux de journées non pluvieuses de 77,6%.

Nous avons toutefois pu identifier au fil du projet plusieurs pistes qui nous permettraient d'améliorer potentiellement les performances. Tout d'abord, nous avons vu dès la phase exploratoire que la feature Sunshine était absente sur environ la moitié des observations alors qu'elle présentait une forte corrélation avec la variable cible. L'importance de Sunshine a d'ailleurs été confirmée dans les analyses d'interprétabilité des différents modèles. Par conséquent, la première action à mener pourrait être de récolter les valeurs de Sunshine pour le maximum d'observations.

Deux autres variables sont très importantes : *Humidity3pm* et *Pressure3pm*. Une autre piste pourrait être de récolter le taux d'humidité et la pression atmosphérique à d'autres heures de la journée. Celles, existantes, de 9h du matin sont bien moins importantes, mais peut-être que les modèles gagneraient à ajouter ces taux à 14h ou 16h, par exemple. D'ailleurs, n'oublions pas que l'Australie s'étale sur plusieurs fuseaux horaires, allant de GMT+8 à GMT+11 : même si *Pressure3pm* est relevée à 15h pour chaque Location sur tout le pays, il ne s'agit en réalité pas de la même heure par rapport au soleil.

*Rainfall* est une variable dont la qualité de renseignement est discutable, alors même qu'elle est la source de notre variable cible *RainTomorrow* : contrairement aux autres variables, qui peuvent simplement être inexistantes pour une journée, Rainfall s'accumule en réalité dans le pluviomètre pendant plusieurs jours jusqu'à ce que le niveau soit relevé par un des bénévoles en charge. Il est probable que les modèles gagneraient en précision si les relevés étaient réellement quotidiens et, *a minima*, que les valeurs ne se cumulent pas en cas d'absence de relevé.

Au-delà des features elles-mêmes, il conviendrait aussi de disposer d'un dataset avec le plus de dates possibles renseignées. Pour rappel, il nous manque pour l'intégralité des lieux trois mois complets, et, pour certains lieux tels Melbourne, il manque plus d'un an et demi en cumulé sur la plage de dates. Ces trous sont de nature à perturber les modélisations par série temporelle. Il serait donc bénéfique de disposer de dates intégralement observées.

Notre dataset porte sur dix années, ce qui peut sembler conséquent, mais qui est en réalité assez peu à l'échelle des possibilités du machine learning. Il serait intéressant de pouvoir disposer de relevés sur une période de plusieurs décennies.

Etant donnée l'immensité de l'Australie, il pourrait être profitable de disposer de relevés d'autres stations météorologiques afin d'avoir d'une part plus de données, mais également un meilleur maillage géographique, qui permettrait peut-être de toutes nouvelles approches de prédictions basées sur les villes voisines.

Avant le rapport final, nous allons explorer les RNN multivariés, dont nous espérons beaucoup. En fonction du temps disponibles, nous expérimenterons peut-être des séries temporelles avec d'autres bibliothèques, voire une modélisation par Transformer.