Rapport de modélisation sur les données météorologiques Australiennes

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

Encadrant : Francesco MADRISOTTI

Réalisateurs : Sophie BERTHIER

Luciano LANGHI

Quyen THIEU MARCAUD

Le ../../2023 – work in progress

Table des matières

[1 Introduction 3](#_Toc152009242)

[1.1 Méthodologie 3](#_Toc152009243)

[1.2 Approches 3](#_Toc152009244)

[2 RainTomorrow 4](#_Toc152009245)

[2.1 Rappel sur déséquilibre 4](#_Toc152009246)

[2.2 Métriques 4](#_Toc152009247)

[2.3 Résultats de la classification par approches « classiques » via scikit-learn 5](#_Toc152009248)

[2.3.1 Modèles étudiés 5](#_Toc152009249)

[2.3.2 Optimisation de métriques 5](#_Toc152009250)

[2.3.3 Impact du feature enginerring 12](#_Toc152009251)

[2.3.4 Seuil de probabilité 13](#_Toc152009252)

[2.3.5 Interprétabilité des modèles 16](#_Toc152009253)

[2.3.6 Conclusions 23](#_Toc152009254)

[2.4 Deep Learning avec Keras et TensorFlow 27](#_Toc152009255)

[2.4.1 DNN 27](#_Toc152009256)

[2.4.2 RNN 27](#_Toc152009257)

[2.5 Exploitabilité et limites 27](#_Toc152009258)

[3 Extension à Rain+J 27](#_Toc152009259)

[4 MaxTemp 27](#_Toc152009260)

[4.1 Métriques 27](#_Toc152009261)

[4.2 Résultats de la régression par approches « classiques » via scikit-learn 27](#_Toc152009262)

[4.3 Séries Temporelles 27](#_Toc152009263)

[4.4 Deep Learning 28](#_Toc152009264)

[4.4.1 DNN 28](#_Toc152009265)

[4.4.2 RNN 28](#_Toc152009266)

[4.5 Interprétabilité des modèles 28](#_Toc152009267)

[4.6 Exploitabilité et limites 28](#_Toc152009268)

[5 Extension à MaxTemp+J (pas sûre qu’on ait le temps de le développer et que ça soit utile) 28](#_Toc152009269)

[6 Autres variables cibles 28](#_Toc152009270)

[7 Conclusion 28](#_Toc152009271)

# Introduction

## 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.

## 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 en 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 génèrerons 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.

Dans un souci de synthèse et de lisibilité, nous ne déclinerons pas systématiquement par la suite chaque modélisation selon chacune des approches évoquée 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.

# RainTomorrow

## 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, les résultats obtenus n’ont pas indiqué de différence significative sur les performances obtenues. Nous avons donc finalement conservé les données sans 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.

## 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 globale 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 déprendrait 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.

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

### 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)

### 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 apporté dans tous les modèles sur ce point, d’une part en nous assurant que l’accuracy du trainset était proche du testset (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'entrainement et l'ensemble de test, nous n’avons pas identifié de problème d'overfitting.

#### Maximisation de l’accuracy

Les résultats présentés dans le tableau ci-dessous montrent les performances de quatre modèles différents en termes de cinq métriques d'évaluation, avec l'optimisation basée sur l’accuracy.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Scores des modèles ayant meilleur accuracy** | | | | | |
|  | **accuracy** | **recall** | **precision** | **f1** | **auc** |
| LogisticRegression | 0.8461 | 0.5004 | 0.7278 | 0.5930 | 0.8694 |
| TreeDecision | 0.8389 | 0.4841 | 0.7049 | 0.5740 | 0.8456 |
| RandomForest | 0.8465 | 0.4761 | 0.7477 | 0.5818 | 0.8653 |
| **XGBoost** | **0.8526** | 0.5407 | 0.7315 | 0.6218 | 0.8844 |

* 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 le graphique ci-dessous 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.

Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement

#### Maximisation de la précision

Les résultats présentés dans le tableau ci-dessous 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 |
| **RandomForest** | 0.7830 | 0.0367 | **0.8897** | 0.0705 | 0.8315 |
| XGBoost | 0.8410 | 0.4144 | 0.7698 | 0.5388 | 0.8567 |

* Précision : La precision mesure la proportion d'exemples positifs parmi ceux que le modèle a identifiés comme positifs. Dans ce cas, Random Forest a la precision 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.

Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement

#### Maximisation du recall

Les résultats présentés dans le tableau ci-dessous 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** |
| **LogisticRegression** | 0.7366 | **0.8464** | 0.4531 | 0.5902 | 0.8640 |
| TreeDecision | 0.8313 | 0.4944 | 0.6669 | 0.5679 | 0.8180 |
| RandomForest | 0.8397 | 0.5062 | 0.6956 | 0.5860 | 0.8469 |
| XGBoost | 0.8292 | 0.5547 | 0.6365 | 0.5928 | 0.8481 |

* 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).

Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement

#### Maximisation du F1-score

Les résultats présentés dans le tableau ci-dessous 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 | 0.8460 | 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 |
| **XGBoost** | 0.8554 | 0.5475 | 0.7397 | **0.6292** | 0.8852 |

* 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.

Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement

#### Maximisation de l’AUC

Les résultats présentés dans le tableau ci-dessous 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 | 0.8460 | 0.5004 | 0.7273 | 0.5929 | 0.8695 |
| TreeDecision | 0.8399 | 0.4478 | 0.7341 | 0.5563 | 0.8493 |
| RandomForest | 0.8497 | 0.4980 | 0.7474 | 0.5978 | 0.8716 |
| **XGBoost** | 0.8538 | 0.5358 | 0.7400 | 0.6216 | **0.8844** |

* 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

Une image contenant texte, capture d’écran, ligne, diagramme

Description générée automatiquement

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

Les résultats précédents 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 ci-après 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 | 0.8538 | 0.5358 | 0.7400 | 0.6216 | 0.8844 |
| Macro (sophie) | 0.8658 | 0.5585 | 0.7635 | 0.6451 | 0.8976 |
| Micro (moyenne) | 0.8437 | 0.3287 | 0.8378 | 0.4626 | 0.8555 |
| Micro (meilleur : Uluru) | 0.9635 | 0.3125 | 1.000 | 0.4762 | 0.9691 |
| Climat (moyenne) | 0.8627 | 0.5552 | 0.7698 | 0.6419 | 0.8955 |
| Climat (meilleur : Centre) | 0.9337 | 0.4444 | 0.7733 | 0.5645 | 0.9275 |

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, Uluru, est celui qui présente les meilleures performances de tous les modèles confondus.

Afin que les données soient comparables, regardons les performances des trois niveaux de modélisation appliqués sur trois *Location* :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Comparaison des modèles pour Uluru** | | | | | |
|  | **accuracy** | **recall** | **precision** | **f1** | **auc** |
| Micro | 0.9635 | 0.3125 | 1.000 | 0.4762 | 0.9691 |
| Global | 0.9401 | 0.4000 | 0.8333 | 0.5405 | 0.8985 |
| Climat – 2, Centre | 0.9490 | 0.4091 | 0.8182 | 0.5454 | 0.9495 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Comparaison des modèles pour SalmonGums** | | | | | |
|  | **accuracy** | **recall** | **precision** | **f1** | **auc** |
| Micro | 0.8401 | 0.0980 | 0.8333 | 0.1754 | 0.7735 |
| Global | 0.8831 | 0.3933 | 0.6863 | 0.5000 | 0.8397 |
| Climat – 4, Intermédiaire | 0.8728 | 0.4432 | 0.6190 | 0.5166 | 0.8463 |

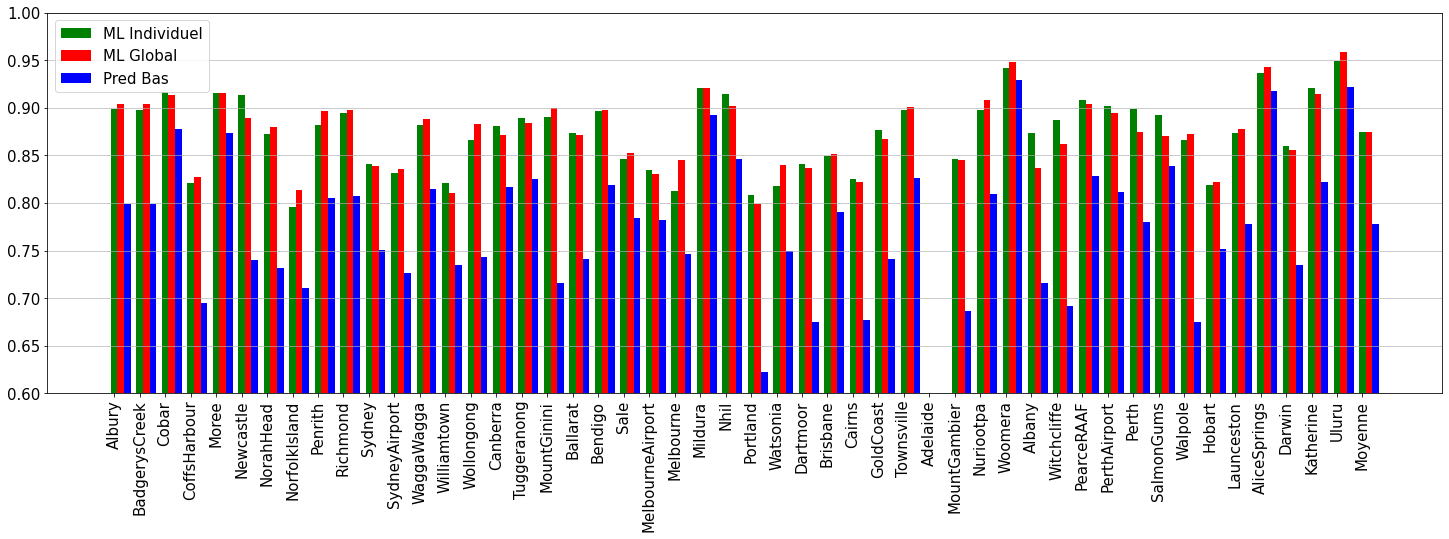
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Comparaison des modèles pour Melbourne** | | | | | |
|  | **accuracy** | **recall** | **precision** | **f1** | **auc** |
| Micro | 0.8087 | 0.3393 | 0.7308 | 0.4634 | 0.8447 |
| Global | 0.8151 | 0.3153 | 0.8750 | 0.4636 | 0.8363 |
| Climat – 3, Sud-Est | 0.8365 | 0.4000 | 0.7018 | 0.5096 | 0.8692 |

Les tableaux ci-dessus illustrent qu’aucune des trois approches, globale, locale ou climatique, n’est meilleure de façon systématique. Pour Uluru, c’est un modèle entraîné spécifiquement sur les données du célèbre rocher qui offre les meilleures performances. Pour SalmonGums, le modèle global performe mieux que les deux autres. Enfin, on préfèrera une modélisation par climat pour Melbourne.

Ces trois niveaux de granularité semblent donc complémentaires.

De façon plus détaillée, observons l’accuracy pour chacune 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 0,875 pour les ML Individuels, 0,8746 pour les ML Global et 0,7782 pour le modèle « Pred Bas »). L’approche micro 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.

### 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 NA sont traités par suppression pure et simple, sans substitution. Les quatre variables ayant un taux très important de NA sont supprimées, faute de quoi un dropna global entraine la suppression de plus de la moitié du dataset.

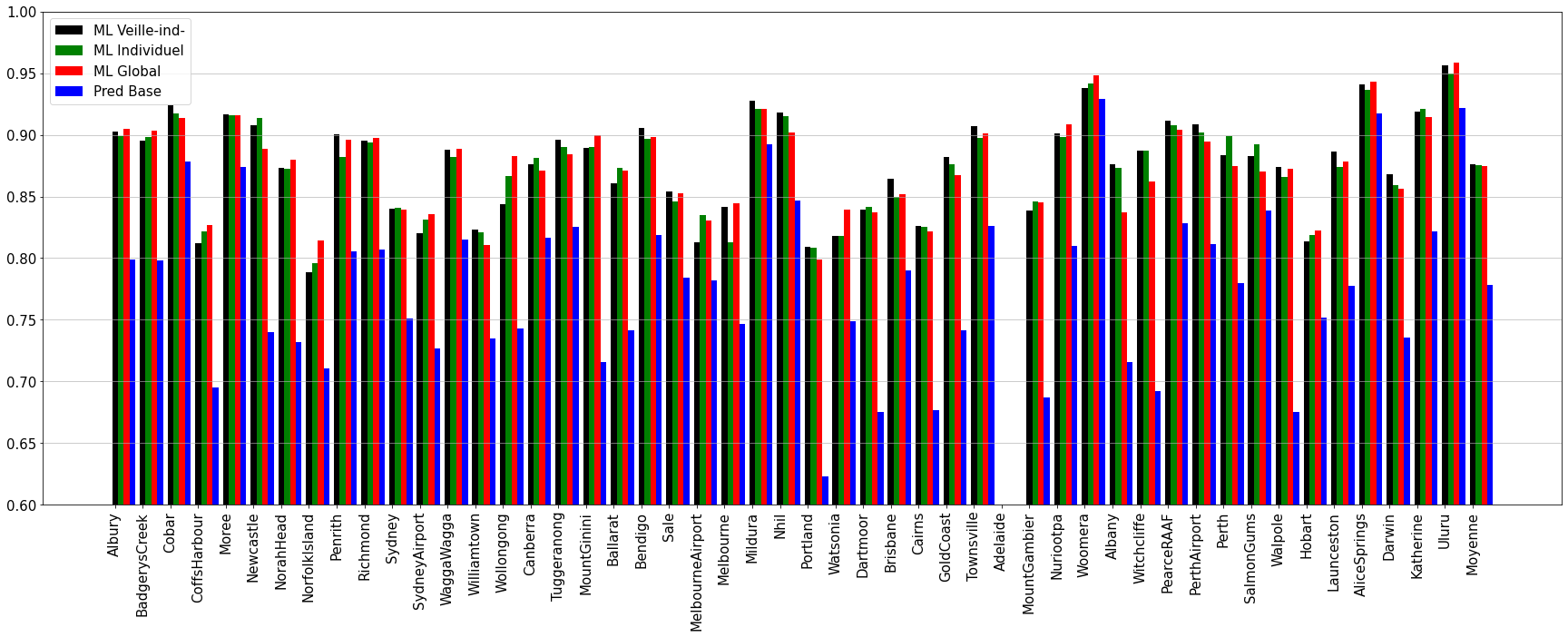
Le modèle dit « avec nouvelles features » est issu des travaux menés dans la première partie du projet. Les valeurs nulles sont gérées par KNN Imputation, les variables catégorielles de direction du vent sont remplacées par des variables quantitatives trigonométriques, les *Location* sont remplacées 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\*(numéro du jour dans l’année/365) et cosinus 4\*pi\*(numéro du jour dans l’année/365).

Nous comparons ci-dessous les performances d’un XGBoost avec les mêmes hyperparamètres 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 | 0.8658 | 0.5585 | 0.7635 | 0.6451 | 0.8976 |
| Avec features d’origine | 0.8571 | 0.5320 | 0.7406 | 0.6192 | 0.8879 |

Certes, le modèle bénéficiant des nouvelles features propose de meilleures performances quelle que soit la métrique observée, mais le gain est très faible.

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.



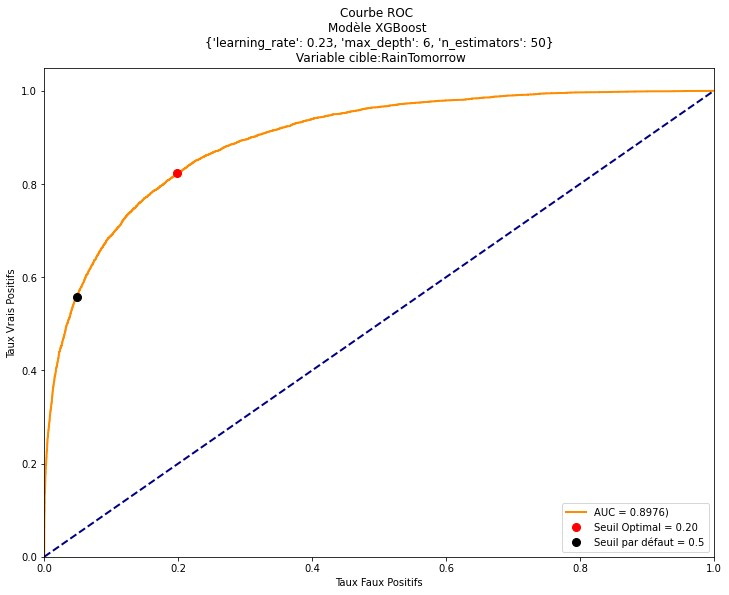
On constate qu'avec les données de la veille, le score du modèle s'améliore légèrement par rapport à un modèle individuel (0.8760 contre 0.8750). Là aussi, nous obtenons donc de meilleures performances avec cet ajout, mais le gain est encore plus faible que précédemment.

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

### 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 prédiction pour l’affiner.



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 capable 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 précédente. 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 ? Préfèrere-t-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.

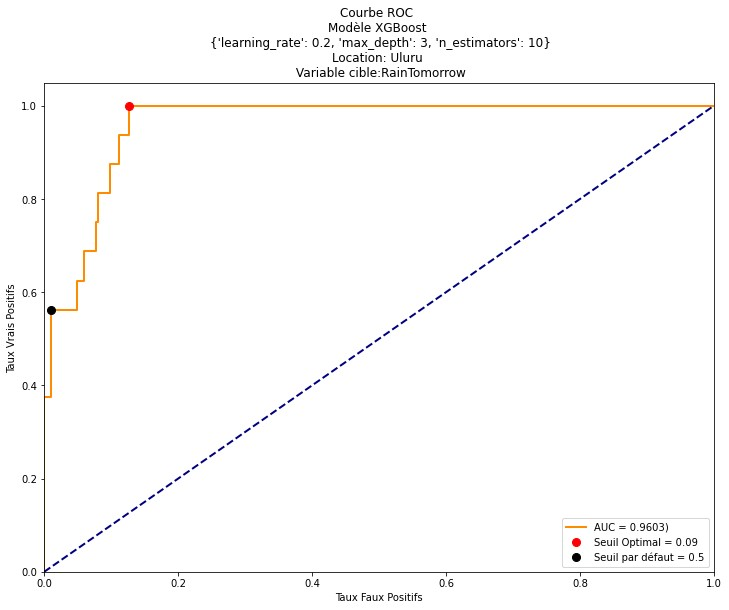
Ce changement réduit logiquement légèrement l’accuracy, mais nous offre un gain souvent significatif sur le recall. Ici, nous voyons que les jours de pluie sont d’environ 0,8 et les faux positifs de 0,2, c’est-à-dire que lorsqu’il pleut, nous pouvons le prévoir environ 4 fois sur 5, de même que lorsqu’il 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 et les matrices de confusions. Notons d’ailleurs qu’au-delà de la légère baisse d’accuracy, nous avons une baisse importance 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) | 0.8658 | 0.5585 | 0.7635 | 0.6451 | 0.8976 |
| Seuil Optimal (0,20 | 0.8060 | 0.8235 | 0.5363 | 0.6496 | 0.8976 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Classe prédite | |  |  |  | Classe prédite | |
|  |  | False | True |  |  |  | False | True |
| Classe réelle | False | 20 503 | 1 042 |  | Classe réelle | False | 17 259 | 4 286 |
| True | 2 658 | 3 363 |  | True | 1 063 | 4 958 |
|  |  |  |  |  |  |  |  |  |
|  |  | Seuil par défaut (0,5) | | |  |  | Seuil Optimal (0,2) | |

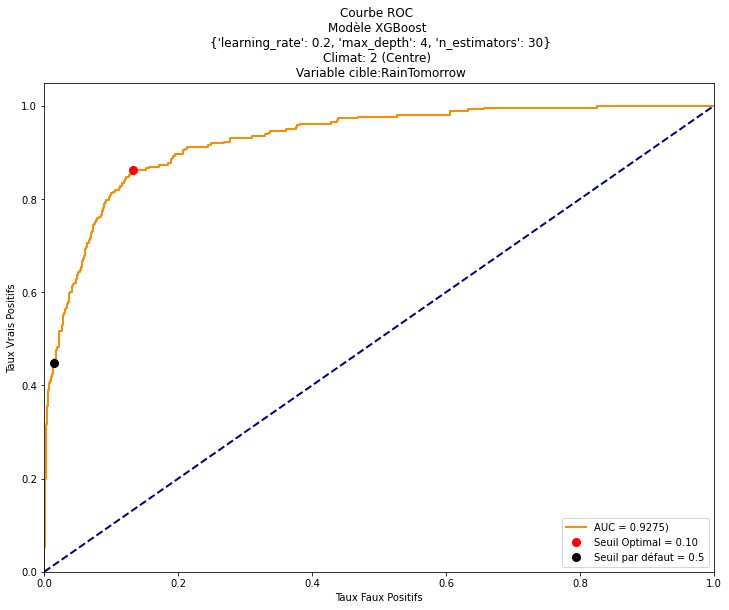
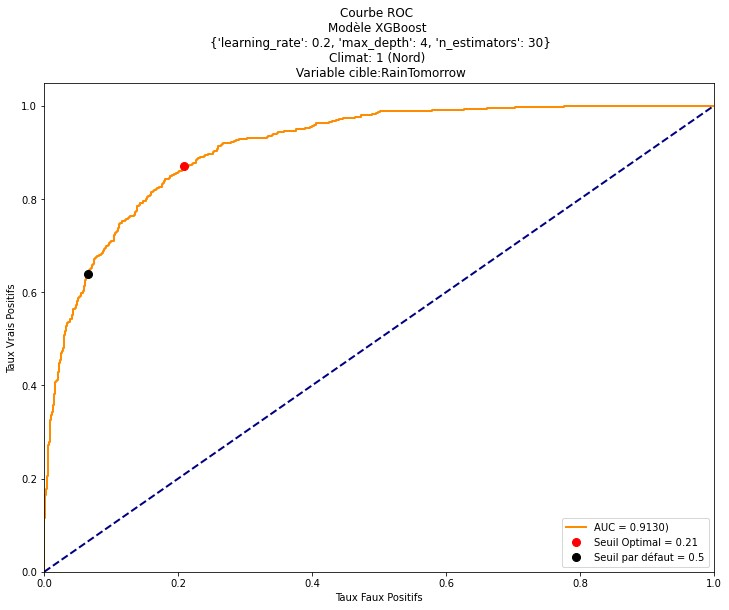
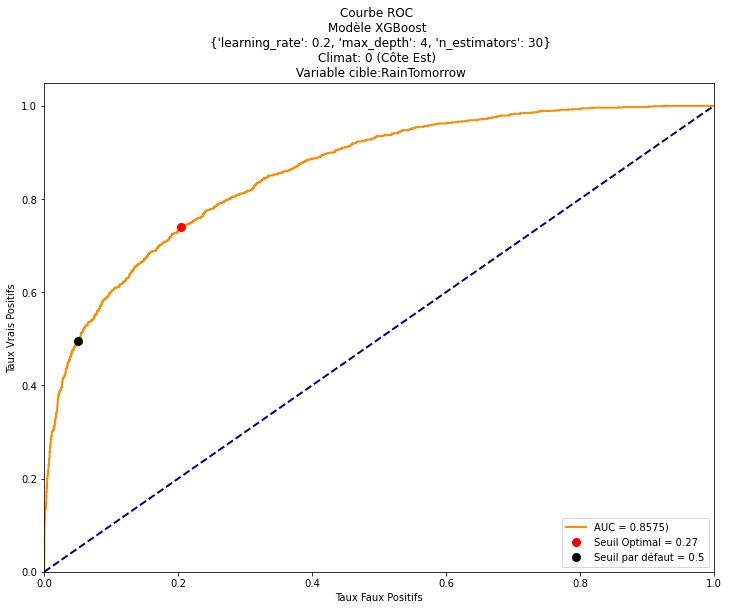
Regardons le cas tout à fait extrême de la station d’Uluru, située en plein désert (zone Centre) :

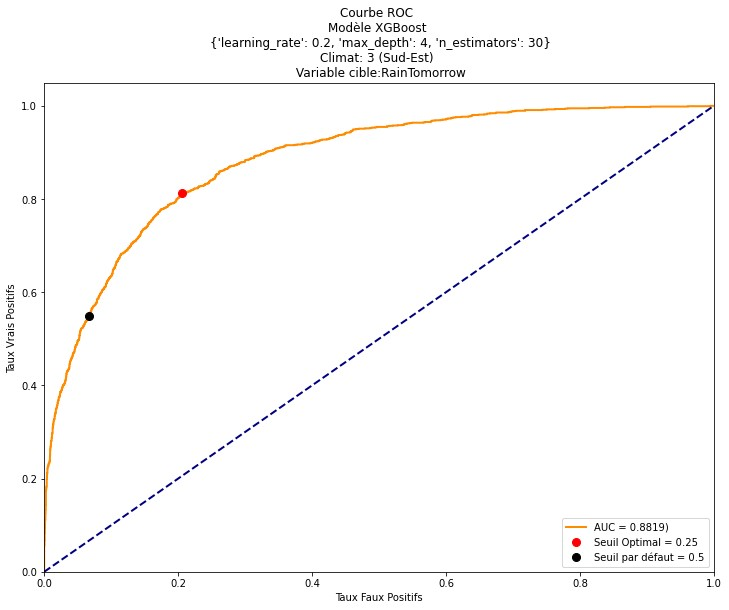
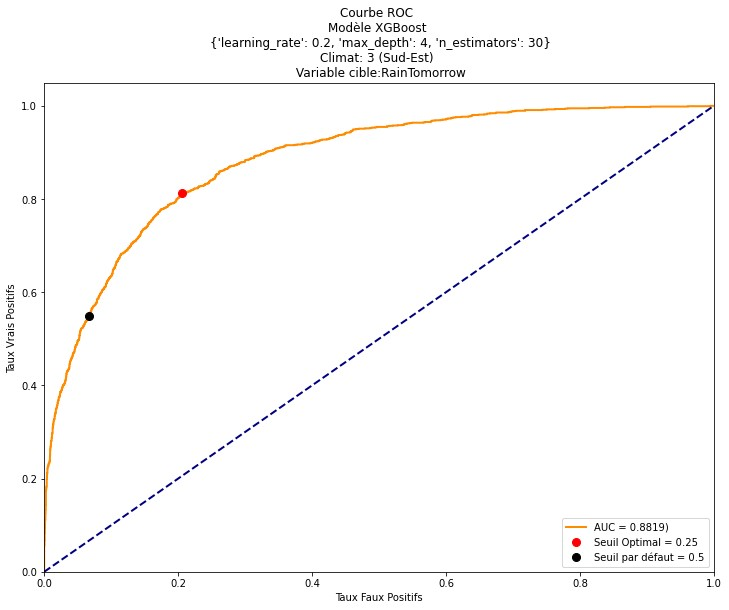
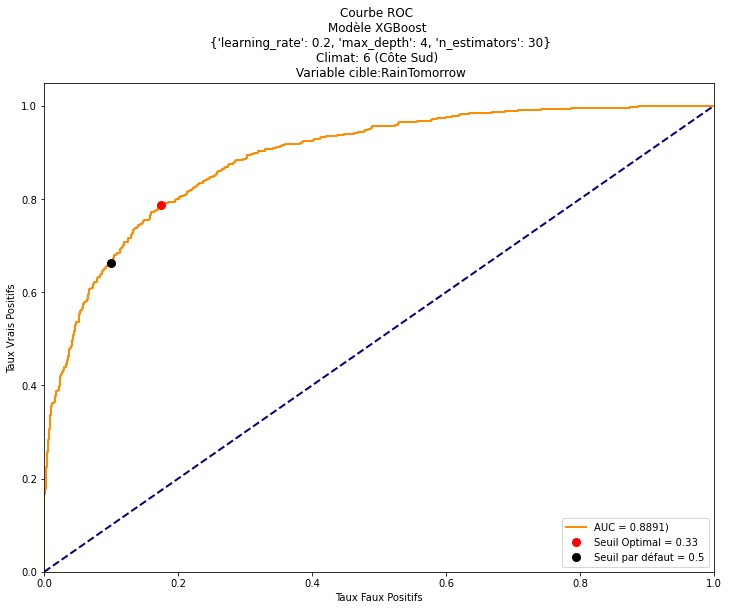


|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Classe prédite | |  |  |  | Classe prédite | |
|  |  | False | True |  |  |  | False | True |
| Classe réelle | False | 282 | 3 |  | Classe réelle | False | 249 | 36 |
| True | 7 | 9 |  | True | 0 | 16 |
|  |  |  |  |  |  |  |  |  |
|  |  | Seuil par défaut (0,5) | |  |  |  | Seuil optimal (0,09) | |

En passant du seuil par défaut au seuil optimal (0,09 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 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.

Ci-après 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 :



### Interprétabilité des modèles

#### Importance des features

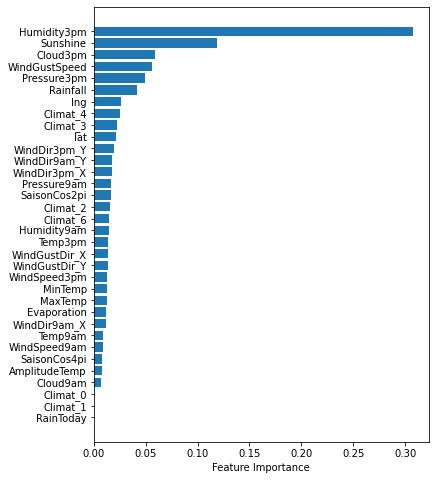
Sur la quasi-totalité des modèles entraînés, la variable la plus importante semble être de loin *Humidity3pm*, indiquant le taux d’humidité à 15h00. Contre toute attente, *RainToday* semble finalement peu importante dans la prédiction de chute de pluie le lendemain.

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 point nettement inférieur dans les features importances.

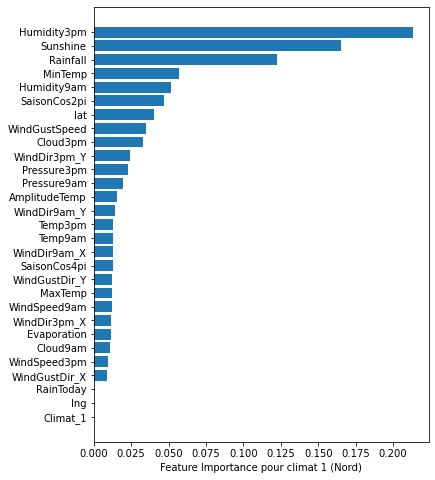
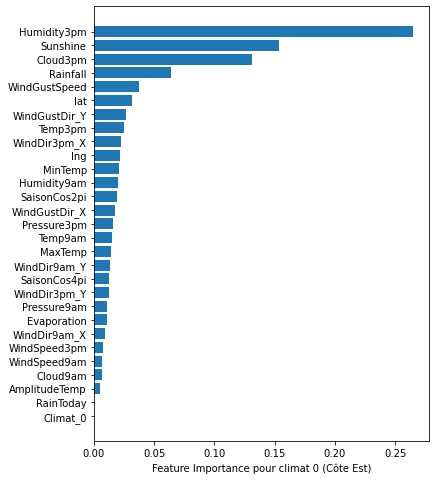
Une image contenant texte, capture d’écran, Parallèle, ligne

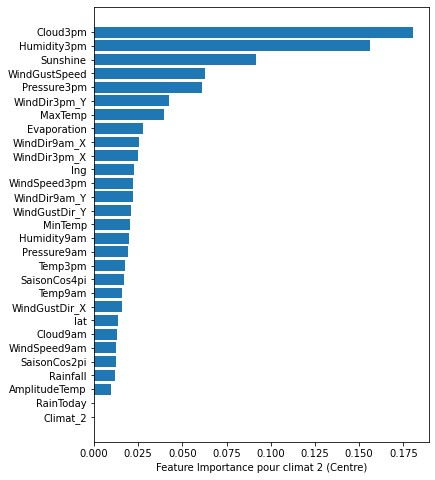
Description générée automatiquement

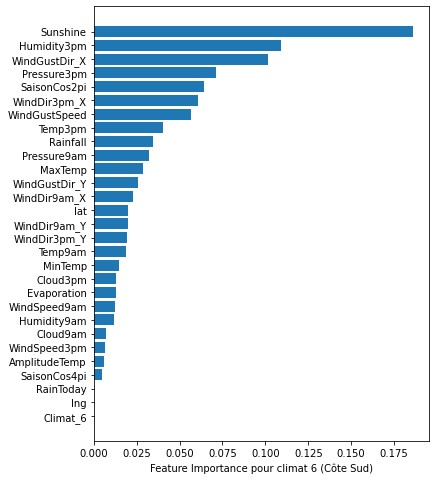
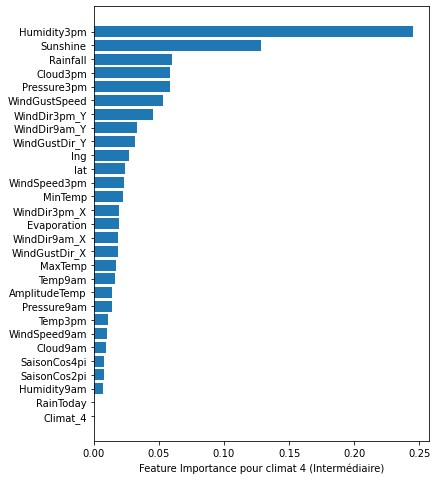
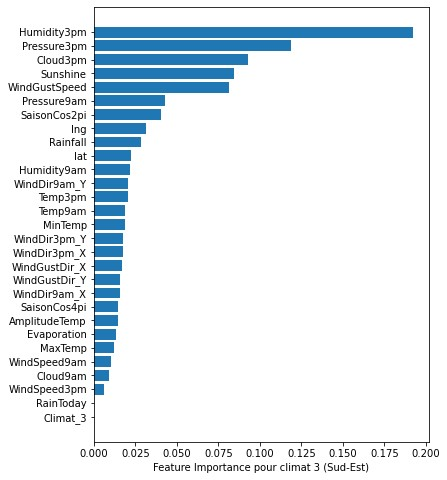


Sur la modèle global, les variables ajoutées de la longitude, la latitude et l’appartenance aux zones climatiques Intermédiaire et Sud-Est sont identifiées par le modèle parmi le quart des features les plus significatives.

Sur 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).







Features importances pour chaque zone climatique

#### 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 le graphique ci-dessous 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.

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

#### 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 le beeswarm plot ci-dessous, nous pouvons commencer à comprendre la nature des relations entre les variables et la pluie du lendement. 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.

Une image contenant texte, capture d’écran

Description générée automatiquement

#### 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, 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.

Une image contenant texte, diagramme, Police, Tracé

Description générée automatiquementUne image contenant texte, diagramme, Tracé, ligne

Description générée automatiquement

Une image contenant texte, diagramme, ligne, Tracé

Description générée automatiquementUne image contenant texte, Tracé, ligne, diagramme

Description générée automatiquement

### Conclusions

#### Waterfall Plot

Ce graphique nous aide à visualiser les valeurs SHAP de chaque échantillon dans nos données individuellement. Visualisons 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.

Une image contenant texte, capture d’écran, nombre, diagramme

Description générée automatiquement

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

Une image contenant texte, capture d’écran, diagramme, nombre

Description générée automatiquement

## Deep Learning avec Keras et TensorFlow

### DNN

Modèle optimal (2 couches, pas trop de neurones, tanh, sigmoid en sortie)

Pas meilleur que XGBoost, mais beaucoup plus long => pas intéressant

### RNN

#### Prédiction monovariée sur une journée

Moins bon que XGBoost

#### Prédiction monovariée sur plusieurs jours

Moins bon que XGBoost. Très mauvais. Très long. A rapprocher de Rain\_J du chapitre 3

#### Prédiction multivariée

Pas encore testé

## Exploitabilité et limites

* Précision restant faible

# Extension à Rain+J

* Tentative de prédire non seulement RainTomorrow (=Rain+1) mais également Rain+J, avec un modèle XGBoost par J
* Résultats sur une année, en global, par zone climatique, par location
* Comparaison avec RNN multivarié sur plusieurs jours
* Conclusions

# MaxTemp

## Métriques

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

Prédiction pour le lendemain uniquement

## Séries Temporelles

Reprendre partie écrite dans 1er rapport. Constat de la saisonnalité. Résidu restant important.

Détermination des valeurs max (p,d,q)(P,D,Q)I

Impossibilité de modéliser via SARIMAX sur une saisonnalité de 365 jours (trop de temps de calcul)

## Deep Learning

### DNN

bof

### RNN

Pas encore testé

## Interprétabilité des modèles

## Exploitabilité et limites

# Extension à MaxTemp+J (pas sûre qu’on ait le temps de le développer et que ça soit utile)

* Tentative de prédire non seulement MaxTemps pour le lendemain, mais également pour plusieurs jours, avec un modèle XGBoost par J
* Résultats sur une année, en global, par zone climatique, par location
* Comparaison avec RNN
* Conclusions

# Autres variables cibles

Rainfall => résultats très mauvais. Pas possible de prédire cette variable.

# Conclusion

Difficultés rencontrées, échecs

Aspects positifs

Pistes de travaux d’amélioration (sunshine à récupérer sur site du BOM si on avait + de temps car souvent utilisé comme critère important par SHAP)