## 1.2. Résultats

Table 1 présente nos modèles et les scores tirés de leurs rapports de classification.

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

**Table 1.** Synthèse des rapports de classification triée par ordre décroissant selon notre métrique de choix, precision\_1 (prec\_1 dans la table : les noms des métriques dans les en-têtes ont été raccourcis pour faciliter la mise en page).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Rang** | **Algorithme** | **Variation** | **acc** | **prec\_0** | **prec\_1** | **rec\_0** | **rec\_1** | **f1\_0** | **f1\_1** |
| 1 | Voting soft  optimisé | StandardScaler + SMOTE + ROC | 0,82 | 0,82 | 0,94 | 1,00 | 0,22 | 0,90 | 0,36 |
| 2 | ROCKET | sort + TSS + tslearn | 0,80 | 0,80 | 0,92 | 1,00 | 0,15 | 0,89 | 0,26 |
| 3 | Stacking  optimisé | StandardScaler + SMOTE + ROC | 0,84 | 0,83 | 0,91 | 0,99 | 0,29 | 0,90 | 0,44 |
| 4 | ROCKET | sort + TSS + MinMaxScaler + tslearn | 0,80 | 0,80 | 0,89 | 0,99 | 0,15 | 0,89 | 0,26 |
| 5 | XGBoost | StandardScaler + SMOTE + ROC | 0,85 | 0,84 | 0,88 | 0,99 | 0,35 | 0,91 | 0,50 |
| 6 | Voting hard  optimisé | StandardScaler + SMOTE | 0,86 | 0,86 | 0,81 | 0,97 | 0,45 | 0,91 | 0,58 |
| 7 | RF | StandardScaler + SMOTE + ROC | 0,86 | 0,87 | 0,80 | 0,97 | 0,50 | 0,92 | 0,61 |
| 8 | TSF | sort + TSS | 0,82 | 0,83 | 0,79 | 0,98 | 0,30 | 0,89 | 0,43 |
| 9 | TSF | sort + TSS | 0,82 | 0,82 | 0,79 | 0,98 | 0,30 | 0,89 | 0,43 |
| 10 | TSF | sort + TSS + StandardScaler | 0,82 | 0,83 | 0,77 | 0,97 | 0,31 | 0,89 | 0,44 |
| 11 | TSF | sort + TSS + SMOTE | 0,83 | 0,84 | 0,75 | 0,96 | 0,37 | 0,90 | 0,49 |
| 12 | DNN | StandardScaler | 0,85 | 0,88 | 0,73 | 0,94 | 0,54 | 0,91 | 0,62 |
| 13 | RF | StandardScaler + SMOTE | 0,83 | 0,89 | 0,67 | 0,88 | 0,70 | 0,88 | 0,68 |
| 14 | TSF | sort + TSS + StandardScaler + SMOTE | 0,83 | 0,87 | 0,64 | 0,91 | 0,54 | 0,89 | 0,59 |
| 15 | LogReg | StandardScaler + SMOTE | 0,82 | 0,91 | 0,63 | 0,84 | 0,77 | 0,87 | 0,70 |
| 16 | KNN-DTW | sort + TSS + StandardScaler | 0,78 | 0,85 | 0,51 | 0,86 | 0,49 | 0,86 | 0,50 |
| 17 | KNN-DTW | sort + TSS + MinMaxScaler | 0,78 | 0,85 | 0,51 | 0,87 | 0,47 | 0,86 | 0,49 |
| 18 | KNN-DTW | sort + TSS + StandardScaler + RandomUnderSampler | 0,70 | 0,89 | 0,41 | 0,70 | 0,71 | 0,79 | 0,52 |
| 19 | ROCKET | sort + TSS + StandardScaler + RandomUnderSampler + tslearn | 0,25 | 0,99 | 0,23 | 0,03 | 1,00 | 0,05 | 0,38 |

La stricte application du seuil de 83% à precision\_1 élimine tous les algorithmes sauf les 5 premiers :

1. Voting soft optimisé avec StandardScaler[[1]](#endnote-1) + SMOTE[[2]](#endnote-2) + ROC[[3]](#endnote-3)
2. ROCKET avec sort[[4]](#endnote-4) + TSS[[5]](#endnote-5) + tslearn[[6]](#endnote-6)
3. Stacking optimisé avec StandardScaler + SMOTE + ROC
4. ROCKET avec sort + TSS + MinMaxScaler[[7]](#endnote-7) + tslearn
5. XGBoost avec StandardScaler + SMOTE + ROC

Les algorithmes 1, 3 et 5 sont des **méthodes d’ensemble.** Ils ont été implémentés avec une mise à l’échelle, un suréchantillonnage et une optimisation du seuil. Cette approche semble très bien fonctionner pour notre jeu de données.

Les algorithmes 2 et 4 sont issus de la **classification des séries temporelles.** Ils ont été implémentés avec un tri par index chronologique, un découpage par ordre chronologique et une conversion en format "Panel". Bien que ROCKET nécessite généralement la normalisation des données[[8]](#endnote-8) (faite seulement pour l’algorithme 4 ici, avec MinMaxScaler), cela conduit étrangement à une légère dégradation du score de precision\_1 (–3%).

En-dessous du seuil de 83% pour precision\_1, le tiers central de la table est dominé par les approches basées sur les **arbres de décision** (RF / Random Forest et TSF / Time Series Forest), avec des scores allant de 80% à 64% selon les variations appliquées. L’algorithme RF, dont les deux instances ont été implémentées avec une mise à l’échelle et un suréchantillonnage, semble particulièrement bénéficier d’une optimisation du seuil (+13%). En revanche, c’est tout le contraire pour l’algorithme TSF : la différence entre les implémentations « basique » et « toutes options incluses » (avec mise à l’échelle et suréchantillonnage) est assez conséquente en faveur de la première (+15%), plus simple.

Enfin, en bas de la table, nous trouvons les approches basées sur les **distances** (LogReg / regréssion logistique et KNN-DTW / K-Nearest Neighbors avec Dynamic Time Warping). En dépit de la mise à l’échelle (obligatoire pour ces algorithmes) et du rééquilibrage (que ce soit par suréchantillonnage ou sous-échantillonnage), les scores de precision\_1 sont faibles (de 63% à 41%).

Pour compléter cette description des résultats, nous abordons les 3 algorithmes restants suivants :

6. Voting hard optimisé avec StandardScaler + SMOTE

12. DNN avec StandardScaler

19. ROCKET avec sort + TSS + StandardScaler + RandomUnderSampler +

tslearn

L’algorithme 6 fait partie des méthodes d’ensemble et affiche sans surprise des bonnes performances : son score de precision\_1 (81%) est juste légèrement en dessous du seuil (83%). En revanche, l’écart est assez important par rapport à l’algorithme 1, son « jumeau », qui le devance de 13%. Cela pourrait s’expliquer par la différence entre leurs méthodes de vote (le *soft* le remporte sur le *hard*) et l’optimisation du seuil (qui semble très bénéfique pour l’algorithme 1).

L’algorithme 12 correspond à un réseau de neurones dense. Son score de precision\_1 (73%) est très proche de la moyenne pour l’ensemble des algorithmes (71%).

Enfin, l’algorithme 19 est une implémentation « toutes options incluses » de ROCKET. Par rapport à l’algorithme 4, son « jumeau », il ajoute un rééquilibrage (effectué par sous-échantillonnage avec RandomUnderSampler : le suréchantillonnage avec SMOTE empêche le calcul d’aboutir, faute de mémoire vive). Cette manipulation est très néfaste pour le score de precision\_1, qui tombe de 66% entre les 2 algorithmes.

1. sklearn.preprocessing.StandardScaler [↑](#endnote-ref-1)
2. imblearn.over\_sampling.SMOTE [↑](#endnote-ref-2)
3. sklearn.metrics.roc\_curve [↑](#endnote-ref-3)
4. pandas.DataFrame.sort\_index [↑](#endnote-ref-4)
5. sklearn.model\_selection.TimeSeriesSplit [↑](#endnote-ref-5)
6. <https://tslearn.readthedocs.io/en/stable/> [↑](#endnote-ref-6)
7. sklearn.preprocessing.MinMaxScaler [↑](#endnote-ref-7)
8. <https://towardsdatascience.com/minirocket-fast-er-and-accurate-time-series-classification-cdacca2dcbfa> [↑](#endnote-ref-8)