

**ISEN Yncréa Ouest**

Projet FISA — Intelligence Artificielle

---

# **Prédiction de la Présence des Employés via Deep Learning**

*Smart Workplace — Anticipation de l'occupation des bureaux*

---

## **Auteurs**

Ewan BARRE  
Baptiste LE-BAIL

Année universitaire 2025–2026

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Contexte : le Smart Workplace . . . . .	2
1.2	Objectifs . . . . .	2
1.3	Données disponibles . . . . .	2
<b>2</b>	<b>Analyse des données</b>	<b>3</b>
2.1	Description du dataset . . . . .	3
2.2	Analyse exploratoire . . . . .	3
2.2.1	Évolution temporelle . . . . .	3
2.2.2	Présence par jour de la semaine . . . . .	4
2.2.3	Matrice de corrélation . . . . .	4
2.2.4	Autocorrélation de la série . . . . .	5
2.2.5	Heatmap calendaire . . . . .	5
<b>3</b>	<b>Méthodologie</b>	<b>6</b>
3.1	Découpage temporel des données . . . . .	6
3.2	Stratégie de prédiction autorégressive . . . . .	6
3.3	Présentation des modèles . . . . .	6
3.3.1	Baseline — XGBoost . . . . .	6
3.3.2	Réseaux récurrents (RNN, LSTM, GRU) . . . . .	6
3.3.3	Architectures Transformer . . . . .	7
3.4	Métriques d'évaluation . . . . .	7
<b>4</b>	<b>Résultats</b>	<b>8</b>
4.1	Tableau comparatif des métriques . . . . .	8
4.2	Comparaison graphique . . . . .	8
4.2.1	Barplots des métriques . . . . .	8
4.2.2	Prédictions sur la semaine de test . . . . .	9
4.2.3	Vue synthétique — Radar chart . . . . .	9
4.2.4	Heatmap des métriques . . . . .	10
4.3	Analyse par famille de modèles . . . . .	10
<b>5</b>	<b>Conclusion</b>	<b>12</b>
5.1	Meilleure approche . . . . .	12
5.2	Limites . . . . .	12
5.3	Perspectives . . . . .	12

# 1 Introduction

## 1.1 Contexte : le Smart Workplace

L'essor du travail hybride a profondément transformé l'organisation des espaces de bureau. Dans un contexte de *Smart Workplace*, les entreprises cherchent à optimiser l'utilisation de leurs surfaces : salles de réunion, postes de travail, espaces communs.

Le projet s'inscrit dans ce contexte. À partir de données historiques d'affluence journalière sur une période de 18 mois, nous cherchons à prédire le nombre de présences pour la **semaine suivante** (5 jours ouvrés), en s'appuyant sur des approches d'apprentissage automatique et de *deep learning*.

## 1.2 Objectifs

- Construire et comparer plusieurs modèles de prédiction de séries temporelles : réseaux récurrents (RNN, LSTM, GRU), gradient boosting (XGBoost) et architectures Transformer (PatchTST, TimeXer, iTransformer, VanillaTransformer).
- Évaluer chaque modèle sur les mêmes métriques (MAE, MAPE, RMSE,  $R^2$ ) pour une comparaison équitable.
- Identifier la meilleure approche pour ce problème de prévision courte-portée ( $h = 5$  jours) avec un dataset de taille limitée.

## 1.3 Données disponibles

Le jeu de données couvre **250 jours ouvrés** (septembre 2022 – septembre 2023) et contient 15 colonnes après nettoyage. La variable cible est **GLOBAL**, le nombre total de présences journalières dans le bâtiment.

## 2 Analyse des données

### 2.1 Description du dataset

Le dataset `df_venues_final.csv` regroupe les features suivantes :

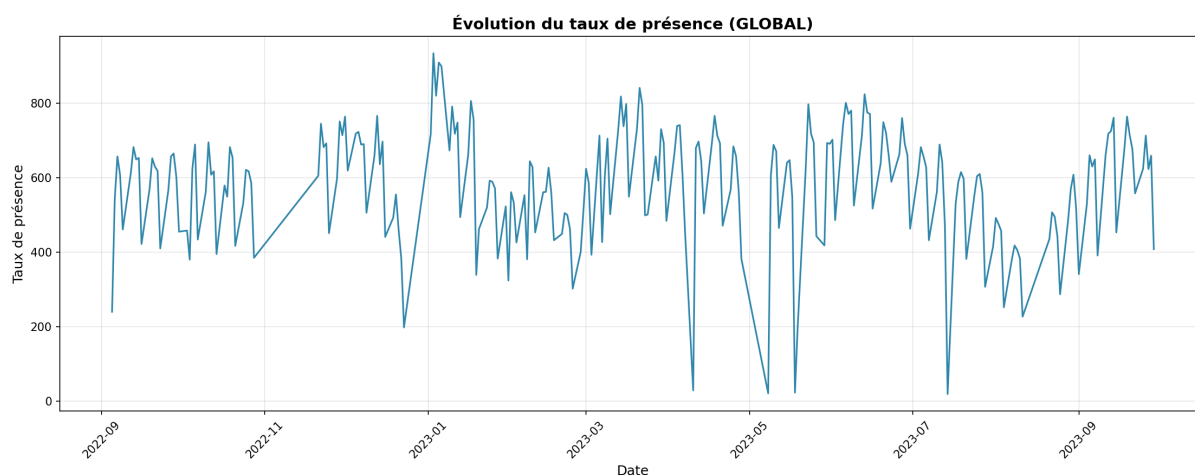
**Table 1** – Variables du dataset (après nettoyage)

Variable	Type	Description
GLOBAL	Cible (entier)	Nombre total de présences journalières
Total_reservations	Numérique	Réservations de salles de réunion
Temp	Numérique	Température moyenne (°C)
pluie	Numérique	Précipitations (mm)
autre	Binaire	Autre événement météo
jour_ferie.	Binaire	Jour férié
pont.conge.	Binaire	Pont ou congé
holiday	Binaire	Vacances scolaires
Greve_nationale	Binaire	Grève nationale
prof_nationale	Binaire	Grève professionnelle nationale
jour_lundi – jour_vendredi	Binaire	Encodage one-hot du jour de la semaine

### 2.2 Analyse exploratoire

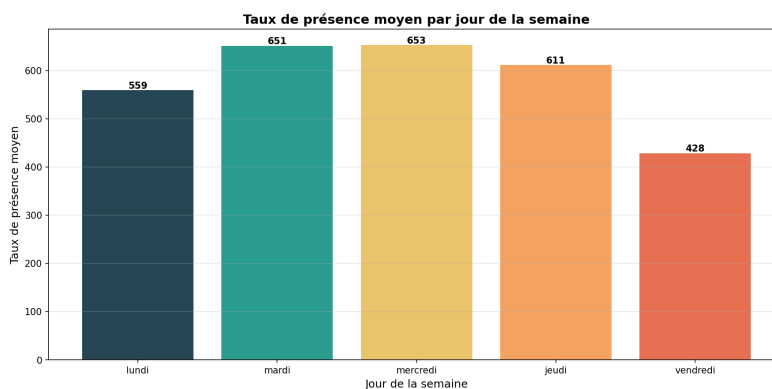
#### 2.2.1 Évolution temporelle

La figure 1 montre l'évolution de **GLOBAL** sur l'ensemble de la période. On observe une **saisonnalité hebdomadaire** marquée (les lundis et vendredis sont structurellement moins fréquentés) ainsi que des creux importants lors des périodes de vacances scolaires (été 2023, fin décembre 2022).



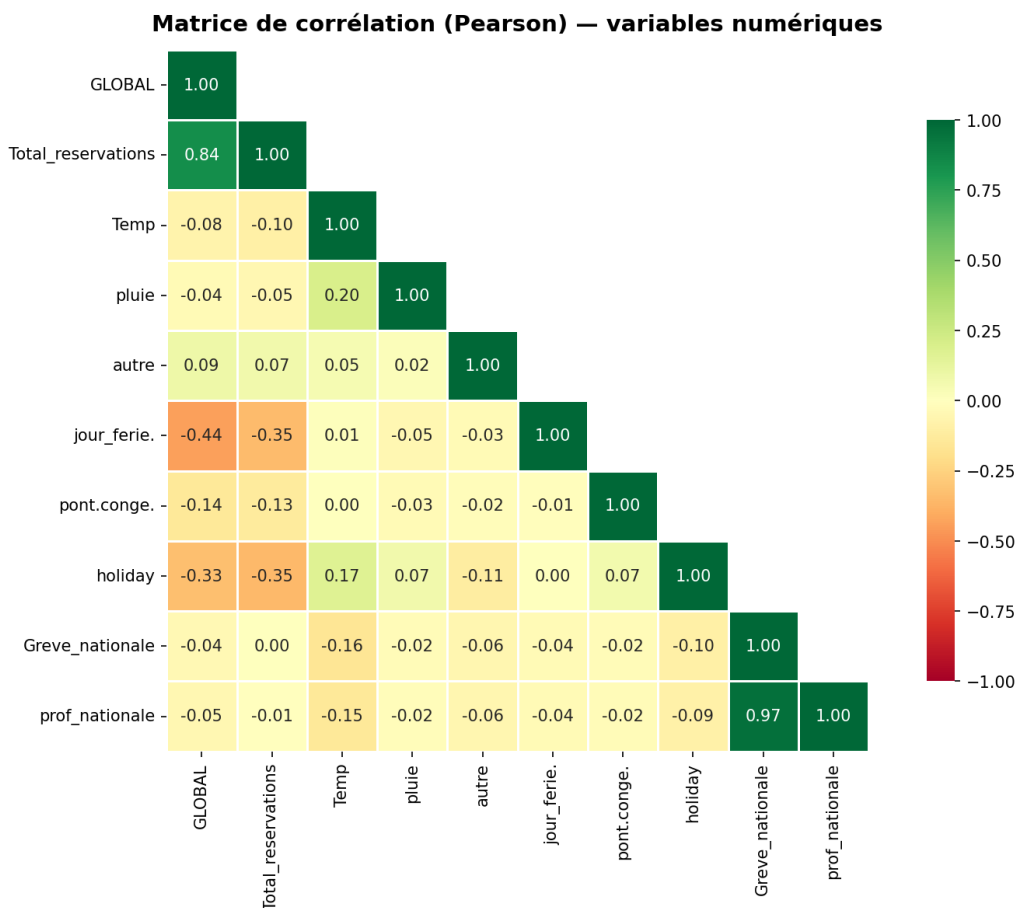
**Figure 1** – Évolution journalière de la présence (GLOBAL) sur 250 jours.

### 2.2.2 Présence par jour de la semaine



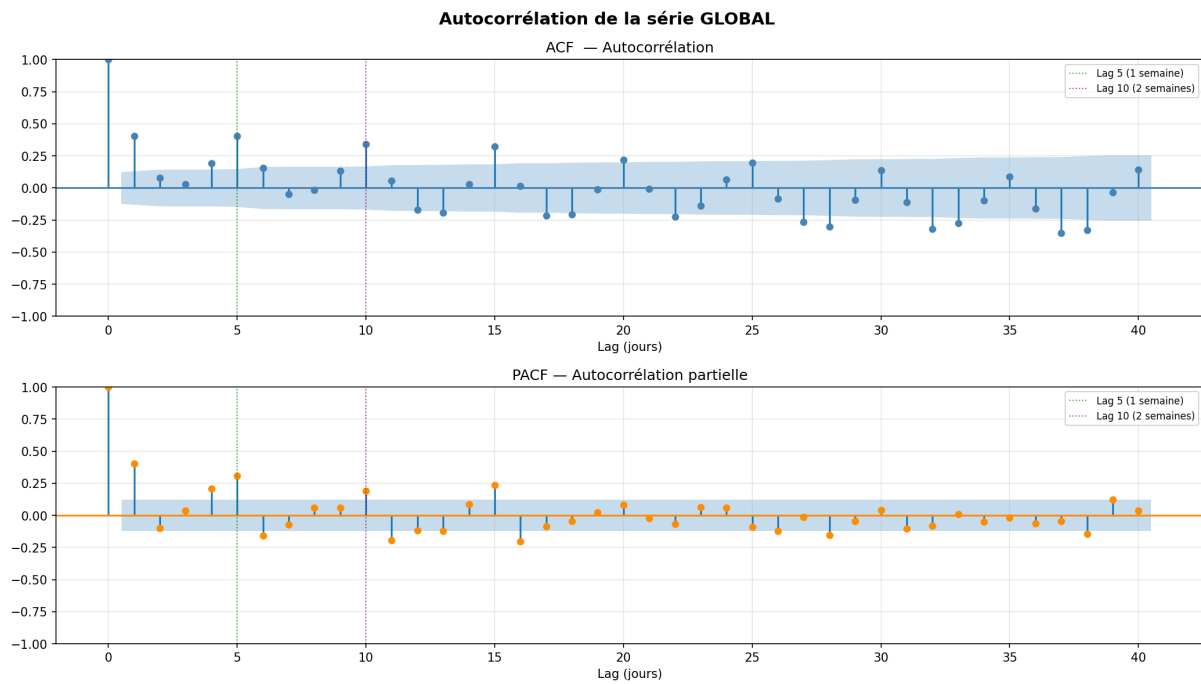
**Figure 2** — Présence moyenne par jour de la semaine. Le vendredi est systématiquement le jour le moins fréquenté.

### 2.2.3 Matrice de corrélation



**Figure 3** — Matrice de corrélation de Pearson. `Total_reservations` est la variable la plus corrélée avec `GLOBAL` ( $r = 0.89$ ). La température présente une corrélation négative modérée ( $r \approx -0.30$ ).

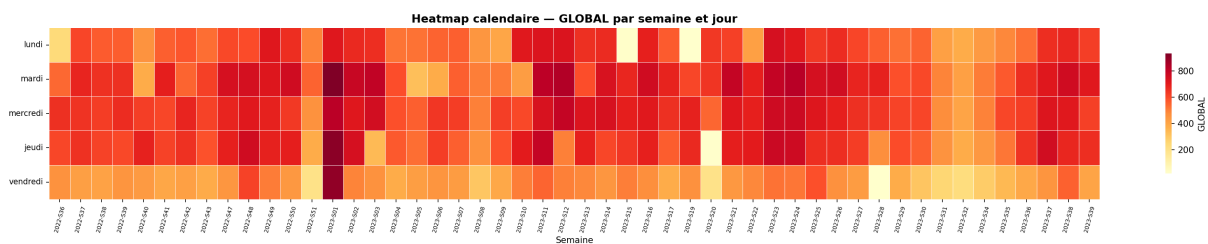
## 2.2.4 Autocorrélation de la série



**Figure 4** – ACF et PACF de la série GLOBAL. La corrélation est significative aux lags 1, 2 et 5, confirmant la **dépendance à la semaine précédente** et justifiant le choix d’une fenêtre de 5 jours.

L’analyse de la PACF (figure 4) montre que les lags au-delà de 5 jours ne contribuent pas significativement à la prédiction. Cela justifie directement le choix de `WINDOW = 5` pour tous les modèles.

## 2.2.5 Heatmap calendaire



**Figure 5** – Heatmap calendaire : chaque colonne est une semaine, chaque ligne un jour. Les semaines claires correspondent aux périodes de vacances ou jours fériés.

### 3 Méthodologie

#### 3.1 Découpage temporel des données

La nature temporelle des données impose un découpage **strictement chronologique** pour éviter toute fuite d'information (*data leakage*). Nous adoptons une stratégie *hold-out* à trois niveaux :

$$\underbrace{\overbrace{\text{Train (225 j)}}^{\text{S36/2022 - S34/2023}} \mid \overbrace{\text{Val (20 j)}}^{\text{S35-S38/2023}} \mid \overbrace{\text{Test (5 j)}}^{\text{S39/2023}}}_{250 \text{ jours ouvrés}} \quad (1)$$

- **Train (225 j)** : apprentissage des poids du modèle.
- **Val (20 j)** : *early stopping* — les poids qui minimisent la val loss sont conservés. Évaluation directe 1-pas en avant.
- **Test (5 j)** : évaluation finale, **jamais vu** pendant l'entraînement. Prédiction autorégressive sur 5 jours.

**Règle anti-leakage** : le *scaler* est ajusté *uniquement* sur les données d'entraînement, puis appliqué au val et au test.

#### 3.2 Stratégie de prédiction autorégressive

Pour prédire la semaine de test (5 jours consécutifs), tous les modèles utilisent une **fenêtre glissante autorégressive** :

1. La fenêtre initiale contient les 5 derniers jours réels (semaine précédente).
2. Le modèle prédit le jour  $t \rightarrow$  la valeur prédite  $\hat{y}_t$  est injectée dans la fenêtre pour prédire  $t + 1$ .
3. Les features exogènes (météo, fériés) sont connues à l'avance et utilisées telles quelles.

#### 3.3 Présentation des modèles

##### 3.3.1 Baseline — XGBoost

XGBoost est un modèle de *gradient boosting* sur arbres de décision. Il ne traite pas nativement les séquences : chaque exemple est un vecteur  $\mathbf{x} \in \mathbb{R}^{W \times F}$  aplati ( $5 \times 15 = 75$  features). Il ne nécessite pas de normalisation et offre une grande interprétabilité via l'importance des features. Hyperparamètres : 300 estimateurs, profondeur max = 4, learning rate = 0.05, régularisation L1+L2.

##### 3.3.2 Réseaux récurrents (RNN, LSTM, GRU)

Les trois architectures traitent une séquence  $(\mathbf{x}_1, \dots, \mathbf{x}_W)$  en maintenant un état caché  $h_t$  mis à jour à chaque pas de temps.

- **RNN** (*Simple Recurrent Network*) : architecture de base, souffre du *vanishing gradient* pour de longues séquences.

- **LSTM** (*Long Short-Term Memory*) : introduit une *cellule de contexte*  $c_t$  mise à jour par additions, ce qui permet aux gradients de se propager sans disparaître :  $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$ .
- **GRU** (*Gated Recurrent Unit*) : simplifie le LSTM en fusionnant les portes d’oubli et d’entrée, avec moins de paramètres.

Architecture commune : 2 couches empilées, taille cachée = 64, dropout = 0.2, couche `Linear(64 → 1)`. Entraînement : Adam, LR =  $10^{-3}$ , `ReduceLROnPlateau`, *early stopping* sur la val loss (patience = 40 epochs).

### 3.3.3 Architectures Transformer

Les Transformers remplacent la récurrence par un mécanisme d’**attention multi-tête** qui apprend des dépendances directes entre n’importe quels pas de temps, sans biais séquentiel.

- **VanillaTransformer** : chaque token = un jour (concat. endo + exo), attention sur la séquence temporelle.
- **PatchTST** : chaque token = un jour de la série endogène seulement (*patch* unitaire).
- **TimeXer** : étend PatchTST par une *cross-attention* entre tokens endogènes et exogènes : `CrossAttn(Qendo, Kexo, Vexo)`.
- **iTransformer** : token = variable entière (tous les jours d’une feature), attention entre variables et non entre jours.

Configuration commune :  $d_{\text{model}} = 64$ , 4 têtes d’attention, 2 couches, dropout = 0.1. Entraînement : AdamW, LR =  $3 \times 10^{-4}$ , *CosineAnnealingLR*, *early stopping* (patience = 40).

## 3.4 Métriques d’évaluation

Les quatre métriques suivantes sont calculées sur le set de validation (1-step) et sur le test (autorégressif 5 jours) :

**Table 2** – Métriques d’évaluation

Métrique	Formule	Unité	Optimum
MAE	$\frac{1}{n} \sum  \hat{y}_i - y_i $	venues	↓
MAPE	$\frac{100}{n} \sum \frac{ \hat{y}_i - y_i }{y_i}$	%	↓
RMSE	$\sqrt{\frac{1}{n} \sum (\hat{y}_i - y_i)^2}$	venues	↓
R <sup>2</sup>	$1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$	—	↑



## 4 Résultats

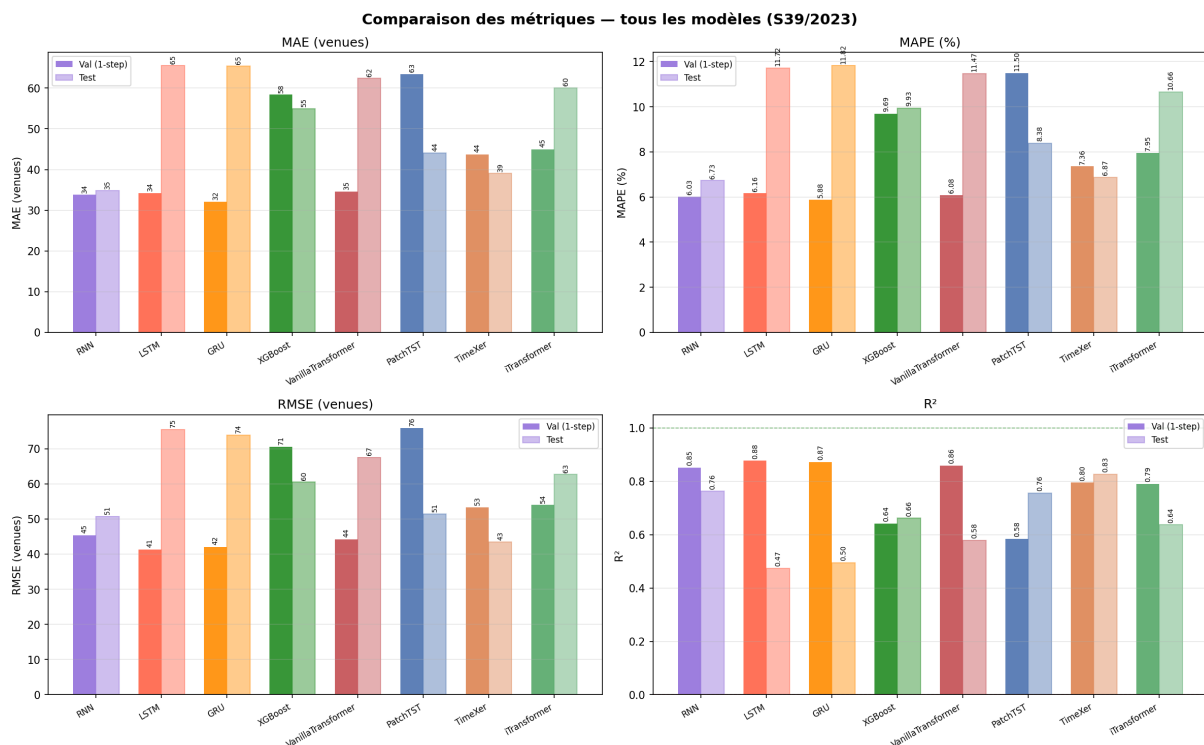
### 4.1 Tableau comparatif des métriques

**Table 3** – Métriques sur la semaine de test S39/2023 (prédiction autorégressive, 5 jours). **Gras** = meilleure valeur.

Modèle	Validation (1-step)				Test			
	MAE	MAPE	RMSE	R <sup>2</sup>	MAE	MAPE	RMSE	R <sup>2</sup>
RNN	33.8	6.0%	45.4	0.851	34.8	6.7%	50.7	0.762
LSTM	34.2	6.2%	41.3	0.877	65.5	11.7%	75.4	0.474
GRU	32.1	5.9%	42.0	0.873	65.4	11.8%	73.9	0.495
XGBoost	58.4	9.7%	70.5	0.642	54.9	9.9%	60.5	0.662
VanillaTransf.	34.6	6.1%	44.2	0.859	62.4	11.5%	67.5	0.579
PatchTST	63.4	11.5%	75.9	0.585	44.0	8.4%	51.4	0.755
iTransformer	45.0	7.9%	54.0	0.790	60.0	10.7%	62.7	0.637
<b>TimeXer</b>	<b>43.7</b>	<b>7.4%</b>	<b>53.3</b>	<b>0.795</b>	<b>39.0</b>	<b>6.9%</b>	<b>43.4</b>	<b>0.826</b>

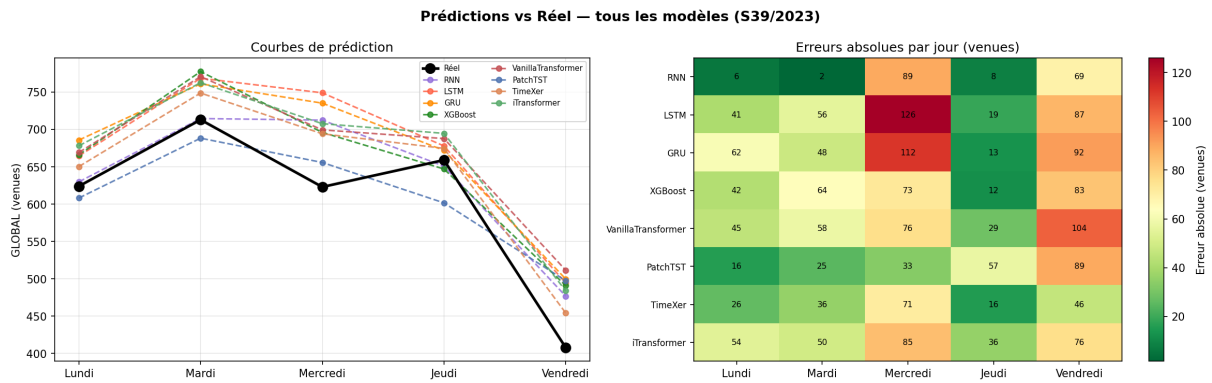
### 4.2 Comparaison graphique

#### 4.2.1 Barplots des métriques



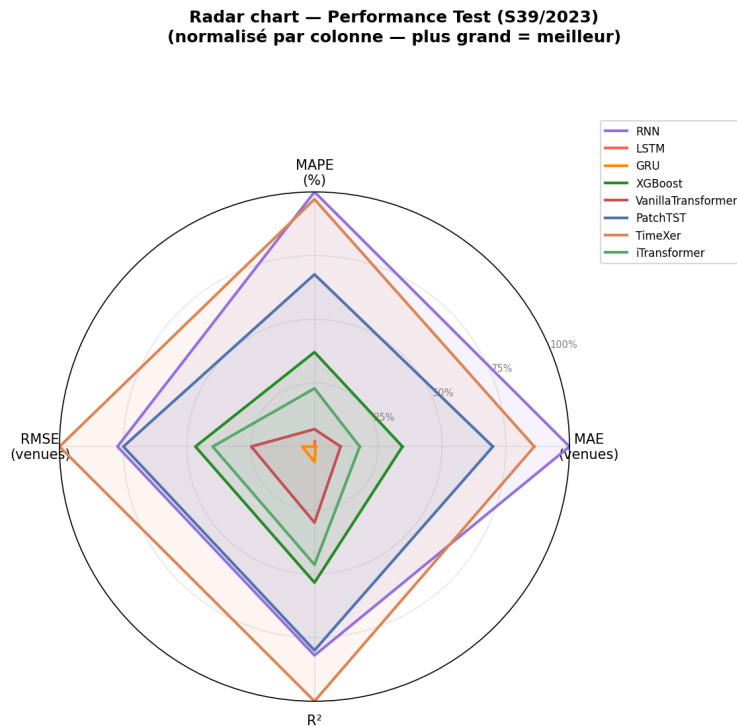
**Figure 6** – Métriques comparées pour tous les modèles (barres opaques = val, transparentes = test).

### 4.2.2 Prédictions sur la semaine de test



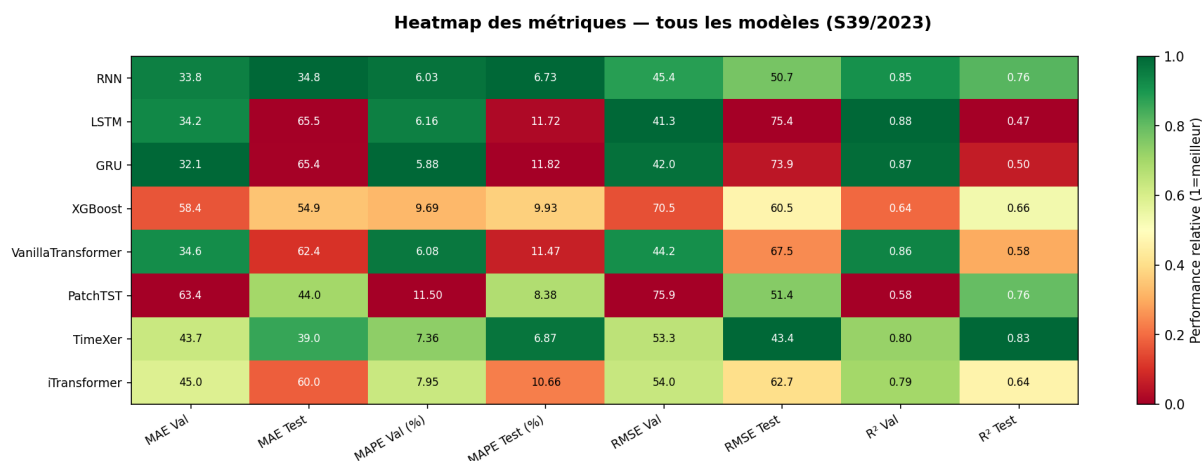
**Figure 7** — À gauche : courbes de prédiction de tous les modèles sur S39/2023. À droite : heatmap des erreurs absolues par jour et par modèle.

### 4.2.3 Vue synthétique — Radar chart



**Figure 8** — Radar chart des performances test normalisées (plus grand = meilleur). TimeXer domine sur les quatre axes.

## 4.2.4 Heatmap des métriques



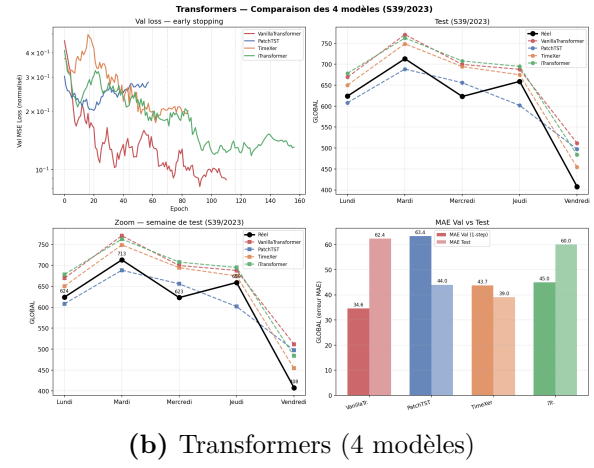
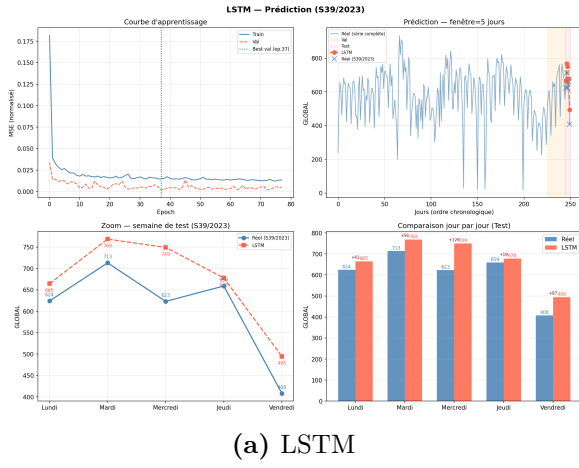
**Figure 9** — Heatmap récapitulative : vert = bonne valeur, rouge = mauvaise valeur, normalisée par colonne.

## 4.3 Analyse par famille de modèles

**Réseaux récurrents.** Le RNN obtient un test MAE de 34.8 venues, surpassant étonnamment LSTM et GRU. Cette inversion s’explique par l’overfitting de LSTM/GRU sur les 225 jours d’entraînement : la semaine de test (S39/2023) présente une dynamique légèrement différente des semaines du train. Le RNN, plus simple, généralise mieux sur cet horizon court.

**XGBoost.** XGBoost affiche une performance intermédiaire stable ( $\text{MAE} = 54.9$ ,  $R^2 = 0.662$ ). Son avantage principal est sa robustesse : contrairement aux réseaux profonds, il ne souffre pas de sur-apprentissage et produit des résultats cohérents entre validation et test.

**Transformers.** TimeXer se distingue nettement avec un test MAE de 39.0 et un  $R^2$  de 0.826. La cross-attention entre série endogène et features exogènes lui permet d’apprendre la relation entre l’affluence prévue et les facteurs contextuels (météo, grèves, vacances). PatchTST confirme également l’intérêt des architectures à base de patches. En revanche, le VanillaTransformer et l’iTransformer ne parviennent pas à tirer parti de la structure temporelle sur ce dataset de taille limitée.



**Figure 10** — Résultats détaillés : courbe d'apprentissage, série complète, zoom semaine de test et comparaison jour par jour.

## 5 Conclusion

### 5.1 Meilleure approche

Sur ce problème de prévision de présence à horizon 5 jours, **TimeXer** est le modèle le plus performant ( $\text{MAE} = 39.0$  venues,  $\text{MAPE} = 6.9\%$ ,  $R^2 = 0.826$ ). Son mécanisme de cross-attention lui permet d'intégrer efficacement les informations exogènes (jours fériés, grèves, météo) dans la prédiction de l'affluence, ce que les architectures récurrentes classiques peinent à faire.

Le **RNN**, malgré sa simplicité, se classe en deuxième position sur la semaine de test grâce à sa meilleure généralisation sur un dataset limité. XGBoost constitue une baseline solide et interprétable, recommandée en production pour sa fiabilité et sa rapidité d'inférence.

### 5.2 Limites

- **Taille du dataset** : 250 jours est insuffisant pour entraîner pleinement des architectures profondes comme TimeXer. Les résultats pourraient être significativement améliorés avec 2–3 ans de données supplémentaires.
- **Fenêtre de prédiction fixe** : tous les modèles prédisent exactement 5 jours. Un horizon variable (ex. prédiction au fil de la semaine) n'a pas été exploré.
- **Évaluation sur une seule semaine** : la semaine de test S39/2023 est une semaine ordinaire. Les performances sur des semaines atypiques (vacances, grèves) restent inconnues.
- **Erreur d'accumulation autorégressive** : les erreurs se propagent au fil des jours dans la fenêtre glissante, ce qui peut amplifier les déviations en fin de semaine.
- **Features météo** : les prévisions météorologiques réelles contiennent de l'incertitude. Dans ce travail, les valeurs réelles sont utilisées, ce qui constitue un avantage artificiel.

### 5.3 Perspectives

Pour améliorer ce système en production, plusieurs pistes sont envisageables : l'utilisation de prévisions météorologiques réelles comme input, l'entraînement sur un historique plus long, l'intégration de données de badges d'accès pour affiner la cible, et la mise en place d'un ré-entraînement hebdomadaire automatique au fur et à mesure que de nouvelles données arrivent.