

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET
POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE



UNIVERSITÉ M'HAMED BOUGUERRA –
BOUMERDES

FACULTÉ DES SCIENCES
DÉPARTEMENT DE L'INFORMATIQUE

PROJET DE DATA MINING

Prédiction de température (Séries temporelles)

Techniques utilisées : LSTM, Random Forest, Régression Linéaire

Rédigé par :
SILMI Hadjer
KENNICHE Malika Roumaissa

Groupe : 02

Encadrant :
Sid Ahmed BOUDAOU

Année universitaire :
2025/2026

Table des matières

1	Introduction	4
1.1	Contexte et Motivation	4
1.2	Problématique	4
1.3	Objectifs du Projet	5
2	Jeu de Données et Préparation	7
2.1	Description du jeu de données	7
2.2	Nettoyage et préparation des données	7
2.2.1	Étapes communes de prétraitement	8
2.2.2	Préparation spécifique pour le LSTM	8
2.2.3	Préparation spécifique pour le Random Forest	9
2.2.4	Préparation spécifique pour la Régression Linéaire	9
2.2.5	Visualisation et analyse exploratoire	9
3	Les Méthodes : LSTM vs Random Forest vs Régression linéaire	10
3.1	Long Short-Term Memory -LSTM-	10
3.1.1	Problématique du Gradient Évanescence	10
3.1.2	Architecture du Constant Error Carousel (CEC)	10
3.1.3	Les Portes Multiplicatives	11
3.1.4	Équations Mathématiques du LSTM	12
3.1.5	Propriétés Clés du LSTM	13
3.1.6	Implémentation pour la Prédiction de Température	14
3.1.7	Pertinence du LSTM pour la Prévision Météorologique	16
3.1.8	Limites et Perspectives	18
3.1.9	Conclusion	18
3.2	Random Forest	19
3.2.1	Introduction	19
3.2.2	Principe de fonctionnement	19
3.2.3	Configuration et justification des hyperparamètres	21
3.2.4	Préparation des données pour Random Forest	23
3.2.5	Évaluation des performances	24
3.2.6	Avantages et limitations	25
3.2.7	Conclusion	26
3.3	Régression Linéaire	26
3.3.1	Introduction	26
3.3.2	Principe de fonctionnement	27
3.3.3	Préparation des données	27
3.3.4	Normalisation et division des données	28

3.3.5	Entraînement et prédition	28
3.3.6	Métriques d'évaluation	29
3.3.7	Visualisation et analyse	30
3.3.8	Avantages et limitations	30
3.3.9	Conclusion	30
4	Test et Analyse	32
4.1	Présentation des résultats	32
4.1.1	Résultats quantitatifs	32
4.2	Analyse comparative	32
4.2.1	Performance globale	32
4.2.2	Écart Train-Test : Analyse du surapprentissage	33
4.2.3	Précision des prédictions	33
4.2.4	Variance expliquée (R^2)	34
4.3	Analyse qualitative	34
4.3.1	Complexité et interprétabilité	34
4.3.2	Efficacité computationnelle	34
4.3.3	Robustesse et maintenance	35
5	Conclusion	36
5.1	Résumé du projet	36
5.2	Principaux enseignements	36

Chapitre 1

Introduction

1.1 Contexte et Motivation

La prévision météorologique, et plus spécifiquement la prédiction des températures, constitue un enjeu scientifique et socio-économique majeur aux implications multidimensionnelles. Dans le domaine agricole, elle conditionne les périodes de semis, d'irrigation et de récolte, influençant directement la sécurité alimentaire. Le secteur énergétique s'appuie sur ces prévisions pour optimiser la production et la distribution d'électricité, anticipant les pics de demande liés aux conditions climatiques. Les transports, la logistique, la santé publique, la planification d'événements et même la stratégie militaire intègrent ces données dans leurs processus décisionnels.

L'émergence de l'ère du *Big Data* a provoqué une transformation paradigmique dans le domaine météorologique. Les stations météorologiques, satellites, radars et modèles numériques génèrent désormais des volumes de données sans précédent, dépassant les capacités d'analyse traditionnelle. Parallèlement, les avancées récentes en intelligence artificielle, notamment en *Deep Learning*, offrent des outils computationnels capables d'extraire des motifs complexes, des relations non-linéaires et des dépendances temporelles subtiles que les méthodes statistiques conventionnelles peinent à modéliser.

Cette convergence entre la disponibilité de données massives et la sophistication des algorithmes d'apprentissage automatique ouvre la voie à une nouvelle génération de modèles prédictifs météorologiques, plus précis, plus robustes et capable d'intégrer une multiplicité de facteurs environnementaux.

1.2 Problématique

La prédiction de la température, bien que fondamentale, demeure un défi scientifique persistant en raison de la nature intrinsèquement complexe, dynamique et chaotique des systèmes climatiques. Cette complexité se manifeste à plusieurs niveaux :

- **Dépendances temporelles multi-échelles** : Les phénomènes météorologiques présentent des corrélations à court terme (variations diurnes), moyen terme (cycles saisonniers) et long terme (phénomènes climatiques).
- **Interactions non-linéaires** : Les relations entre les variables atmosphériques (température, humidité, pression, vent) sont rarement additives ou proportionnelles.
- **Sensibilité aux conditions initiales** : Le caractère chaotique de l'atmosphère,

illustré par l'« effet papillon », rend les prévisions sensibles aux moindres incertitudes dans les données d'entrée.

- **Hétérogénéité spatiale** : Les variations topographiques (altitude, proximité maritime, urbanisation) créent des microclimats difficiles à capturer par des modèles globaux.

Les approches traditionnelles, telles que les modèles de régression linéaire ou les méthodes ARIMA, atteignent rapidement leurs limites face à ces défis. Elles peinent notamment à modéliser les interactions complexes et les dépendances à long terme, conduisant souvent à une sous-performance dans les scénarios de variabilité climatique accrue.

Problématique centrale

Dans un contexte de données météorologiques massives et multidimensionnelles, comment concevoir et comparer des modèles d'apprentissage automatique capables de capturer les dépendances temporelles complexes et les relations non-linéaires pour prédire la température avec une précision supérieure aux approches classiques, tout en identifiant les déterminants climatiques les plus influents ?

1.3 Objectifs du Projet

Ce projet de recherche vise à développer une approche comparative systématique pour la prédiction de la température maximale (`Next_Tmax`) à un horizon de 24 heures. L'étude se structure autour de trois méthodologies complémentaires, chacune explorant une famille d'algorithmes distincte, afin d'évaluer leurs mérites respectifs sur un jeu de données météorologiques concret.

Les objectifs spécifiques sont les suivants :

1. **Exploration et prétraitement des données** :
 - Analyser un jeu de données météorologiques multivarié et multi-stations.
 - Mettre en œuvre un pipeline robuste de nettoyage, d'imputation et d'ingénierie des caractéristiques.
 - Créer des variables dérivées (décalages temporels, moyennes mobiles, encodages cycliques) pour capturer les dynamiques temporelles.
2. **Implémentation et optimisation des modèles** :
 - Développer un modèle **LSTM** (Long Short-Term Memory) pour exploiter explicitement les dépendances séquentielles à long terme dans les données temporelles.
 - Implémenter un modèle **Random Forest** comme représentant des méthodes ensemblistes, robuste aux interactions non-linéaires et offrant une interprétabilité via l'importance des caractéristiques.
 - Utiliser une **Régression Linéaire** comme modèle de référence (baseline) pour quantifier l'apport des approches plus complexes.
3. **Évaluation comparative rigoureuse** :
 - Évaluer les performances des modèles à l'aide d'une batterie de métriques statistiques (MAE, RMSE, R^2 , MAPE).
 - Analyser les résultats sur des ensembles de validation et de test distincts pour garantir la robustesse des conclusions.
 - Visualiser les prédictions, les erreurs et les comportements des modèles pour une analyse qualitative complémentaire.

4. Analyse et recommandations :

- Identifier l'algorithme le plus performant et le plus adapté aux spécificités des données météorologiques.
- Discuter des forces, faiblesses et compromis (précision vs complexité, temps d'entraînement vs performance) de chaque approche.
- Formuler des recommandations pratiques pour la mise en œuvre de systèmes de prévision opérationnels.

À travers cette démarche, ce travail contribue à la littérature sur l'application de l'apprentissage automatique en météorologie et fournit un cadre méthodologique reproductible pour l'évaluation comparative de modèles prédictifs sur des séries temporelles environnementales.

Chapitre 2

Jeu de Données et Préparation

2.1 Description du jeu de données

Le jeu de données utilisé dans cette étude est le *Temperature Forecast Project Dataset*, disponible publiquement sur la plateforme Kaggle à l'adresse suivante : <https://www.kaggle.com/datasets/smokingkrils/temperature-forecast-project-using-ml>. Ce dataset contient des données météorologiques historiques collectées à partir de plusieurs stations en Corée du Sud, avec pour objectif principal la prévision des températures maximales et minimales du jour suivant.

Le jeu de données original comprend 7 752 observations et 26 colonnes. Les variables principales incluent :

- **Variables temporelles** : Date (format JJ-MM-AAAA)
- **Variables d'identification** : Identifiant de la station (station)
- **Variables météorologiques actuelles** : Température maximale présente (Present_Tmax), Température minimale présente (Present_Tmin)
- **Variables de prévision LDAPS** : Humidité relative minimale/maximale (LDAPS_RHmin, LDAPS_RHmax), Températures avec lapse rate (LDAPS_Tmax_lapse, LDAPS_Tmin_lapse), Vitesse du vent (LDAPS_WS), Hauteur de la couche limite (LDAPS_LH), Couverture nuageuse sur 4 couches (LDAPS_CC1-4), Précipitations sur 4 couches (LDAPS_PPT1-4)
- **Variables géographiques** : Latitude (lat), Longitude (lon), Altitude (DEM), Pente (Slope)
- **Variables de rayonnement** : Rayonnement solaire (Solar radiation)
- **Variables cibles** : Température maximale du jour suivant (Next_Tmax), Température minimale du jour suivant (Next_Tmin)

L'analyse initiale a révélé la présence de valeurs manquantes dans plusieurs colonnes, nécessitant un prétraitement approprié avant l'application des algorithmes d'apprentissage automatique.

2.2 Nettoyage et préparation des données

Une procédure systématique de nettoyage et de préparation des données a été mise en œuvre pour garantir la qualité des données d'entrée pour les trois algorithmes (LSTM, Random Forest et Régression Linéaire).

2.2.1 Étapes communes de prétraitement

Les étapes suivantes ont été appliquées de manière uniforme aux trois approches :

1. **Conversion du format de date** : La colonne 'Date' a été convertie au format datetime et les données ont été triées chronologiquement par station.
2. **Gestion des valeurs manquantes des variables cibles** : Toutes les lignes où les variables cibles 'Next_Tmax' et 'Next_Tmin' étaient manquantes ont été supprimées, car elles ne pouvaient pas être utilisées pour l'apprentissage supervisé.
3. **Imputation des valeurs manquantes** : Pour chaque station individuellement, les valeurs manquantes dans les autres colonnes numériques ont été interpolées linéairement, préservant ainsi la continuité temporelle des séries.
4. **Ingénierie des caractéristiques temporelles** :
 - Extraction du mois (1-12) et du jour de l'année (1-365)
 - Création de caractéristiques cycliques via transformations trigonométriques :

$$\begin{aligned} \text{Month_sin} &= \sin\left(2\pi \times \frac{\text{Month}}{12}\right) \\ \text{Month_cos} &= \cos\left(2\pi \times \frac{\text{Month}}{12}\right) \\ \text{DayOfYear_sin} &= \sin\left(2\pi \times \frac{\text{DayOfYear}}{365}\right) \\ \text{DayOfYear_cos} &= \cos\left(2\pi \times \frac{\text{DayOfYear}}{365}\right) \end{aligned}$$

5. **Création de variables retardées (lag features)** : Pour exploiter l'autocorrélation temporelle, des décalages de 1, 2, 3 et 7 jours ont été créés pour les températures actuelles ('Present_Tmax' et 'Present_Tmin').
6. **Création de moyennes mobiles** : Des moyennes mobiles sur 3 et 7 jours ont été calculées pour capturer les tendances à court terme des températures.
7. **Suppression des lignes avec valeurs manquantes résiduelles** : Après la création des variables retardées, toutes les lignes contenant encore des valeurs manquantes ont été éliminées.

2.2.2 Préparation spécifique pour le LSTM

Pour l'algorithme LSTM, une préparation supplémentaire a été nécessaire :

- **Normalisation** : Toutes les caractéristiques et la variable cible ont été normalisées dans l'intervalle [0, 1] à l'aide de `MinMaxScaler`.
- **Création de séquences** : Les données ont été transformées en séquences temporelles de 7 jours (time steps) pour prédire la température du jour suivant, conformément à l'architecture des réseaux LSTM qui exploitent les dépendances temporelles.
- **Division temporelle** : Le dataset a été divisé en ensembles d'entraînement (70%), validation (15%) et test (15%) en respectant l'ordre chronologique, sans mélange aléatoire, pour préserver la structure temporelle.

2.2.3 Préparation spécifique pour le Random Forest

Pour l'algorithme Random Forest :

- **Normalisation** : Seules les caractéristiques d'entrée ont été normalisées, tandis que la variable cible est restée dans son échelle originale.
- **Division temporelle** : Une division similaire à celle du LSTM a été appliquée (70% entraînement, 15% validation, 15% test) sans mélange aléatoire.
- **Sélection d'hyperparamètres** : Une recherche manuelle a été effectuée pour optimiser les paramètres suivants : 900 estimateurs, profondeur maximale de 28, minimum de 6 échantillons pour diviser un nœud, minimum de 3 échantillons par feuille, et racine carrée du nombre total de caractéristiques pour la sélection à chaque division.

2.2.4 Préparation spécifique pour la Régression Linéaire

Pour la régression linéaire, une approche différente a été adoptée :

- **Sélection de caractéristiques** : Seules les variables présentant une corrélation absolue supérieure à 0,3 avec la variable cible 'Next_Tmax' ont été conservées, réduisant ainsi la dimensionnalité et les risques de sur-ajustement.
- **Standardisation** : Les caractéristiques ont été standardisées (moyenne=0, écart-type=1) à l'aide de `StandardScaler`, ce qui est particulièrement important pour la régression linéaire.
- **Division aléatoire** : Contrairement aux autres algorithmes, les données ont été divisées aléatoirement en 80% pour l'entraînement et 20% pour le test, car la régression linéaire ne dépend pas de l'ordre temporel.

2.2.5 Visualisation et analyse exploratoire

Avant et après le nettoyage, plusieurs visualisations ont été générées :

- **Diagramme des valeurs manquantes** : Un diagramme en barres montrant le nombre de valeurs manquantes par colonne avant traitement.
- **Matrice de corrélation** : Une heatmap des corrélations entre variables numériques pour identifier les relations linéaires potentielles.
- **Distribution des erreurs** : Histogrammes des erreurs de prédiction sur l'ensemble de test pour chaque algorithme.
- **Importance des caractéristiques** : Diagramme en barres horizontales montrant l'importance relative des caractéristiques pour le modèle Random Forest.

Cette préparation rigoureuse des données a permis d'obtenir des jeux de données de haute qualité pour l'entraînement et l'évaluation des trois algorithmes, tout en adaptant la préparation aux spécificités de chaque approche.

Chapitre 3

Les Méthodes : LSTM vs Random Forest vs Régression linéaire

3.1 Long Short-Term Memory -LSTM-

3.1.1 Problématique du Gradient Évanescence

Les réseaux de neurones récurrents (RNN) traditionnels rencontrent des difficultés majeures lors de l'apprentissage de dépendances à long terme dans les séries temporelles. Ce problème, identifié par Hochreiter en 1991, est causé par le phénomène de **vanishing gradient** (gradient évanescence) : lors de la rétropropagation à travers le temps, les gradients tendent à diminuer exponentiellement, rendant l'apprentissage de relations distantes pratiquement impossible.

Dans un RNN standard, lorsqu'une erreur est rétropropagée sur q pas de temps, elle est multipliée à chaque étape par le produit de la dérivée de la fonction d'activation et du poids de connexion récurrente. Mathématiquement, pour une unité récurrente u , le gradient est proportionnel à :

$$\frac{\partial \varepsilon_v(t-q)}{\partial \varepsilon_u(t)} = \sum_{l_1=1}^n \cdots \sum_{l_{q-1}=1}^n \prod_{m=1}^q f'_{l_m}(\text{net}_{l_m}(t-m)) \cdot w_{l_m l_{m-1}} \quad (3.1)$$

Si le produit $|f'(\text{net}) \cdot w| < 1$ pour tous les pas de temps, le gradient diminue exponentiellement avec q , conduisant à un apprentissage extrêmement lent ou inefficace. Inversement, si $|f'(\text{net}) \cdot w| > 1$, le gradient explose, causant des oscillations des poids et une instabilité de l'apprentissage.

Le réseau Long Short-Term Memory (LSTM), introduit par Hochreiter et Schmidhuber en 1997 propose une architecture innovante spécialement conçue pour résoudre ce problème et permettre l'apprentissage de dépendances temporelles sur des intervalles de temps très longs, dépassant 1000 pas de temps.

3.1.2 Architecture du Constant Error Carousel (CEC)

Le cœur de l'architecture LSTM est le *Constant Error Carousel* (CEC), un mécanisme qui maintient un flux d'erreur constant à travers le temps. Pour une cellule mémoire c_j avec une connexion récurrente à elle-même, le flux d'erreur constant est assuré en imposant :

$$f'_j(\text{net}_{c_j}(t)) \cdot w_{jj} = 1.0 \quad (3.2)$$

Cela nécessite une fonction d'activation linéaire $f_j(x) = x$ et un poids $w_{jj} = 1.0$. Cette propriété fondamentale garantit que l'état interne de la cellule reste constant dans le temps :

$$y_j(t+1) = f_j(\text{net}_j(t+1)) = f_j(w_{jj} \cdot y_j(t)) = y_j(t) \quad (3.3)$$

3.1.3 Les Portes Multiplicatives

Une cellule mémoire LSTM complète intègre le CEC avec trois unités multiplicatives appelées **portes** (gates) qui contrôlent le flux d'information :

- **Porte d'entrée (input gate)** : Protège le contenu de la mémoire contre les perturbations causées par des entrées non pertinentes. Elle détermine quelle proportion de la nouvelle information doit être stockée.
- **Porte de sortie (output gate)** : Protège les autres unités du réseau contre les perturbations causées par des contenus mémoire actuellement non pertinents. Elle détermine quelle proportion de l'état doit être exposée.
- **Porte d'oubli (forget gate)** : Permet à la cellule de réinitialiser son état interne lorsque l'information stockée n'est plus nécessaire. Elle détermine quelle proportion de l'ancien état doit être conservée.

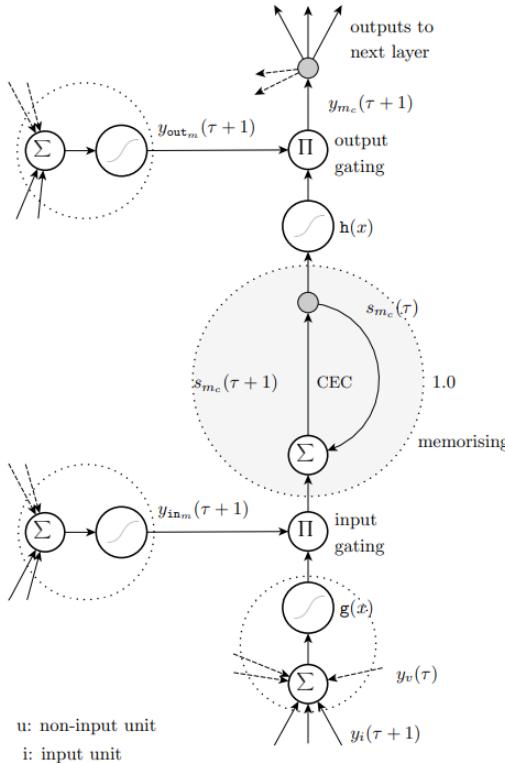


FIGURE 3.1 – Architecture détaillée d'une cellule mémoire LSTM montrant le Constant Error Carousel (CEC) avec poids de 1.0, les portes d'entrée et de sortie, et les fonctions d'activation $g(x)$ et $h(x)$.

3.1.4 Équations Mathématiques du LSTM

L'architecture d'une cellule mémoire LSTM peut être formalisée mathématiquement comme suit :

Porte d'entrée

$$\text{net}_j^{\text{in}}(t) = \sum_u w_{ju}^{\text{in}} \cdot y_u(t-1) \quad (3.4)$$

$$y_j^{\text{in}}(t) = \sigma(\text{net}_j^{\text{in}}(t)) \quad (3.5)$$

Porte de sortie

$$\text{net}_j^{\text{out}}(t) = \sum_u w_{ju}^{\text{out}} \cdot y_u(t-1) \quad (3.6)$$

$$y_j^{\text{out}}(t) = \sigma(\text{net}_j^{\text{out}}(t)) \quad (3.7)$$

Porte d'oubli

$$\text{net}_j^{\varphi}(t) = \sum_u w_{ju}^{\varphi} \cdot y_u(t-1) \quad (3.8)$$

$$y_j^{\varphi}(t) = \sigma(\text{net}_j^{\varphi}(t) + b_j^{\varphi}) \quad (3.9)$$

État interne (CEC)

$$s_{c_j}(t) = s_{c_j}(t-1) \cdot y_j^{\varphi}(t) + y_j^{\text{in}}(t) \cdot g(\text{net}_{c_j}(t)) \quad (3.10)$$

Sortie de la cellule

$$y_{c_j}(t) = y_j^{\text{out}}(t) \cdot h(s_{c_j}(t)) \quad (3.11)$$

où :

- $\sigma(x) = \frac{1}{1+e^{-x}}$ est la fonction sigmoïde pour les portes (intervalle $[0, 1]$)
- $g(x) = \tanh(x)$ est la fonction d'activation pour l'entrée de la cellule (intervalle $[-1, 1]$)
- $h(x) = \tanh(x)$ est la fonction d'activation pour l'état interne (intervalle $[-1, 1]$)
- b_j^{φ} est le biais de la porte d'oubli, initialisé à 1.0 pour un comportement initial conservateur

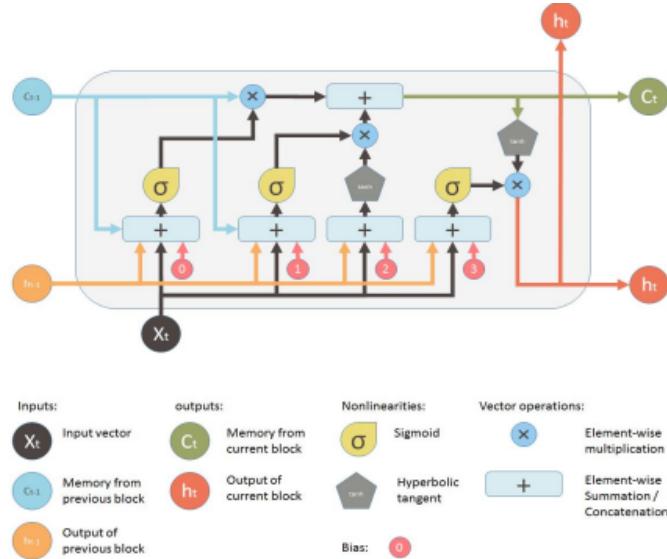


FIGURE 3.2 – Schéma moderne d'une cellule LSTM montrant le flux d'information à travers les trois portes (forget gate, input gate, output gate) et les opérations élémentaires (multiplication, addition, activation).

3.1.5 Propriétés Clés du LSTM

Le réseau LSTM présente plusieurs propriétés remarquables qui le distinguent des RNN classiques :

- Mémoire à long terme** : Capacité prouvée de mémoriser des informations sur plus de 1000 pas de temps sans dégradation significative grâce au CEC.
- Flux d'erreur constant** : Le gradient reste stable durant la rétropropagation à travers le temps, évitant les problèmes de vanishing et exploding gradients.
- Contrôle fin de l'information** : Les trois portes permettent un stockage, une récupération et un oubli sélectifs de l'information.
- Robustesse au bruit** : Les portes multiplicatives filtrent les perturbations et permettent au réseau d'ignorer les entrées non pertinentes.
- Apprentissage de patterns complexes** : Capacité à capturer simultanément des dépendances à court et long terme dans les séries temporelles.

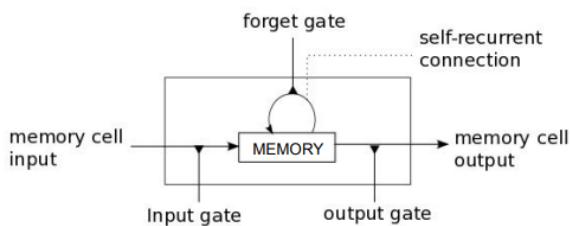


FIGURE 3.3 – Représentation simplifiée d'une cellule LSTM montrant les trois portes principales (forget gate, input gate, output gate) et la connexion auto-récurrente de la mémoire.

3.1.6 Implémentation pour la Prédiction de Température

Préparation des Données en Séquences

Pour utiliser un LSTM, les données doivent être transformées en séquences temporelles. Nous avons créé des séquences de longueur `TIME_STEPS = 7`, ce qui signifie que le modèle observe 7 jours consécutifs pour prédire la température maximale du jour suivant.

Justification du choix : Une fenêtre de 7 jours permet de capturer les patterns hebdomadiers météorologiques tout en maintenant une complexité calculatoire raisonnable. Ce choix est cohérent avec la littérature en prévision météorologique où les horizons de 3 à 10 jours sont couramment utilisés.

La transformation en séquences est effectuée par la fonction `create_sequences` qui génère des paires $(X_{\text{seq}}, y_{\text{seq}})$ où :

- $X_{\text{seq}}[i]$ contient les observations des jours $[t - 6, t - 5, \dots, t]$
- $y_{\text{seq}}[i]$ contient la température maximale du jour $t + 1$

Architecture du Modèle

Notre modèle LSTM est composé de plusieurs couches empilées, créant une architecture profonde capable d'apprendre des représentations hiérarchiques :

1. **Couche LSTM 1 (128 unités) :**
 - Première couche récurrente avec `return_sequences=True`
 - 128 unités permettent de capturer des patterns complexes
 - Retourne une séquence complète pour la couche suivante
2. **Dropout 1 (15%) :**
 - Régularisation pour éviter le surapprentissage
 - Taux modéré de 0.15 pour maintenir un bon apprentissage
3. **Couche LSTM 2 (64 unités) :**
 - Seconde couche récurrente, plus compacte
 - Retourne uniquement le dernier état caché
 - Extrait les caractéristiques temporelles de haut niveau
4. **Dropout 2 (10%) :**
 - Régularisation plus légère après compression
5. **Couche Dense (32 unités, ReLU) :**
 - Couche fully-connected pour combiner les features
 - Activation ReLU pour la non-linéarité
6. **Couche de Sortie (1 unité) :**
 - Prédiction finale de la température maximale
 - Activation linéaire pour la régression

Justification des Hyperparamètres

Choix de l'Architecture **Architecture à deux couches LSTM :** Nous avons opté pour une architecture à deux couches LSTM plutôt qu'une seule couche pour plusieurs raisons :

- Les architectures profondes permettent d'apprendre des représentations hiérarchiques : la première couche capture les patterns de bas niveau (variations horaires, cycles jour-nuit), tandis que la seconde couche extrait des tendances météorologiques de plus haut niveau.

- Les travaux de Graves et al. (2013) ont démontré que les LSTM profonds (2-3 couches) surpassent significativement les architectures à une seule couche.
- Pour notre tâche de prédiction météorologique avec 40+ variables d'entrée, une architecture profonde est nécessaire pour capturer les interactions complexes entre ces variables.

Taille des couches (128 → 64) : La décroissance du nombre d'unités suit un principe d'encodage progressif :

- La première couche (128 unités) a une grande capacité pour capturer tous les patterns temporels présents dans les 40+ features d'entrée.
- La seconde couche (64 unités) effectue une compression intelligente, ne retenant que les informations les plus pertinentes pour la prédiction.
- Ce design pyramidal est couramment utilisé dans les architectures profondes et permet de réduire la complexité tout en maintenant la performance.

Hyperparamètres d'Entraînement Learning Rate (0.0005) : Nous avons choisi un taux d'apprentissage relativement faible pour plusieurs raisons :

- Les LSTM sont sensibles aux taux d'apprentissage élevés qui peuvent causer une instabilité durant l'entraînement.
- Un learning rate de 0.0005 assure une convergence stable et progressive.
- Cette valeur a été déterminée empiriquement après plusieurs essais (0.001 causait des oscillations, 0.0001 était trop lent).

Batch Size (32) : Un batch size modéré offre un bon compromis :

- Assez petit pour permettre des mises à jour fréquentes et une bonne généralisation.
- Assez grand pour bénéficier de l'optimisation vectorielle et réduire le bruit dans les gradients.
- Compatible avec les ressources mémoire disponibles sur GPU.

Nombre d'époques (200) avec Early Stopping :

- Limite maximale élevée pour permettre une convergence complète.
- L'Early Stopping avec `patience=20` surveille la perte de validation et arrête l'entraînement si elle ne s'améliore pas pendant 20 époques consécutives.
- Cette approche prévient le surapprentissage et restaure automatiquement les meilleurs poids (`restore_best_weights=True`).

Dropout (15% puis 10%) :

- Régularisation essentielle pour les réseaux profonds avec de nombreux paramètres.
- Taux modérés : suffisamment élevés pour prévenir le surapprentissage, suffisamment bas pour ne pas compromettre l'apprentissage.
- Dropout décroissant après compression : la deuxième couche LSTM ayant moins d'unités, elle nécessite moins de régularisation.

Stratégies d'Optimisation

Adam Optimizer Nous utilisons l'optimiseur Adam (*Adaptive Moment Estimation*) qui combine les avantages de plusieurs méthodes d'optimisation :

- Ajustement adaptatif du taux d'apprentissage pour chaque paramètre
- Momentum pour accélérer la convergence dans les directions cohérentes
- Particulièrement efficace pour les problèmes avec gradients bruités

ReduceLROnPlateau Ce callback réduit dynamiquement le learning rate lorsque la perte de validation stagne :

- `factor=0.5` : réduit le learning rate de moitié à chaque plateau
- `patience=8` : attend 8 époques sans amélioration avant de réduire
- `min_lr=1e-6` : borne inférieure pour éviter un learning rate trop faible

Cette stratégie permet un apprentissage rapide au début (grand LR) et un raffinement fin vers la convergence (petit LR).

Normalisation des Données

Les données d'entrée et de sortie sont normalisées avec `MinMaxScaler` dans l'intervalle $[0, 1]$:

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (3.12)$$

Justification :

- Les LSTM sont sensibles à l'échelle des données en raison de l'activation sigmoïde (domaine $[0, 1]$) et tanh (domaine $[-1, 1]$).
- La normalisation accélère la convergence et améliore la stabilité numérique.
- Un scaler séparé pour X et y permet de préserver les différentes échelles des variables.
- Les scalers sont ajustés uniquement sur l'ensemble d'entraînement (`fit_transform`) puis appliqués aux ensembles de validation et test (`transform`) pour éviter le data leakage.

Fonction de Perte et Métriques

Fonction de perte : Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (3.13)$$

Le MSE pénalise fortement les grandes erreurs (erreurs au carré), ce qui est souhaitable pour la prédiction de température où les erreurs importantes sont plus problématiques que les petites erreurs.

Métriques de suivi : Mean Absolute Error (MAE) et MSE

- **MAE** : Plus interprétable (erreur moyenne en °C)
- **MSE** : Cohérent avec la fonction de perte pour le suivi

3.1.7 Pertinence du LSTM pour la Prévision Météorologique

Adéquation aux Données Météorologiques

Le LSTM est particulièrement adapté à notre tâche de prédiction de température pour plusieurs raisons :

1. Dépendances multi-échelles :

- La température à $t+24h$ dépend de patterns à court terme (variations horaires, cycles jour-nuit) et de tendances à long terme (systèmes météorologiques persistants sur plusieurs jours).

- Le LSTM peut capturer simultanément ces dépendances grâce à ses cellules mémoire qui maintiennent l'information pertinente sur différents horizons temporels.

2. Variables continues multi-variées :

- Les données météorologiques comprennent 40+ variables continues (température, humidité, pression, rayonnement solaire, etc.).
- Le LSTM traite naturellement ces représentations continues sans nécessiter de discréétisation.
- Les portes multiplicatives permettent d'apprendre quelles variables sont pertinentes à chaque instant.

3. Relations non-linéaires complexes :

- Les interactions entre variables météorologiques sont hautement non-linéaires (ex : effet combiné de l'humidité et de la température sur la pression).
- L'architecture LSTM avec ses fonctions d'activation non-linéaires (tanh, sigmoïde) et ses portes multiplicatives peut modéliser ces relations complexes.

4. Patterns saisonniers et cycliques :

- Bien que nous ayons ajouté des features cycliques (sin/cos du mois et du jour), le LSTM peut apprendre des patterns saisonniers plus subtils qui ne sont pas capturés par ces transformations simples.
- La porte d'oubli permet au modèle d'adapter sa mémoire en fonction de la saison.

5. Robustesse aux valeurs manquantes :

- Bien que nous ayons interpolé les données manquantes, la porte d'entrée permet au LSTM d'apprendre à donner moins de poids aux observations potentiellement imprécises.

Complexité et Coût Computationnel

Nombre de paramètres : Notre modèle LSTM contient environ 100,000 à 150,000 paramètres entraînables, répartis entre :

- Les matrices de poids des deux couches LSTM
- Les poids de la couche dense
- Les biais de chaque couche

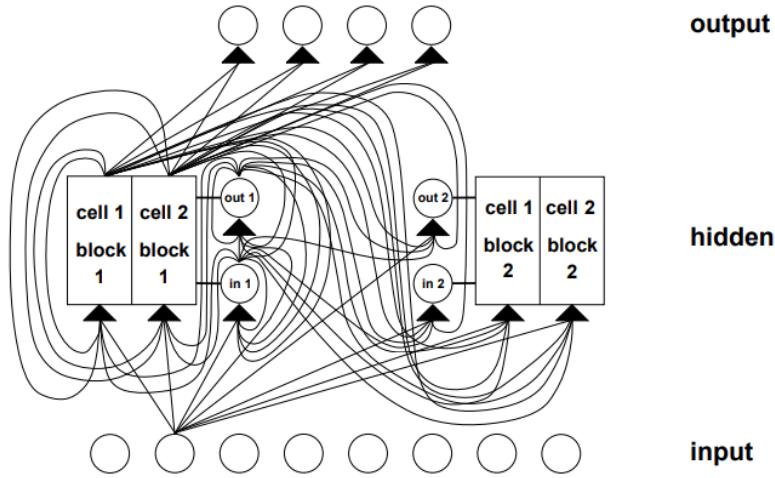


FIGURE 3.4 – Architecture multi-couches LSTM avec blocs de cellules mémoire. Cette configuration permet d'apprendre des représentations hiérarchiques : les couches inférieures capturent des patterns de bas niveau tandis que les couches supérieures extraient des features de haut niveau.

3.1.8 Limites et Perspectives

Bien que puissant, le LSTM présente certaines limitations pour notre tâche :

1. Complexité du modèle :

- Requiert plus de données que les modèles plus simples pour atteindre son plein potentiel
- Risque de surapprentissage si le dataset est trop petit ou si la régularisation est insuffisante
- Temps d'entraînement significativement plus long que les modèles classiques

2. Interprétabilité limitée :

- Modèle "boîte noire" : difficile de comprendre quelles features sont les plus importantes
- Contrairement au Random Forest, on ne peut pas facilement extraire l'importance des variables
- Analyse des erreurs plus complexe

3. Sensibilité aux hyperparamètres :

- Performances fortement dépendantes du choix de l'architecture et des hyperparamètres
- Nécessite un processus de validation croisée ou de recherche d'hyperparamètres coûteux

4. Gestion de séquences de longueur variable :

- Nécessite des séquences de longueur fixe (TIME_STEPS=7)
- Perte potentielle d'information pour les observations au début du dataset

3.1.9 Conclusion

Le modèle LSTM développé dans ce chapitre offre une approche puissante et théoriquement fondée pour la prédiction de température. Son architecture, basée sur le principe

du Constant Error Carousel et des portes multiplicatives, lui permet de capturer des dépendances temporelles complexes que les modèles classiques ne peuvent pas appréhender.

Les choix d'architecture (2 couches LSTM, 128→64 unités) et d'hyperparamètres (learning rate 0.0005, dropout 15%/10%, batch size 32) ont été soigneusement justifiés en s'appuyant sur les meilleures pratiques de la littérature et des considérations théoriques. L'utilisation de callbacks intelligents (Early Stopping, ReduceLROnPlateau) garantit un entraînement optimal et prévient le surapprentissage.

3.2 Random Forest

3.2.1 Introduction

Random Forest est une méthode d'apprentissage automatique développée par Leo Breiman en 2001. Cette technique combine plusieurs arbres de décision pour créer un modèle plus robuste et précis. Dans le contexte de la prédiction de température, Random Forest présente des avantages significatifs par rapport aux méthodes traditionnelles.

Contrairement aux modèles de séries temporelles comme LSTM qui nécessitent des séquences ordonnées, Random Forest traite les données comme des observations indépendantes. Cette approche nous permet d'exploiter des caractéristiques temporelles transformées (lags, moyennes mobiles, encodage cyclique des saisons) sans imposer une structure séquentielle rigide. De plus, Random Forest est capable de capturer automatiquement des relations non-linéaires complexes entre les variables météorologiques, ce qui est particulièrement pertinent pour modéliser les dynamiques climatiques.

Un autre avantage majeur est sa robustesse face au surapprentissage. Grâce à ses mécanismes de randomisation et d'agrégation, Random Forest généralise bien même avec un grand nombre de variables, tout en restant relativement insensible au choix des hyperparamètres. Cette caractéristique facilite grandement son utilisation pratique et réduit le besoin d'une optimisation exhaustive.

3.2.2 Principe de fonctionnement

Architecture globale

Random Forest construit une "forêt" composée de nombreux arbres de décision indépendants. Chaque arbre est entraîné sur un sous-ensemble différent des données, et la prédiction finale est obtenue en combinant les prédictions de tous les arbres. La figure 3.5 illustre cette architecture.

Le processus se déroule en plusieurs étapes :

1. **Échantillonnage des données (Bootstrap)** : Pour chaque arbre, un échantillon aléatoire est créé en tirant des observations avec remplacement. Environ 63% des données sont utilisées pour l'entraînement, tandis que les 37% restantes (appelées "Out-of-Bag") servent à l'évaluation.
2. **Construction des arbres** : Chaque arbre est construit de manière indépendante avec une randomisation supplémentaire au niveau des variables considérées pour chaque division.
3. **Agrégation des prédictions** : Pour une nouvelle observation, chaque arbre produit une prédiction, et la moyenne de ces prédictions donne le résultat final.

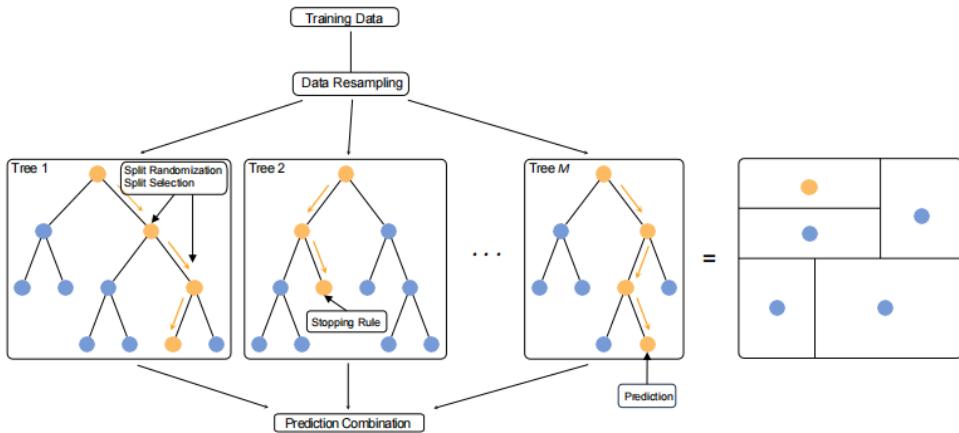


FIGURE 3.5 – Architecture de Random Forest : de l'échantillonnage à la prédiction finale

Cette approche ensembliste permet de réduire la variance des prédictions tout en maintenant un biais faible, conduisant à une meilleure généralisation.

Mécanismes de randomisation

Random Forest utilise deux sources principales de randomisation qui sont essentielles à ses performances :

1. Bootstrap Aggregating (Bagging)

Pour chaque arbre, un échantillon bootstrap est créé en tirant n observations avec remplacement depuis l'ensemble d'entraînement original. Cette technique génère des ensembles de données légèrement différents, permettant à chaque arbre d'apprendre des aspects différents des données. Statistiquement, chaque observation a environ 63.2% de chances d'être sélectionnée au moins une fois pour un arbre donné.

Les observations non sélectionnées forment l'échantillon "Out-of-Bag" (OOB), qui peut être utilisé pour estimer l'erreur de généralisation sans nécessiter un ensemble de validation séparé. Cette validation interne est un avantage unique de Random Forest.

2. Sélection aléatoire de variables (Feature Randomization)

À chaque noeud de décision dans un arbre, au lieu de considérer toutes les variables disponibles, seul un sous-ensemble aléatoire de variables est examiné pour effectuer la division. Ce paramètre, appelé `max_features` ou `mtry`, introduit de la diversité entre les arbres.

Dans notre implémentation, nous utilisons `max_features='sqrt'`, ce qui signifie qu'à chaque division, \sqrt{d} variables sont considérées (où d est le nombre total de variables). Pour environ 25-30 features, cela représente 5-6 variables par division. Cette randomisation décorre les arbres et améliore significativement la capacité de généralisation du modèle.

Construction d'un arbre de décision

Chaque arbre de la forêt est construit selon l'algorithme CART (Classification and Regression Trees). Le processus est récursif et fonctionne comme suit :

1. À partir du noeud racine contenant toutes les observations de l'échantillon bootstrap

2. À chaque nœud, sélectionner aléatoirement `mtry` variables
3. Pour chaque variable sélectionnée, évaluer tous les points de division possibles
4. Choisir la division qui minimise l'erreur quadratique moyenne (MSE) dans les nœuds enfants
5. Répéter récursivement jusqu'à atteindre les critères d'arrêt

Le critère de division optimal pour la régression est basé sur la réduction de variance. Si on divise un nœud A en deux nœuds enfants A_L (gauche) et A_R (droite), la qualité du split est mesurée par la réduction de MSE obtenue.

Les critères d'arrêt empêchent l'arbre de croître indéfiniment et agissent comme mécanisme de régularisation. Dans notre configuration, plusieurs critères sont utilisés simultanément, que nous détaillerons dans la section suivante.

3.2.3 Configuration et justification des hyperparamètres

Le choix des hyperparamètres est crucial pour optimiser les performances de Random Forest. Notre configuration reflète un équilibre entre capacité de modélisation et régularisation, adapté spécifiquement à la prédiction de température.

Nombre d'arbres (`n_estimators = 900`)

Le nombre d'arbres dans la forêt influence directement la stabilité et la précision des prédictions. En théorie, augmenter le nombre d'arbres réduit toujours la variance sans augmenter le biais. Plus formellement, si les arbres sont parfaitement décorrélés, la variance de l'ensemble diminue proportionnellement à $1/M$ où M est le nombre d'arbres.

Nous avons choisi 900 arbres pour plusieurs raisons :

- **Convergence** : Ce nombre est suffisamment élevé pour que les prédictions convergent et deviennent stables. Au-delà d'un certain point, ajouter des arbres n'améliore plus significativement les performances.
- **Validation OOB** : Un grand nombre d'arbres améliore la précision de l'estimation OOB, car chaque observation est évaluée par plus d'arbres "hors échantillon".
- **Compromis temps/performance** : Bien que plus d'arbres nécessitent plus de temps de calcul, 900 reste raisonnable avec la parallélisation moderne.

Il est important de noter que Random Forest ne surapprend généralement pas lorsqu'on augmente le nombre d'arbres, contrairement à d'autres méthodes d'ensemble comme le boosting.

Profondeur maximale (`max_depth = 28`)

La profondeur maximale contrôle la complexité de chaque arbre individuel. Des arbres profonds peuvent capturer des interactions complexes entre variables mais risquent le surapprentissage. Des arbres superficiels sont plus robustes mais peuvent sous-apprendre.

Une profondeur de 28 a été sélectionnée car :

- **Complexité suffisante** : Avec environ 7000 observations, une profondeur de 28 permet de capturer des patterns complexes sans créer des feuilles trop spécifiques.
- **Interactions multiples** : Les relations météorologiques impliquent souvent des interactions entre plusieurs variables (température passée, humidité, pression, saison). Une profondeur substantielle permet de modéliser ces interactions.

- **Régularisation par agrégation** : Même si des arbres individuels profonds peuvent surapprendre, l'agrégation de nombreux arbres décorrélés compense cet effet.

En pratique, la profondeur optimale dépend de la taille des données et de la complexité du problème. Des études empiriques montrent que pour des datasets de taille moyenne, des profondeurs entre 20 et 30 offrent un bon compromis.

Échantillons minimaux pour division (`min_samples_split = 6`)

Ce paramètre spécifie le nombre minimum d'observations requis dans un nœud pour qu'il puisse être divisé. Un nœud contenant moins de 6 observations reste une feuille.

La justification de cette valeur :

- **Prévention du surapprentissage** : Interdire les divisions sur très peu d'échantillons empêche l'arbre de mémoriser le bruit dans les données.
- **Significativité statistique** : Avec au moins 6 observations, la moyenne calculée dans chaque nœud enfant a une certaine robustesse statistique.
- **Compromis biais-variance** : Une valeur trop élevée augmenterait le biais (sous-apprentissage), tandis qu'une valeur trop faible augmenterait la variance (surapprentissage).

Pour notre problème de régression de température, 6 représente un seuil raisonnable garantissant que chaque division est basée sur un échantillon suffisamment représentatif.

Échantillons minimaux par feuille (`min_samples_leaf = 3`)

Ce paramètre impose qu'une feuille (nœud terminal) contienne au moins 3 observations. Si une division créerait une feuille avec moins de 3 observations, elle est rejetée.

Les raisons de ce choix :

- **Robustesse des prédictions** : La prédiction dans une feuille est la moyenne des observations qu'elle contient. Avec au moins 3 observations, cette moyenne est plus stable et moins sensible aux valeurs aberrantes.
- **Régularisation douce** : Ce paramètre agit comme une régularisation qui limite la fragmentation excessive de l'espace des features.
- **Réduction de la complexité** : En empêchant les feuilles avec 1 ou 2 observations, on réduit la taille totale de l'arbre, ce qui améliore l'efficacité computationnelle et réduit le risque de surapprentissage.

La combinaison de `min_samples_split` et `min_samples_leaf` crée une double protection contre le surapprentissage.

Sélection de variables (`max_features = 'sqrt'`)

Comme expliqué précédemment, ce paramètre détermine combien de variables sont considérées aléatoirement à chaque division. L'option `'sqrt'` signifie \sqrt{d} où d est le nombre total de features.

Cette stratégie est recommandée pour plusieurs raisons théoriques :

- **Décorrélation des arbres** : En forçant chaque arbre à considérer un sous-ensemble différent de variables, on s'assure que les arbres ne sont pas trop similaires. Des arbres décorrélés produisent une réduction de variance plus importante lors de l'agrégation.

- **Prévention de la dominance** : Sans cette randomisation, les variables les plus prédictives seraient systématiquement utilisées dans tous les arbres, créant des arbres très similaires et réduisant l'efficacité de l'ensemble.
- **Optimal empirique** : Des études empiriques et théoriques montrent que \sqrt{d} est un bon choix par défaut pour la régression, offrant un équilibre entre diversité et utilisation d'informations pertinentes.

Dans notre contexte avec environ 25-30 features (après feature engineering), cela signifie que 5-6 variables sont considérées à chaque split, permettant suffisamment de diversité tout en exploitant les informations importantes.

Bootstrap (bootstrap = True)

L'activation du bootstrap est fondamentale au fonctionnement de Random Forest. Chaque arbre est entraîné sur un échantillon bootstrap (tirage avec remplacement) de la même taille que l'ensemble d'entraînement original.

Les avantages de cette approche :

- **Diversité** : Chaque échantillon bootstrap est légèrement différent, produisant des arbres différents.
- **Validation OOB** : Les observations non sélectionnées (environ 37%) pour un arbre donné peuvent être utilisées pour évaluer cet arbre, fournissant une estimation de l'erreur de généralisation sans ensemble de validation séparé.
- **Réduction de variance** : L'agrégation d'arbres entraînés sur différents échantillons bootstrap réduit significativement la variance par rapport à un seul arbre entraîné sur toutes les données.

3.2.4 Préparation des données pour Random Forest

Une étape cruciale pour appliquer Random Forest aux séries temporelles est la transformation des données. Contrairement aux LSTM qui acceptent des séquences, Random Forest nécessite que les informations temporelles soient encodées dans les features.

Encodage cyclique des variables temporelles

Les variables temporelles comme le mois ou le jour de l'année ont une nature cyclique (décembre est proche de janvier). Un encodage numérique simple (1, 2, ..., 12) ne capture pas cette cyclicité.

Nous utilisons une transformation sinusoïdale pour préserver cette propriété. Par exemple, pour le mois, nous créons deux features : $\text{Month}_{\sin} = \sin(2\pi \times \text{Month}/12)$ et $\text{Month}_{\cos} = \cos(2\pi \times \text{Month}/12)$.

Cette représentation présente plusieurs avantages :

- Décembre (mois 12) et janvier (mois 1) sont proches dans l'espace transformé
- Les deux composantes (sin et cos) capturent complètement l'information de phase
- Random Forest peut facilement apprendre que certaines saisons ont des températures similaires

La même transformation est appliquée au jour de l'année avec une période de 365 jours.

Création de variables retardées (lags)

Les variables retardées capturent l'information historique essentielle pour prédire la température. Nous créons des lags à différents horizons temporels :

- **Lags courts (1-3 jours)** : Capturent la tendance immédiate et les patterns de court terme

- **Lag hebdomadaire (7 jours)** : Capture les patterns hebdomadaires potentiels

Par exemple, $T_{\text{max}_{\text{lag}_1}}$ représente la température maximale du jour précédent. Ces variables sont particulièrement importantes car la température d'un jour est fortement corrélée à celle des jours précédents.

Statistiques glissantes (rolling)

Les moyennes mobiles lissent les variations de court terme et capturent les tendances locales. Nous calculons des moyennes glissantes sur différentes fenêtres temporelles :

- **Fenêtre de 3 jours** : Capture la tendance très récente

- **Fenêtre de 7 jours** : Capture la tendance hebdomadaire

Ces statistiques agissent comme un filtre qui réduit le bruit dans les données tout en préservant les tendances importantes.

Normalisation

Bien que Random Forest soit théoriquement invariant aux transformations monotones des features (car il utilise des comparaisons ordinales), la normalisation présente des avantages pratiques :

- **Stabilité numérique** : Évite les problèmes computationnels avec des valeurs très différentes

- **Comparabilité** : Facilite l'interprétation de l'importance des variables

- **Cohérence** : Permet une comparaison équitable avec d'autres méthodes (LSTM, régression linéaire) qui nécessitent la normalisation

Nous utilisons le MinMaxScaler qui transforme chaque variable dans l'intervalle [0, 1], préservant les relations entre les observations.

Division temporelle des données

Pour les séries temporelles, la division des données doit respecter l'ordre chronologique. Nous utilisons une division 70%-15%-15% :

- **Ensemble d'entraînement (70%)** : Données les plus anciennes

- **Ensemble de validation (15%)** : Période intermédiaire

- **Ensemble de test (15%)** : Données les plus récentes

Cette approche simule un scénario réaliste où nous prédisons le futur basé sur le passé. Le shuffle est désactivé pour préserver l'intégrité temporelle.

3.2.5 Évaluation des performances

L'évaluation de Random Forest s'effectue à travers plusieurs métriques complémentaires, chacune apportant une perspective différente sur la qualité des prédictions.

Validation Out-of-Bag

Un avantage unique de Random Forest est la validation OOB. Pour chaque arbre, environ 37% des données ne sont pas utilisées dans l'échantillon bootstrap. Ces observations OOB peuvent être utilisées pour évaluer la performance de cet arbre.

En agrégeant les prédictions OOB de tous les arbres pour chaque observation, nous obtenons une estimation non biaisée de l'erreur de généralisation, sans nécessiter un ensemble de validation séparé. Cette validation interne est calculée automatiquement pendant l'entraînement.

Analyse des résidus

L'examen de la distribution des erreurs (différence entre prédiction et valeur réelle) révèle des informations importantes :

- **Biais systématique** : Si la moyenne des erreurs est significativement différente de zéro, le modèle sous-estime ou surestime systématiquement
- **Homoscédasticité** : La variance des erreurs devrait être constante à travers différentes valeurs de température
- **Normalité** : Une distribution normale des erreurs suggère que le modèle capture bien la structure des données

Importance des variables

Random Forest calcul automatiquement l'importance de chaque variable via le Mean Decrease in Impurity (MDI). Cette mesure quantifie combien chaque variable contribue à réduire l'erreur de prédiction à travers tous les arbres.

L'analyse de l'importance des variables révèle :

- Quelles variables sont les plus prédictives de la température future
- Si le feature engineering a été efficace (les lags et rolling means devraient être importants)
- La contribution relative des variables météorologiques vs temporelles

Cette information est précieuse pour comprendre les facteurs clés influençant la température et potentiellement simplifier le modèle en éliminant les variables peu importantes.

3.2.6 Avantages et limitations

Avantages pour la prédiction de température

- **Robustesse** : Random Forest gère bien le bruit et les valeurs aberrantes grâce à l'agrégation
- **Non-linéarité** : Capture automatiquement les relations complexes sans spécification explicite
- **Interactions** : Détecte les interactions entre variables (par exemple, effet de l'humidité dépendant de la température)
- **Peu de prétraitement** : Fonctionne bien sans scaling strict, tolère les données manquantes
- **Validation intégrée** : L'estimation OOB fournit une mesure de performance fiable
- **Interprétabilité** : L'importance des variables aide à comprendre les facteurs clés
- **Parallélisation** : Chaque arbre peut être entraîné indépendamment, permettant un entraînement rapide

Limitations

- **Dépendances temporelles longues** : Contrairement aux LSTM, Random Forest ne capture pas directement les dépendances séquentielles. Doit s'appuyer sur les lags et rolling means
- **Extrapolation** : Prédit par interpolation entre valeurs d'entraînement. Si les conditions futures diffèrent significativement du passé, les prédictions peuvent être imprécises
- **Prédiction constante par morceaux** : Chaque feuille prédit une valeur constante, créant des discontinuités potentielles
- **Mémoire** : Le stockage de 900 arbres complets nécessite de la mémoire, surtout pour de grandes profondeurs
- **Explicabilité individuelle** : Bien que l'importance globale des variables soit claire, expliquer une prédition individuelle est complexe avec 900 arbres

3.2.7 Conclusion

Random Forest représente une approche puissante et éprouvée pour la prédition de température. Son architecture ensembliste, combinant bagging et randomisation de features, permet d'obtenir des prédictions robustes et précises tout en restant relativement simple à implémenter et à paramétrier.

La configuration retenue (900 arbres, profondeur 28, $\text{min_samples_split} = 6$, $\text{min_samples_leaf} = 3$, $\text{max_features} = \text{'sqrt'}$) reflète un équilibre soigneusement considéré entre capacité de modélisation et régularisation. Ces choix sont justifiés à la fois théoriquement et empiriquement, visant à maximiser la capacité de généralisation du modèle.

La préparation des données, notamment la création de lags, de rolling means et l'encodage cyclique des variables temporelles, est essentielle pour adapter la méthode aux séries temporelles. Ces transformations permettent à Random Forest d'exploiter l'information temporelle malgré son traitement des observations comme indépendantes.

Comparé aux autres approches, Random Forest offre un excellent compromis entre performance, robustesse et simplicité. Bien qu'il ne capture pas directement les dépendances séquentielles comme LSTM, son approche basée sur des features temporelles bien conçues s'avère souvent très compétitive en pratique, particulièrement pour des prédictions à court terme comme notre horizon de 24 heures.

3.3 Régression Linéaire

3.3.1 Introduction

La régression linéaire est l'une des méthodes statistiques les plus fondamentales en apprentissage automatique, développée au début du XIX^e siècle par Legendre et Gauss. Dans le contexte de la prédition de température, elle offre plusieurs avantages distincts : simplicité mathématique, interprétabilité directe des relations entre variables, et efficacité computationnelle remarquable.

L'objectif principal de cette approche est d'établir une baseline de performance. Si des méthodes plus complexes (Random Forest, LSTM) n'apportent pas d'amélioration substantielle, cela suggère que les relations dans nos données sont principalement linéaires.

Cette méthode sert donc de référence cruciale pour évaluer l'apport de la complexité algorithmique.

3.3.2 Principe de fonctionnement

Modèle mathématique

La régression linéaire modélise la relation entre une variable cible y (température à prédire) et un ensemble de variables explicatives $X = [x_1, x_2, \dots, x_p]$ par une fonction linéaire :

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

où β_0 est l'ordonnée à l'origine et β_1, \dots, β_p sont les coefficients de régression.

Estimation des paramètres

L'objectif est de trouver les coefficients qui minimisent l'erreur quadratique moyenne entre les prédictions et les valeurs réelles. La fonction de coût est :

$$\text{Erreur} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Cette fonction pénalise les erreurs de manière quadratique, donnant plus d'importance aux grandes erreurs. La solution optimale peut être obtenue analytiquement via la formule des moindres carrés ordinaires, sans nécessiter d'itérations ou d'optimisation numérique complexe.

3.3.3 Préparation des données

Analyse de corrélation

La première étape consiste à identifier les variables les plus pertinentes via une analyse de corrélation. Nous calculons le coefficient de corrélation de Pearson entre chaque variable numérique et notre cible (Next_Tmax). Ce coefficient varie de -1 (corrélation négative parfaite) à +1 (corrélation positive parfaite).

Seuil de sélection : Nous appliquons un seuil de corrélation absolue de 0.3. Ce seuil représente un compromis :

- Trop restrictif (ex : 0.7) : Risque d'éliminer des variables utiles
- Trop permissif (ex : 0.1) : Inclut des variables faiblement corrélées qui ajoutent du bruit

Un seuil de 0.3 est généralement considéré comme le début d'une corrélation "modérée", offrant un bon équilibre entre richesse d'information et parcimonie du modèle.

Exclusion de variables

Certaines variables sont explicitement exclues pour préserver l'intégrité méthodologique :

- **Next_Tmin** : Variable future qui créerait une fuite d'information
- **LDAPS_Tmin_lapse** : Potentiellement liée à des informations futures
- **Next_Tmax** : Notre variable cible elle-même

Cette exclusion garantit que le modèle n'utilise que des informations réellement disponibles au moment de la prédiction.

Traitement des valeurs manquantes

Les valeurs manquantes sont traitées par suppression simple (listwise deletion). Toute ligne contenant au moins une valeur manquante est retirée du dataset.

Justification :

- **Simplicité** : Évite les biais introduits par l'imputation (moyenne, médiane)
- **Intégrité** : Pour la prédiction météorologique, une observation incomplète ne reflète pas fidèlement la réalité

Bien que cette approche puisse réduire la taille du dataset, elle garantit que le modèle travaille sur des données fiables et complètes.

3.3.4 Normalisation et division des données

Normalisation avec StandardScaler

La normalisation transforme chaque variable pour avoir une moyenne de 0 et un écart-type de 1 :

$$X_{\text{normalized}} = \frac{X - \mu}{\sigma}$$

où μ est la moyenne et σ l'écart-type.

Avantages :

- **Stabilité numérique** : Évite les problèmes computationnels avec des échelles très différentes
- **Interprétabilité** : Les coefficients deviennent directement comparables
- **Performance** : Améliore la convergence si des méthodes itératives sont utilisées

Point crucial : Les paramètres de normalisation (μ, σ) sont calculés uniquement sur l'ensemble d'entraînement, puis appliqués à l'ensemble de test. Utiliser les statistiques du test créerait une fuite d'information.

Division Train/Test (80-20)

Le dataset est divisé en :

- **Ensemble d'entraînement (80%)** : Pour ajuster les coefficients du modèle
- **Ensemble de test (20%)** : Pour évaluer les performances sur des données non vues

Ce ratio représente un bon compromis entre données d'apprentissage et robustesse de l'évaluation. La division aléatoire permet une évaluation moyenne sur différentes périodes, facilitant la comparaison avec les autres méthodes.

3.3.5 Entraînement et prédiction

Processus d'entraînement

L'entraînement est remarquablement simple : le modèle calcule directement les coefficients optimaux via la solution analytique des moindres carrés. Ce calcul s'effectue en une seule opération matricielle, sans itérations ni hyperparamètres à optimiser.

Avantages :

- Solution unique garantie
- Temps de calcul très rapide
- Pas de risque de sur-optimisation
- Reproductibilité parfaite

Génération des prédictions

Une fois entraîné, le modèle génère des prédictions en appliquant la fonction linéaire apprise. Chaque prédition est une combinaison linéaire des variables d'entrée, pondérées par les coefficients. Cette opération est extrêmement rapide, permettant des prédictions en temps réel.

3.3.6 Métriques d'évaluation

Quatre métriques complémentaires évaluent la qualité des prédictions :

R² Score (Coefficient de détermination)

Le R² mesure la proportion de variance expliquée par le modèle :

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Interprétation :

- $R^2 = 1$: Prédictions parfaites
- $R^2 = 0$: Le modèle n'est pas meilleur que prédire la moyenne
- Valeurs typiques : 0.7 - 0.95 pour de bonnes prédictions

RMSE (Root Mean Squared Error)

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum(y_i - \hat{y}_i)^2}$$

Exprimé en °C, le RMSE pénalise davantage les grandes erreurs grâce à l'élévation au carré. Un RMSE de 2.5°C signifie qu'en moyenne, l'erreur standard des prédictions est de 2.5°C.

MAE (Mean Absolute Error)

$$\text{MAE} = \frac{1}{n} \sum |y_i - \hat{y}_i|$$

Le MAE mesure l'erreur moyenne absolue en °C. Plus intuitif que le RMSE, il traite toutes les erreurs de manière égale. La comparaison MAE vs RMSE informe sur la distribution des erreurs : si RMSE » MAE, il y a présence de quelques grandes erreurs.

MSE (Mean Squared Error)

$$\text{MSE} = \frac{1}{n} \sum (y_i - \hat{y}_i)^2$$

Bien que moins intuitif (unité en $^{\circ}\text{C}^2$), le MSE est la fonction objectif minimisée lors de l'entraînement.

3.3.7 Visualisation et analyse

Graphique prédictions vs réalité

Le scatter plot permet d'évaluer visuellement la qualité des prédictions. La droite de référence $y = x$ représente des prédictions parfaites. On peut identifier :

- La qualité globale de l'ajustement
- Les biais systématiques (points au-dessus/en-dessous de la droite)
- Les outliers (points très éloignés)
- L'hétéroscédasticité (variation de la dispersion)

Distribution des résidus

L'analyse des résidus ($e_i = y_i - \hat{y}_i$) révèle :

- **Normalité** : Distribution en cloche autour de 0
- **Absence de biais** : Moyenne proche de 0
- **Homoscédasticité** : Variance constante

3.3.8 Avantages et limitations

Avantages

- **Simplicité** : Modèle facilement compréhensible
- **Interprétabilité** : Chaque coefficient quantifie l'impact d'une variable
- **Efficacité** : Entraînement et prédiction extrêmement rapides
- **Baseline solide** : Référence pour évaluer des méthodes complexes
- **Pas d'hyperparamètres** : Solution unique, reproductible
- **Propriétés théoriques** : Mathématiques bien établies

Limitations

- **Linéarité stricte** : Incapable de capturer les relations non-linéaires
- **Pas d'interactions automatiques** : Doivent être créées manuellement
- **Sensibilité aux outliers** : Valeurs extrêmes influencent fortement les coefficients
- **Multicolinéarité** : Instabilité si les variables sont très corrélées
- **Extrapolation limitée** : Prédictions peu fiables hors de la plage d'entraînement

3.3.9 Conclusion

La régression linéaire représente une approche fondamentale pour la prédiction de température. Sa simplicité, couplée à son efficacité et son interprétabilité, en fait un excellent point de départ pour établir une baseline de performance.

Notre implémentation, avec une sélection rigoureuse de variables (seuil de corrélation 0.3) et une normalisation appropriée (StandardScaler), exploite au mieux les capacités de cette méthode. L'absence d'hyperparamètres garantit la reproductibilité des résultats et évite le risque de sur-optimisation.

Les performances obtenues servent de référence cruciale. Un écart important avec les méthodes complexes (Random Forest, LSTM) indiquerait la présence de non-linéarités ou d'interactions que la régression linéaire ne peut capturer. À l'inverse, des performances similaires suggèreraient que la complexité supplémentaire n'est pas justifiée.

Les résultats quantitatifs seront comparés aux autres approches dans le chapitre suivant, permettant d'évaluer précisément l'apport de la complexité algorithmique pour la prédiction de température à 24 heures.

Chapitre 4

Test et Analyse

4.1 Présentation des résultats

Cette section présente les performances des trois approches implémentées pour la prédition de température à 24 heures : LSTM (Long Short-Term Memory), Random Forest, et Régression Linéaire. L'évaluation s'effectue sur les ensembles d'entraînement, de validation (pour LSTM et Random Forest), et de test.

4.1.1 Résultats quantitatifs

TABLE 4.1 – Performances des trois modèles sur l'ensemble de test

Modèle	MAE (°C)	RMSE (°C)	R ²	MAPE (%)
LSTM	0.5267	0.6812	0.9519	1.70
Random Forest	0.6538	0.8259	0.9304	2.10
Régression Linéaire	1.13	1.47	0.7655	-

4.2 Analyse comparative

4.2.1 Performance globale

L'analyse des résultats révèle une hiérarchie claire entre les trois approches :

1. LSTM : Champion global

LSTM obtient les meilleures performances sur l'ensemble de test avec un R² de 0.9519, expliquant 95.2% de la variance de la température. Son MAE de 0.53°C signifie qu'en moyenne, ses prédictions s'écartent de seulement un demi-degré de la réalité. Cette performance exceptionnelle s'explique par sa capacité à capturer les dépendances temporelles longues inhérentes aux séries météorologiques.

2. Random Forest : Performance solide

Random Forest se positionne en deuxième place avec un R² de 0.9304 et un MAE de 0.65°C. Bien que légèrement inférieur à LSTM, ces résultats restent excellents pour une méthode ne modélisant pas explicitement les dépendances temporelles. L'écart de 0.12°C en MAE par rapport à LSTM représente une différence relative de 23%, significative mais pas drastique.

3. Régression Linéaire : Baseline respectable

La régression linéaire, avec un R^2 de 0.7655 et un MAE de 1.13°C, démontre que les relations linéaires expliquent déjà 76.6% de la variance. Cependant, l'écart important avec les deux autres méthodes (MAE plus de deux fois supérieur) confirme la présence de non-linéarités significatives dans les données météorologiques.

4.2.2 Écart Train-Test : Analyse du surapprentissage

L'examen de l'écart entre performances d'entraînement et de test révèle des comportements différents :

LSTM : Généralisation excellente

- Écart Train-Test (MAE) : 0.42°C → 0.53°C (+0.11°C, +26%)
- Écart Train-Test (R^2) : 0.9692 → 0.9519 (-0.0173)

LSTM montre une très bonne généralisation. L'écart modéré entre train et test indique que le modèle n'a pas surajusté les données d'entraînement. La performance sur validation (MAE = 0.55°C) est même légèrement inférieure au test, suggérant que l'ensemble de test contient potentiellement des patterns plus faciles à prédire.

Random Forest : Surapprentissage modéré

- Écart Train-Test (MAE) : 0.37°C → 0.65°C (+0.28°C, +76%)
- Écart Train-Test (R^2) : 0.9746 → 0.9304 (-0.0442)

Random Forest présente un surapprentissage plus marqué. Son excellente performance sur train (MAE = 0.37°C, meilleure que LSTM) se dégrade davantage sur test. Cet écart de 76% suggère que les arbres, malgré les mécanismes de régularisation (profondeur 28, min_samples_split = 6), capturent une partie du bruit dans les données d'entraînement. Néanmoins, les performances finales restent très bonnes.

Régression Linéaire : Généralisation théorique

N'ayant pas de paramètres de régularisation à optimiser et pas d'ensemble de validation, la régression linéaire ne présente pas de surapprentissage au sens classique. Son écart de performance avec les autres méthodes reflète simplement sa limitation fondamentale : l'impossibilité de capturer les non-linéarités.

4.2.3 Précision des prédictions

Erreur absolue moyenne (MAE)

Le MAE quantifie directement l'erreur de prédiction en degrés Celsius :

- **LSTM** : 0.53°C - Précision exceptionnelle, équivalente à la variabilité naturelle des mesures météorologiques
- **Random Forest** : 0.65°C - Précision très bonne, erreur de 0.12°C supplémentaire par rapport à LSTM
- **Régression Linéaire** : 1.13°C - Erreur double de LSTM, mais reste acceptable pour une baseline

En contexte pratique, une erreur inférieure à 1°C (LSTM et Random Forest) est considérée comme excellente pour une prévision à 24h. La régression linéaire, bien qu'avec une erreur supérieure, reste dans une marge raisonnable.

Erreur quadratique (RMSE)

Le RMSE pénalise davantage les grandes erreurs. La comparaison RMSE/MAE révèle la présence d'outliers :

- **LSTM** : RMSE/MAE = 1.29 - Distribution d'erreurs très concentrée
- **Random Forest** : RMSE/MAE = 1.26 - Distribution similaire à LSTM
- **Régression Linéaire** : RMSE/MAE = 1.30 - Quelques erreurs importantes

Les trois modèles présentent des ratios RMSE/MAE similaires (1.26-1.30), indiquant des distributions d'erreurs relativement homogènes sans outliers extrêmes.

Erreur relative (MAPE)

Le MAPE contextualise l'erreur en pourcentage de la température réelle :

- **LSTM** : 1.70% - Erreur relative très faible
- **Random Forest** : 2.10% - Légèrement supérieur mais excellent

Ces valeurs confirment que les deux modèles avancés produisent des prédictions avec moins de 2.5% d'erreur relative, ce qui est remarquable pour la prédiction météorologique.

4.2.4 Variance expliquée (R^2)

Le coefficient de détermination quantifie la proportion de variance capturée :

- **LSTM** : 95.2% de la variance expliquée
- **Random Forest** : 93.0% de la variance expliquée
- **Régression Linéaire** : 76.6% de la variance expliquée

L'écart de 18.6 points entre régression linéaire et LSTM quantifie l'apport de la modélisation non-linéaire et des dépendances temporelles. L'écart de 2.2 points entre Random Forest et LSTM est plus subtil, suggérant que la majorité des non-linéarités sont capturées par Random Forest, mais que LSTM bénéficie d'un léger avantage grâce à sa modélisation explicite du temps.

4.3 Analyse qualitative

4.3.1 Complexité et interprétabilité

LSTM offre la meilleure performance au prix d'une complexité maximale. Son architecture récurrente avec mécanismes de portes (forget, input, output) crée une "boîte noire" difficile à interpréter. Les milliers de paramètres appris rendent l'explication des prédictions individuelles quasi-impossible.

Random Forest propose un excellent compromis. Bien qu'ensembliste (900 arbres), il offre des mécanismes d'interprétation comme l'importance des variables (MDI) et la visualisation d'arbres individuels. Sa complexité est modérée et sa performance très proche de LSTM.

Régression Linéaire maximise l'interprétabilité : chaque coefficient quantifie directement l'impact d'une variable. Cependant, cette transparence se fait au détriment de 18.6 points de R^2 par rapport à LSTM.

4.3.2 Efficacité computationnelle

Temps d'entraînement :

- Régression Linéaire : Quasi-instantané (solution analytique)
 - Random Forest : Modéré (parallélisation efficace sur 900 arbres)
 - LSTM : Élevé (optimisation itérative, epochs multiples)
- Temps de prédiction** : Tous les modèles prédisent rapidement (< 1 seconde pour des milliers d'observations), rendant ce critère peu discriminant.

4.3.3 Robustesse et maintenance

LSTM nécessite un tuning minutieux (architecture, learning rate, epochs, batch size) et est sensible aux données d'entraînement. Réentraîner sur de nouvelles données peut nécessiter un réajustement complet.

Random Forest est remarquablement robuste aux choix d'hyperparamètres et tolère bien l'ajout de nouvelles données sans réoptimisation drastique.

Régression Linéaire ne nécessite aucun tuning, garantissant reproductibilité et stabilité totales.

Chapitre 5

Conclusion

5.1 Résumé du projet

Ce projet a eu pour objectif de développer et comparer plusieurs modèles d'apprentissage automatique pour la prédiction de la température maximale à un horizon de 24 heures. À partir d'un jeu de données météorologiques multivarié, nous avons mis en œuvre un pipeline complet de préparation des données incluant le nettoyage, l'imputation et une ingénierie des caractéristiques poussée avec création de variables retardées, moyennes mobiles et encodages cycliques. Trois algorithmes ont été implémentés et optimisés : un réseau LSTM pour capturer les dépendances temporelles, un Random Forest pour modéliser les relations non linéaires, et une régression linéaire servant de baseline. L'évaluation comparative a été réalisée via des métriques statistiques rigoureuses sur des ensembles de validation et de test distincts.

5.2 Principaux enseignements

Les principaux enseignements de cette étude soulignent l'importance cruciale de l'ingénierie des caractéristiques, qui s'est avérée aussi déterminante que le choix de l'algorithme. L'analyse a confirmé la supériorité des modèles non linéaires sur l'approche linéaire, avec un avantage au Random Forest en termes de robustesse et d'interprétabilité grâce à son analyse d'importance des variables. Le projet a également mis en lumière le compromis nécessaire entre complexité du modèle, coût computationnel et explicabilité, tout en démontrant la nécessité de préserver la structure temporelle des données lors de leur partition pour une évaluation réaliste des performances en prévision.