Fiche de Synthèse: De la Régression Linéaire aux Réseaux de Neurones : Optimisation de la Prédiction des Performances des Éoliennes

Module: Deep Learning et Applications

Option Sciences de Données et Digitalisation

École Centrale de Casablanca

Année universitaire 2024/2025

Binôme : Adam LOZI
Hamza EL OTMANI

16 mars 2025

Résumé

Ce projet vise à prédire la puissance active produite par des éoliennes à partir de données opérationnelles. Nous comparons les performances d'un réseau de neurones avec des méthodes classiques de machine learning.

Table des matières

Table des figures

Listings

1 Introduction

1.1 Contexte

Dans un contexte de transition énergétique mondiale, les énergies renouvelables jouent un rôle central pour répondre aux défis climatiques. L'éolien, en particulier, constitue un pilier stratégique pour ENGIE.

Ce projet s'inscrit dans le cadre d'un challenge proposé par la chaire « Science des données » du Collège de France, visant à optimiser la production des parcs éoliens existants. L'enjeu opérationnel est clair : détecter les écarts anormaux entre la production théorique et réelle des turbines, un levier essentiel pour améliorer la maintenance prédictive et maximiser le rendement énergétique.

1.2 Problématique

L'objectif technique consiste à prédire la **puissance active instantanée** (en MW) de quatre éoliennes à partir de 58 paramètres opérationnels internes, en excluant délibérément les mesures de vitesse du vent jugées peu fiables. Les données disponibles incluent :

— Des séries temporelles de températures (nacelle, générateur, rotor)
— Des mesures électriques (tension, fréquence du réseau)
— Des dynamiques mécaniques (vitesses de rotation, angles d'orientation)

La principale difficulté réside dans la modélisation des relations complexes entre ces signaux multivariés (échantillonnés toutes les 10 minutes) et la puissance de sortie, avec une métrique d'évaluation exigeante : l'**Erreur Absolue Moyenne** (MAE). Ce choix impose un équilibre entre précision des prédictions et interprétabilité des modèles.

2 Données et Préparation

2.1 Sources de données

Les données proviennent de deux fichiers structurés fournis par ENGIE :

— **engie_X.csv** : Contient 58 variables explicatives mesurées sur 4 éoliennes (WT1 à WT4) avec une résolution temporelle de 10 minutes. Les catégories clés incluent :
  — *Températures* : 10 capteurs (ex : `Gearbox_bearing_1_temperature_avg`)
  — *Électrique* : Tension (`Grid_voltage_avg`), fréquence (`Grid_frequency_avg`)
  — *Cinématique* : Vitesses de rotation (`Rotor_speed_std`), angles (`Pitch_angle_max`)
— **engie_Y.csv** : Variable cible unique `TARGET` représentant la puissance active normalisée (échelle 0-1).

Contrainte majeure : Aucune donnée de vitesse du vent n'est incluse, conformément au cahier des charges.

2.2 Exploration des dimensions et des types de données

2.2.1 Dimensions

Les données utilisées dans ce projet sont composées de 617386 observations et 79 colonnes, dont 78 variables explicatives et 1 variable cible.

- **Nombre total d'individus (observations)** : 617386
- **Nombre total de variables** : 79
- **Nombre de variables explicatives** : 78
- **Variable cible** : TARGET (puissance active des éoliennes)

2.2.2 Types des variables

Les données sont constituées de plusieurs types de variables :

- **Variables numériques** : La majorité des variables sont de type *float64*, représentant des mesures physiques comme la température des composants, l'angle de la nacelle, la fréquence du réseau, etc.
- **Variables catégorielles** : Une seule variable est de type *object* : *MAC_CODE*, identifiant chaque éolienne.
- **Identifiants et variables temporelles** :
  - *ID* : Entier unique identifiant chaque ligne (int64).
  - *Date_time* : Variable temporelle représentant les timestamps toutes les 10 minutes (float64).

2.2.3 Statistiques descriptives

Une analyse statistique des variables numériques principales est réalisée pour mieux comprendre la distribution des données :

- **Moyenne et écart-type** : La variable *Pitch_angle* a une moyenne de 13.01 et un écart-type de 27.23, indiquant une grande variabilité.
- **Valeurs extrêmes** : Certaines variables présentent des valeurs minimales et maximales très éloignées, comme *Pitch_angle_min* avec un minimum de -179.57 et un maximum de 360.00, nécessitant une normalisation ou un traitement des outliers.
- **Distribution des données** :
  - Le premier quartile (Q1) et le troisième quartile (Q3) permettent d'identifier les valeurs médianes et les écarts potentiels.
  - La variable *Absolute_wind_direction_c* varie de 0 à 360 degrés, ce qui peut nécessiter une transformation circulaire pour une meilleure modélisation.

2.3 Analyse de la répartition de `TARGET` par turbine

Nous utilisons un boxplot pour visualiser la distribution de la puissance active en fonction de chaque turbine.

2.3.1 Observations

- **Dispersion des valeurs :** La distribution est similaire pour toutes les turbines, avec une médiane relativement basse et une large dispersion. La majorité des valeurs se concentrent entre 0 et 500, avec des valeurs extrêmes atteignant plus de 2000.
- **Présence de valeurs aberrantes :** De nombreuses valeurs au-delà des moustaches du boxplot indiquent des outliers, suggérant des fluctuations importantes de production.

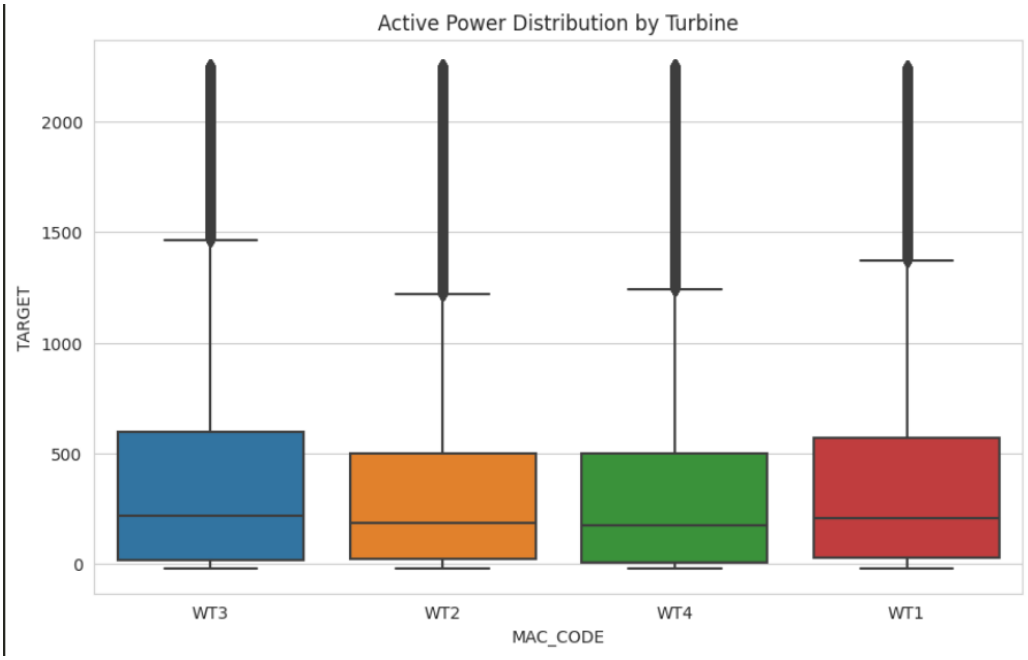

FIGURE 1 – Distribution de la puissance active par turbine

— **Comparaison entre turbines :** Aucune différence marquée entre les turbines, ce qui peut indiquer un comportement homogène en termes de puissance générée. WT1 et WT3 semblent avoir des médianes légèrement plus élevées que WT2 et WT4.

2.3.2 Interprétations et implications

— L'analyse des valeurs extrêmes pourrait permettre d'identifier les conditions entraînant des pics de production.
— Une modélisation des outliers pourrait être utile pour détecter des anomalies ou optimiser la maintenance des turbines.

3 Prétraitement des Données

3.1 Traitement des valeurs manquantes

Nous avons identifié les colonnes contenant un grand nombre de valeurs manquantes. Un seuil a été défini : si une colonne dépasse ce seuil de valeurs manquantes, elle est supprimée.

Pourquoi ?

— Une colonne avec trop de valeurs manquantes peut introduire du bruit et fausser l'apprentissage du modèle.
— Si la majorité des valeurs sont absentes, l'imputation risque d'être imprécise.

3.2 Imputation des valeurs manquantes restantes

Pour les variables numériques, les valeurs manquantes ont été remplacées par la **médiane** plutôt que la moyenne.

Pourquoi ?

— La médiane est plus robuste aux outliers (valeurs extrêmes) qui pourraient fausser l'apprentissage du modèle.
— La moyenne peut être biaisée si certaines valeurs sont anormalement grandes ou petites.

3.3 Encodage des variables catégorielles

'MAC_CODE', qui est une variable catégorielle, a été transformée en variables binaires à l'aide du **one-hot encoding**.

Pourquoi ?

— Les modèles de machine learning ne comprennent pas les variables textuelles.

— Le one-hot encoding permet d'éviter l'ordre arbitraire introduit par d'autres techniques comme le label encoding.

3.4 Division temporelle "time aware" du dataset (train/test)

Nous avons trié les données par 'Date_time' avant de les séparer en **train (80%)** et **test (20%)**.

Pourquoi ?

— Cela préserve la structure temporelle des données, évitant ainsi **le data leakage**.

— Utile pour des modèles séquentiels comme **LSTM**, qui apprennent mieux sur des séries temporelles triées.

3.5 Visualisation des corrélations avec 'TARGET'

Nous avons affiché un **barplot** des **features les moins corrélées** avec 'TARGET'.

Pourquoi ?

— Permet de mieux comprendre quelles variables ont un impact sur la puissance active.

— Éliminer les variables non pertinentes permet d'améliorer la performance et la rapidité du modèle.

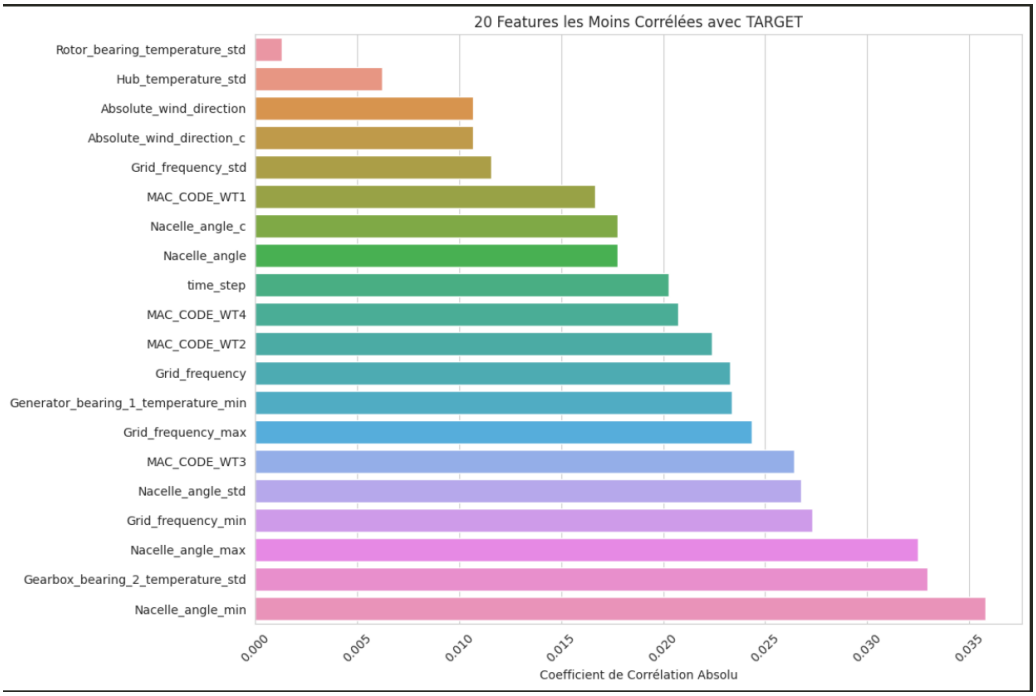

FIGURE 2 – Corrélations avec 'TARGET'

3.6 Sélection des variables pertinentes

Nous avons supprimé les **caractéristiques faiblement corrélées** avec la cible 'TARGET', sauf 'time_step' qui est préservé pour tout entrainement time aware.

Pourquoi ?

— Éliminer les variables inutiles réduit le bruit et accélère l'apprentissage du modèle.

— 'time_step' est important car il aide à capturer la dépendance temporelle des données.

3.7 Mise à l'échelle des données

Nous avons utilisé **RobustScaler** pour normaliser les données.

Pourquoi ?

— Ce scaler réduit l'impact des outliers **contrairement à StandardScaler**.

— Il transforme les données en les centrant autour de la médiane et en les divisant par l'**écart interquartile**.

En résumé, ce prétraitement optimise le dataset en supprimant le bruit et en préparant les données pour un apprentissage efficace, notamment pour les modèles LSTM et autres modèles temporels.

4 Modèles et Résultats

4.1 Introduction

Nous visons à développer un modèle prédictif performant pour estimer la **puissance active** en fonction de diverses variables explicatives. L'objectif est de capter efficacement les tendances sous-jacentes des données afin d'obtenir des prédictions précises.

Nous avons exploré plusieurs approches de modélisation. Les modèles testés sont les suivants :

- **Régression Linéaire** : modèle de base servant de référence dans notre étude.
- **Random Forest** : méthode d'ensemble robuste basée sur des arbres de décision.
- **XGBoost** : algorithme de boosting optimisé pour la performance.
- **Réseau de Neurones (MLP)** : modèle exploitant des relations complexes entre les variables.
- **LSTM (Long Short-Term Memory)** : réseau neuronal récurrent adapté aux séries temporelles (donc qu'on peut adapter à notre jeu de données).

Pour comparer ces modèles, nous avons choisi la **Mean Absolute Error (MAE)** comme métrique principale, metrique spécifiée dans le challenge de Data Science dans lequel figurent ces données, garantissant une évaluation intuitive des erreurs de prédiction. Nous analyserons également l'efficacité computationnelle en comparant les **temps d'entraînement**.
Dans la suite de cette section nous allons aborder les points suivants :

1. Présentation détaillée des modèles et de leurs architectures.
2. Analyse des performances en termes de précision et d'efficacité computationnelle.
3. Discussion et conclusions sur les résultats obtenus.

4.2 Modèle de Régression Linéaire

4.2.1 Hypothèses Simplificatrices et Intérêt

La **régression linéaire** est un modèle prédictif fondé sur une relation linéaire entre les variables explicatives et la variable cible. Elle repose sur plusieurs hypothèses simplificatrices :

- **Linéarité** : la relation entre les variables explicatives et la cible est supposée linéaire.
- **Indépendance des erreurs** : les erreurs de prédiction ne doivent pas être corrélées entre elles.
- **Homoscedasticité** : la variance des erreurs est constante à travers toutes les observations.
- **Non-colinéarité** : les variables explicatives ne doivent pas être fortement corrélées entre elles.

En raison de sa simplicité, la régression linéaire est utilisée comme **modèle de référence (baseline)**, permettant d'évaluer la complexité et l'amélioration apportées par les modèles plus avancés.

4.2.2 Formulation Mathématique

Le modèle de régression linéaire suppose que la variable cible y est une combinaison linéaire des p variables explicatives $x_1, x_2, \ldots, x_p$, avec des coefficients β_i et un terme d'erreur ε :

$$y = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \varepsilon \tag{1}$$

Les coefficients β_i sont estimés par la **minimisation de l'erreur quadratique moyenne (MSE)**, qui se définit comme :

$$\text{MSE} = \frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_j)^2 \tag{2}$$

où y_j est la valeur réelle et $\hat{y}_j$ la valeur prédite.

4.2.3 Performance du Modèle

Nous évaluons la précision du modèle via la **Mean Absolute Error (MAE)**, définie par :

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^{n} |y_j - \hat{y}_j| \tag{3}$$

La MAE permet de quantifier l'écart moyen entre les valeurs prédites et les valeurs réelles, en mettant toutes les erreurs sur le même pied d'égalité sans accentuer les écarts extrêmes comme le ferait le MSE.

Résultat obtenu : le modèle de régression linéaire a produit une MAE d'à peu près **90** et s'est entrainé en **1-2 secondes**.

4.3 Modèle de Forêt Aléatoire (Random Forest)

4.3.1 Principe et Intérêt

La **Forêt Aléatoire** (*Random Forest*) est un modèle d'apprentissage supervisé basé sur un ensemble d'arbres de décision. Elle fonctionne selon le principe du **bagging** (*Bootstrap Aggregating*), qui consiste à entraîner plusieurs arbres sur des sous-échantillons de données et à agréger leurs prédictions pour améliorer la robustesse et la précision.

Les principaux avantages de ce modèle sont :

- **Réduction du sur-apprentissage** : en combinant plusieurs arbres, la variance est diminuée.
- **Robustesse aux valeurs aberrantes et aux données bruitées**.
- **Capacité à capturer des relations complexes et non linéaires** entre les variables.
- **Mesure intégrée de l'importance des caractéristiques**, facilitant l'interprétation du modèle.

En revanche, un inconvénient majeur réside dans le coût computationnel élevé du modèle. Dans notre cas, l'entraînement a duré environ **30 minutes à 1 heure**, bien plus long que la régression linéaire.

4.3.2 Structure et Modèle Mathématique

Une forêt aléatoire est une collection de M arbres de décision $h_m(X)$, où chaque arbre est entraîné sur un sous-échantillon aléatoire des données :

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} h_m(X) \tag{4}$$

Chaque arbre $h_m(X)$ est un arbre de décision construit selon la méthode CART (*Classification and Regression Trees*), où les partitions des données sont définies en minimisant un critère d'impureté tel que l'**erreur quadratique moyenne (MSE)** pour les problèmes de régression :

$$\text{MSE} = \sum_{i \in L} (y_i - \bar{y}_L)^2 + \sum_{i \in R} (y_i - \bar{y}_R)^2 \tag{5}$$

où L et R sont les partitions gauche et droite de l'arbre lors d'un split, et $\bar{y}_L$ et $\bar{y}_R$ sont les moyennes des cibles dans ces partitions.

4.3.3 Évaluation et Performance

Le modèle a été évalué à l'aide de la **Mean Absolute Error (MAE)**, définie par :

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^{n} |y_j - \hat{y}_j| \tag{6}$$

Nous avons obtenu une **MAE d'à peu près 17**, montrant une nette amélioration par rapport à la régression linéaire (**90.00**).

Analyse des caractéristiques importantes : L'un des atouts majeurs de la Forêt Aléatoire est sa capacité à estimer l'importance des variables dans la prédiction. L'analyse montre que les trois caractéristiques les plus déterminantes sont :

- **Generator Speed** (≈ 0.7)
- **Rotor Speed** (≈ 0.2)
- **Pitch Angle Std** (≈ 0.1)

Le graphe ci-dessous illustre les dix caractéristiques les plus influentes dans le modèle :

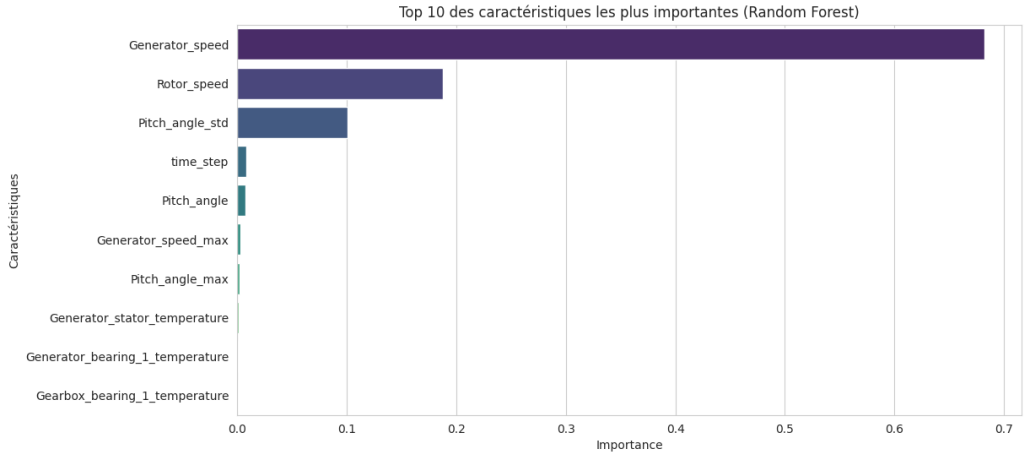

FIGURE 3 – Top 10 des caractéristiques les plus importantes selon la Forêt Aléatoire.

4.4 Modèle XGBoost

4.4.1 Principe et Intérêt

Le modèle **XGBoost** (*Extreme Gradient Boosting*) est une amélioration du boosting de gradient (*Gradient Boosting Machine, GBM*). Il repose sur l'entraînement séquentiel d'arbres de décision, où chaque nouvel arbre corrige les erreurs du précédent. Parmi ses principaux atouts :

- **Meilleure gestion des erreurs résiduelles** grâce à l'optimisation du boosting par second ordre.
- **Grande efficacité computationnelle** (parallélisation, gestion de mémoire optimisée).

— **Robustesse face aux données bruitées** grâce à la régularisation L_1 et L_2.
— **Excellente performance sur des tâches complexes** de régression et de classification.

En termes de temps d'exécution, XGBoost se distingue par son efficacité, avec un entraînement réalisé en **1 à 10 secondes** dans notre cas, bien plus rapide que Random Forest (**30 minutes à 1 heure**).

4.4.2 Structure et Modèle Mathématique

XGBoost suit le principe du **boosting de gradient**, où chaque nouvel arbre $h_m(X)$ est construit pour minimiser une fonction de coût $\mathcal{L}$ basée sur les erreurs des prédictions précédentes :

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(X_i)) + \Omega(f_t) \tag{7}$$

avec :
— $\mathcal{L}^{(t)}$: fonction de perte à l'itération t.
— $l(y_i, \hat{y}_i)$: perte entre la valeur réelle et la prédiction.
— $f_t(X)$: nouvel arbre entraîné sur le résidu.
— $\Omega(f_t)$: terme de régularisation pour contrôler la complexité du modèle.
Contrairement aux autres méthodes, XGBoost exploite une **approximation de second ordre** de la fonction de perte pour accélérer l'optimisation et améliorer la stabilité.

4.4.3 Évaluation et Performance

La **Mean Absolute Error (MAE)** obtenue est de **14.5**, soit la meilleure performance parmi les modèles évalués jusqu'ici :

Régression Linéaire : 90.00 | Random Forest : 17.00 | XGBoost : 14.50

4.5 Réseau de Neurones

4.5.1 Présentation du modèle

Les réseaux de neurones artificiels (ANN) sont des modèles d'apprentissage inspirés du cerveau humain, capables de capturer des relations complexes dans les données. Contrairement aux modèles classiques comme la régression linéaire ou les forêts aléatoires, un réseau de neurones est composé de plusieurs couches de neurones interconnectés, permettant d'extraire des caractéristiques non linéaires.

Nous avons ici construit un réseau de neurones dense (*feedforward*) avec deux couches cachées et des fonctions d'activation ReLU. Ce modèle est particulièrement adapté aux relations complexes présentes dans nos données.

4.5.2 Modèle mathématique et principes

Un réseau de neurones profond se compose de plusieurs couches de transformation. À chaque couche l, les sorties $h^{(l)}$ sont calculées comme :

$$h^{(l)} = f(W^{(l)} h^{(l-1)} + b^{(l)}) \tag{8}$$

où :

— $W^{(l)}$ est la matrice des poids de la couche l,
— $b^{(l)}$ est le vecteur des biais,
— $f(\cdot)$ est la fonction d'activation, ici la fonction ReLU :

$$\text{ReLU}(x) = \max(0, x) \tag{9}$$

L'apprentissage est réalisé en minimisant la fonction de perte, ici l'erreur absolue moyenne (MAE) :

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{10}$$

où y_i représente les valeurs cibles et $\hat{y}_i$ les prédictions du modèle.

4.5.3 Expérimentations et ajustements

Différents paramètres ont été testés afin d'optimiser la performance du modèle :
— **Effet des variables faiblement corrélées** : En entraînant le modèle avec toutes les variables, nous avons observé une MAE supérieure à 50. L'élimination des variables les moins corrélées a significativement amélioré la précision.
— **Effet de la taille des batchs** : Avec une taille de batch de 32, la MAE restait au-dessus de 30. L'augmentation à 1024 a permis une stabilisation plus rapide et une **MAE d'environ 20**.
— **Précaution contre l'overfitting** : Un mécanisme d'arrêt précoce a été mis en place pour éviter le sur-apprentissage.
L'évolution de la fonction de perte au cours de l'entraînement est représentée ci-dessous :

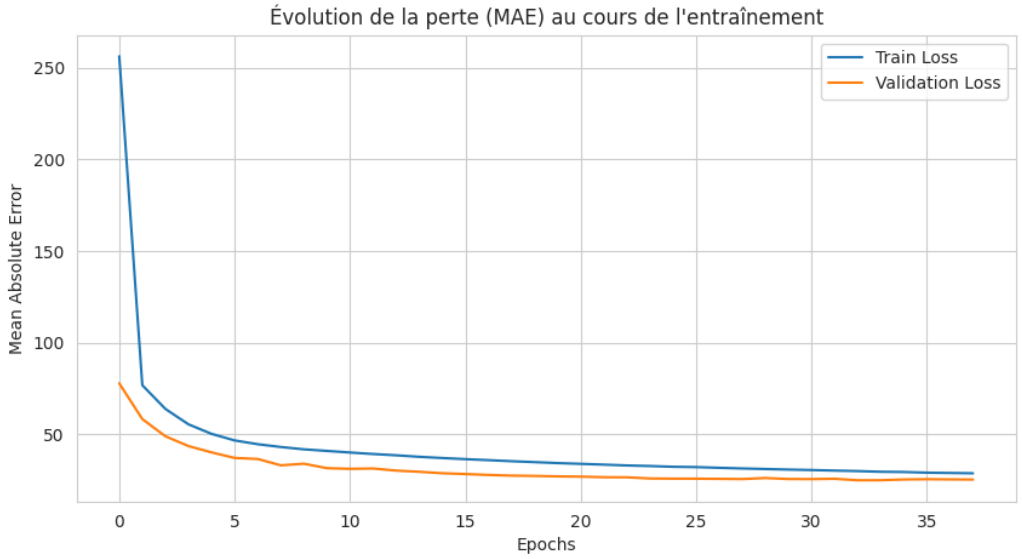

FIGURE 4 – Évolution de la MAE en fonction des époques

4.5.4 Performance et analyse des résultats

Le réseau de neurones a atteint une **MAE de 20** après un entraînement d'environ 5 minutes. Performance un peu inférieure à celle du Random Forest et de XGBoost, avec un temps relativement très optimal par rapport à Random Forest.

4.6 LSTM (Long Short-Term Memory)

4.6.1 Présentation du modèle

Les réseaux LSTM (Long Short-Term Memory) sont des modèles récurrents adaptés pour capturer des dépendances temporelles dans les données séquentielles. Dans le contexte de la régression, ces modèles sont souvent utilisés lorsque l'ordre des données ou des séquences est crucial. Cependant, dans notre cas, même si nous avons une variable temporelle dans les données, nous allons tester l'efficacité de ce modèle sur cette tâche.

Le modèle LSTM consiste en des "cellules" qui mémorisent des informations à long terme tout en supprimant les informations non pertinentes. Cela permet de mieux gérer des séries temporelles et d'exploiter des patterns séquentiels. Le modèle est composé d'une couche LSTM suivie d'une couche dense pour la régression.

4.6.2 Modèle mathématique et principes

LSTM fonctionne avec des cellules mémoire qui se caractérisent par des opérations sur des portes :

— La porte d'oubli f_t, qui décide des informations à oublier.
— La porte d'entrée i_t, qui décide des nouvelles informations à mémoriser.
— La porte de sortie o_t, qui décide de la sortie du modèle.

Les équations des LSTM sont les suivantes :

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{11}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{12}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{13}$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{14}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{15}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{16}$$

Le modèle d'apprentissage dans le cadre de la régression, avec la fonction de perte MAE, est le même que pour les réseaux de neurones. La fonction de perte est alors calculée comme suit :

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{17}$$

4.6.3 Expérimentations et ajustements

Nous avons modifié les dimensions de nos données d'entrée pour qu'elles soient compatibles avec le modèle LSTM. Pour ce faire, nous avons redimensionné les données d'entraînement et de test pour qu'elles aient la forme suivante : $(samples, time\_steps, features)$.

— **Impact des données temporelles** : Bien que nous ayons des variables temporelles dans nos données, l'entraînement d'un modèle LSTM a donné des résultats moins bons que le modèle de réseau de neurones classique, avec une MAE de 27. Cela suggère que, bien que les données soient temporelles, l'information temporelle n'est pas essentielle pour la prédiction dans ce cas particulier.
— **Optimisation** : Le modèle a été entraîné avec un batch size de 1024, avec des callbacks pour l'arrêt précoce et une patience de 3 époques.

4.6.4 Performance et analyse des résultats

Le modèle LSTM a donné une **MAE de 27**, ce qui est moins bon que les autres modèles comme le réseau de neurones et le Random Forest. Ce résultat suggère que, même si les données contiennent une dimension temporelle, cette dernière ne joue pas un rôle important dans la tâche de régression. Les modèles classiques ont donc montré de meilleures performances.

4.7 Comparaison des performances et du temps d'entraînement

La figure 5 présente la comparaison entre les différents modèles.

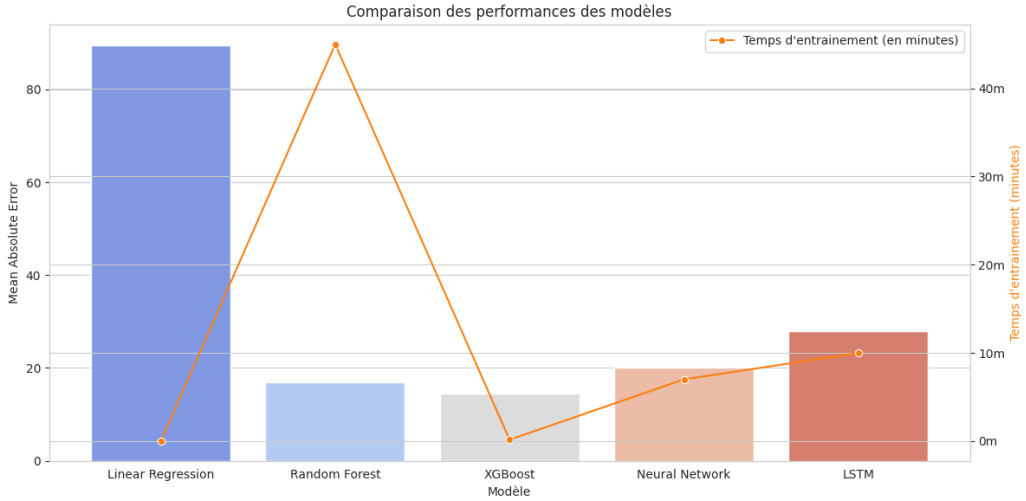

FIGURE 5 – Comparaison des performances (MAE) et du temps d'entraînement des modèles.

Les résultats des modèles comparés révèlent des compromis intéressants entre le temps d'entraînement et la précision des prédictions. La régression linéaire, avec un temps d'entraînement de seulement 1 seconde, présente une *MAE* élevée de 90. Bien qu'il s'agisse d'un modèle de base peu précis, il se distingue par sa rapidité d'exécution. En revanche, des modèles tels que *XGBoost*, avec un temps d'entraînement de 10 secondes et une *MAE* de 14, offrent le meilleur compromis entre performance et temps, apparaissant comme une option idéale lorsque la rapidité et la précision sont essentielles.

Le modèle *Random Forest*, qui prend environ 45 minutes pour s'entraîner, offre une *MAE* de 17. Bien que nécessitant des ressources computationnelles considérables, il représente un bon compromis en termes de performance.

Le réseau de neurones (*NN*), avec un temps d'entraînement de 7 minutes et une *MAE* de 20, présente de bonnes performances mais reste relativement plus lent à l'entraînement. Cette lenteur peut être attribuée à la complexité de son architecture, qui nécessite des ajustements pour optimiser ses résultats. Enfin, le modèle *LSTM*, avec un temps d'entraînement d'environ 10 minutes et une *MAE* de 27, ne semble pas apporter une amélioration significative par rapport aux autres modèles. Cela suggère que, bien que les données présentent une composante temporelle, la capture des dépendances temporelles n'est pas cruciale dans ce cas particulier.

En effet, en examinant l'importance des caractéristiques dans le modèle *Random Forest*, nous constatons que la caractéristique "time steps" a une importance inférieure à 0.1. Cela indique que, bien que cette donnée temporelle soit présente, elle n'a pas un impact majeur sur

les prédictions. Cette observation renforce l'idée que la capture des dépendances temporelles, telle qu'elle est effectuée dans le modèle *LSTM*, n'est pas essentielle pour ce jeu de données particulier, comme le montre la figure de l'importance des caractéristiques.