RAPPORT DE PROJET :

Deep Learning

Réalise Par:

Abdoulfatah Omar Hassan

Encadré par : Pr. Okar

École Nationale des Sciences Appliquées de Tétouan

2025 - 2026

SOMMAIRE

Rapport de Projet : Prévision de la Demande Électrique Britannique

Introduction

Ce projet explore différentes approches de prévision pour anticiper la consommation électrique du réseau national britannique. En exploitant les données historiques du National Grid ESO couvrant la période 2009-2024, nous comparons systématiquement plusieurs algorithmes de pointe en intelligence artificielle et apprentissage profond. L'objectif principal est d'identifier la méthode la plus performante pour prédire la demande électrique à court et moyen terme, un enjeu crucial pour la gestion optimale des réseaux électriques et la sécurité d'approvisionnement.

Méthodologie générale

Le projet suit une approche structurée en deux phases principales :

Phase 1 : Préparation des Données

- Analyse exploratoire et nettoyage des données
- Ingénierie des caractéristiques temporelles
- Détection des tendances et cycles saisonniers
- Intégration des jours fériés britanniques

Phase 2 : Implémentation des Modèles

- XGBoost : Algorithme de boosting performant pour les séries temporelles
- Arbres Linéaires : Approche hybride innovante combinant arbres et modèles linéaires
- Prophet : Solution spécialisée de Meta pour les séries temporelles
- LSTM : Réseaux neuronaux récurrents avec mécanisme de mémoire
- LSTM Profonds : Architectures multi-couches pour une modélisation avancée
- GRU (Gated Recurrent Unit) : Variante plus efficace des LSTM

1. Préparation et Analyse des Données

1.1 Chargement et Exploration Initiale

Le jeu de données initial comprend 279,264 observations avec 20 variables caractérisant la demande électrique et les flux d'interconnexion. Après analyse, nous avons conservé les variables les plus pertinentes et éliminé les colonnes présentant un taux important de valeurs manquantes (nsl_flow et eleclink_flow).

Variables clés retenues :

- settlement_date : Date de l'observation
- settlement_period : Intervalle de 30 minutes (1-48 par jour)
- tsd (Target System Demand) : Variable cible, demande électrique en MW
- Variables météorologiques et temporelles dérivées

1.2 Nettoyage et Prétraitement

Plusieurs étapes de nettoyage ont été nécessaires :

- Suppression des valeurs aberrantes : Élimination des lignes où settlement_period > 48 (incohérent avec la réalité)
- Gestion des valeurs nulles : Suppression des jours complets où la demande était égale à 0 (erreurs de mesure)
- Normalisation temporelle : Conversion des dates et création d'un index datetime cohérent

1.3 Feature Engineering

L'ingénierie des caractéristiques a été cruciale pour capturer les patterns temporels :

```python
# Maintenant créer les features
df["day_of_month"] = df.index.day
df["day_of_week"] = df.index.day_of_week
df["day_of_year"] = df.index.day_of_year
df["quarter"] = df.index.quarter
df["month"] = df.index.month
df["year"] = df.index.year
df["week_of_year"] = df.index.isocalendar().week.astype("int64")
df["hour"] = df.index.hour if hasattr(df.index, 'hour') else 0
df["is_weekend"] = df["day_of_week"].isin([5, 6]).astype(int)
```

Intégration des jours fériés : Une analyse comparative a confirmé que l'Angleterre et le Pays de Galles partagent les mêmes jours fériés. L'ajout de cette variable s'est avéré essentiel pour capturer les variations spécifiques lors des périodes de congés.

1.4 Analyse Exploratoire

L'analyse visuelle a révélé plusieurs patterns importants :

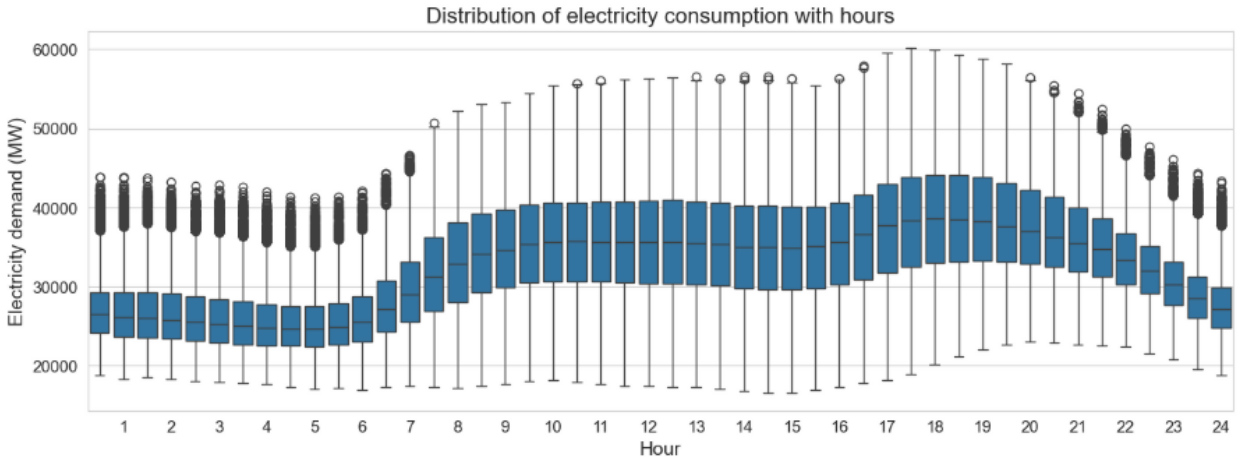
Tendance générale :

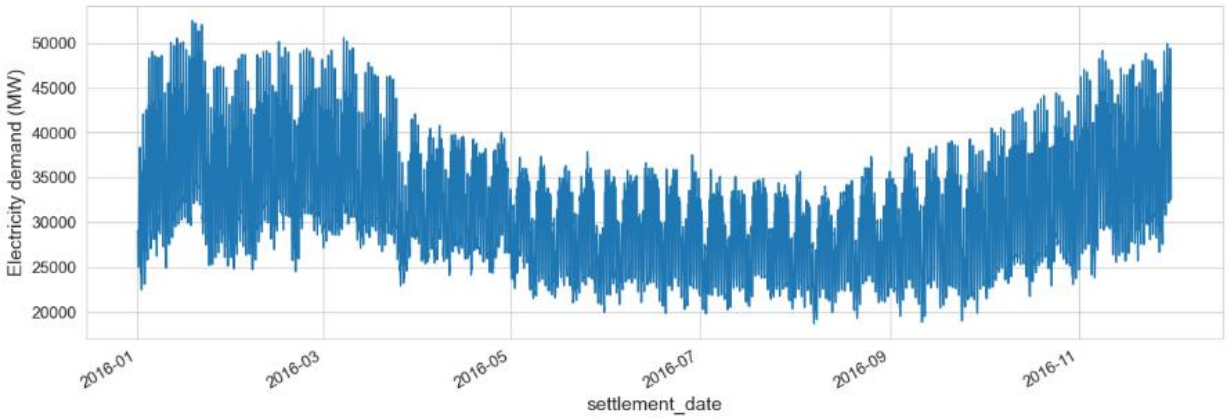
Une diminution progressive de la demande électrique sur la période 2009-2024, probablement liée à l'amélioration de l'efficacité énergétique et au développement des énergies renouvelables décentralisées.

Saisonnalités :

- Saisonnalité annuelle : Demande plus élevée en hiver qu'en été
- Saisonnalité hebdomadaire : Pattern typique semaine/week-end
- Saisonnalité quotidienne : Cycles de 24 heures avec pics et creux

Impact des jours fériés : Réduction significative de la demande les jours fériés en semaine, mais augmentation les week-ends fériés.





2. Implémentation et Évaluation des Modèles

2.1 Méthodologie d'Évaluation

Pour comparer objectivement les modèles, nous avons utilisé deux métriques principales :

- MAPE (Mean Absolute Percentage Error) : Pour évaluer l'erreur relative
- RMSE (Root Mean Square Error) : Pour évaluer l'erreur absolue en MW

La séparation des données a suivi une approche temporelle stricte :

- Période d'entraînement : 2009-2019
- Période de test : 2019-2021
- Période de validation : 2021-2024

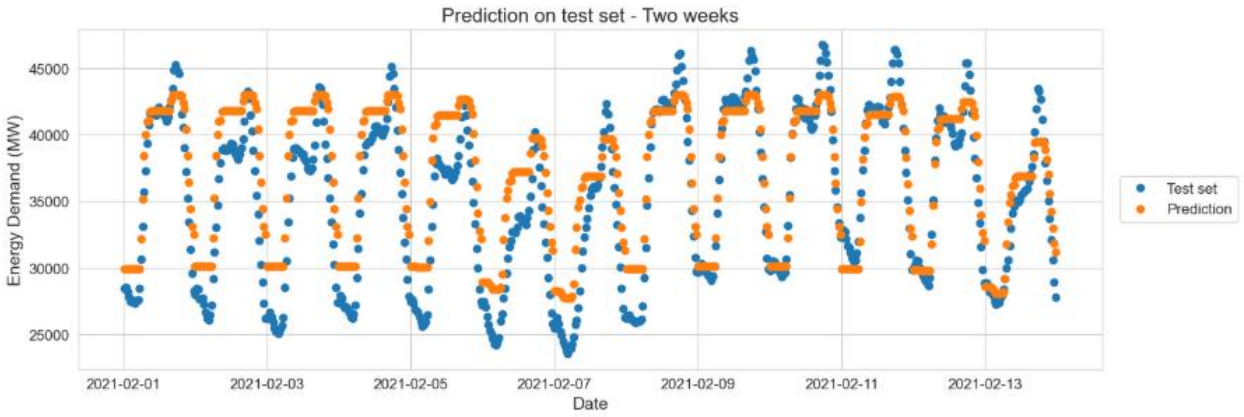
2.2 Modèles Implémentés

2.2.1 XGBoost Simple

Le modèle XGBoost de base a servi de référence avec une configuration standard :

```
# Initialize and fit the XGBoost model
xgb_simple = xgb.XGBRegressor(
    n_estimators=500,
    max_depth=3,
    learning_rate=0.01,
    early_stopping_rounds=50,
    random_state=43,
)
```

Résultats : MAPE = 11.29%, RMSE = 3786.88 MW



Le modèle capture bien la tendance générale mais peine sur les pics et creux.

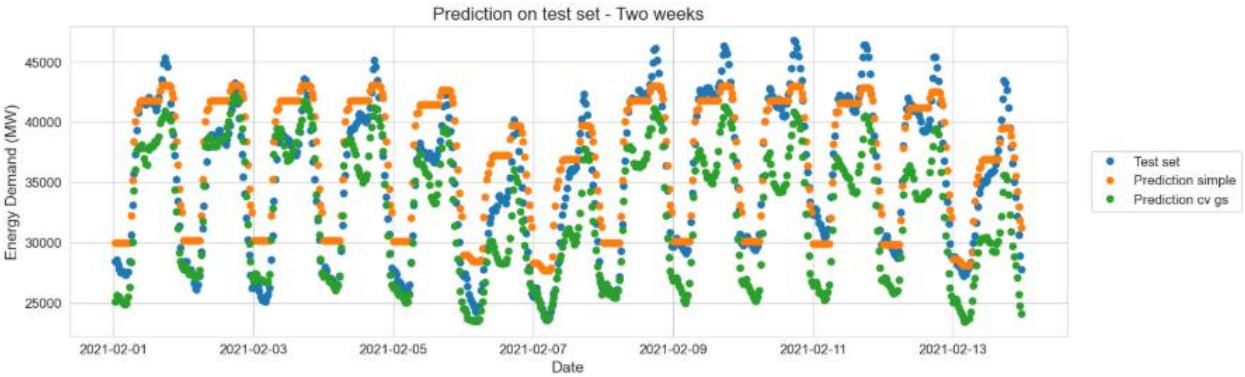
2.2.2 XGBoost avec Validation Croisée et Grid Search

L'optimisation hyperparamétrique a considérablement amélioré les performances :

```
# Define parameters to optimiser
param_search = {
    "max_depth": [3, 5],
    "n_estimators": [100, 200, 300],
    "subsample": [0.8, 0.9],
}
```

Meilleurs paramètres : max_depth=5, n_estimators=300, subsample=0.8

Résultats : MAPE = 7.43%, RMSE = 2794.67 MW



2.2.3 Linear Trees

Approche hybride combinant arbres de décision et régression linéaire :

```
linear_boost = LinearBoostRegressor(
    base_estimator=LinearRegression(n_jobs=-1),
    n_estimators = 350,
    max_depth = 6,
    random_state = 43,
)
```

Résultats : MAPE = 8.49%, RMSE = 3045.82 MW

Bon compromis entre performance et interprétable.

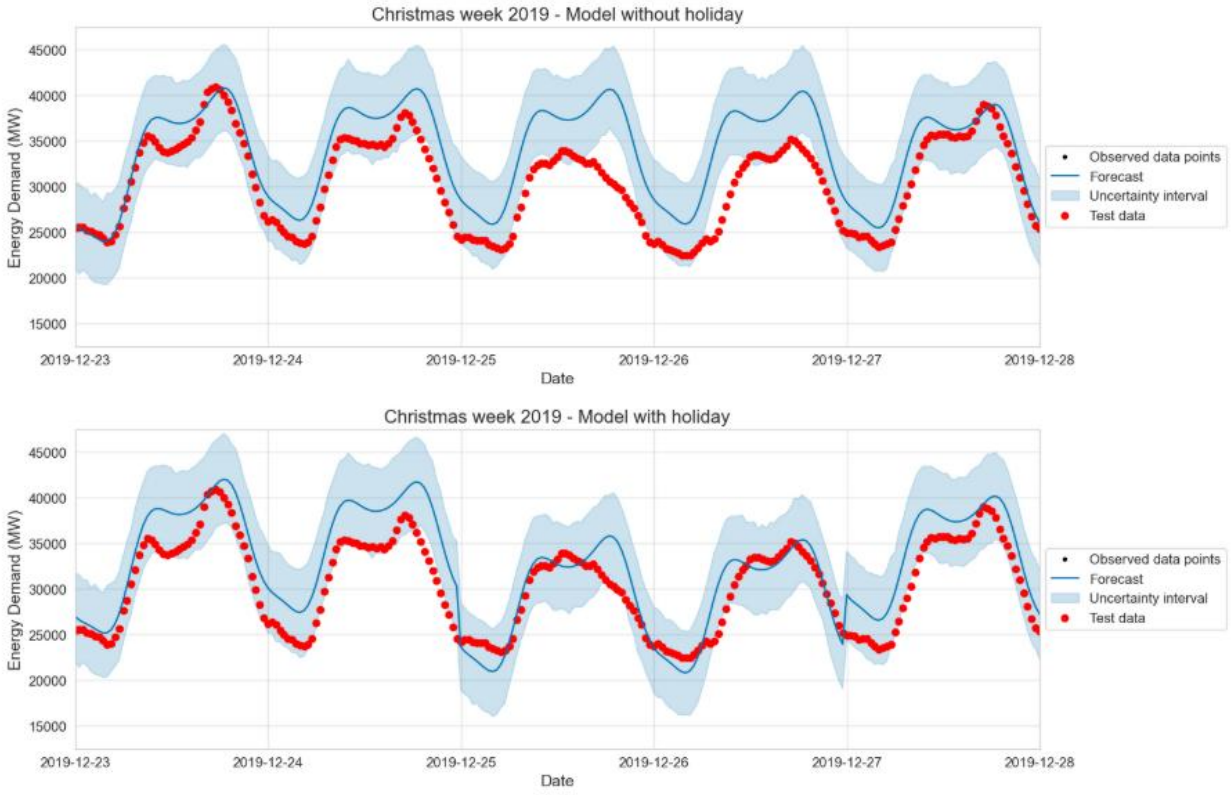
2.2.4 Prophet

Modèle spécialisé dans les séries temporelles avec composantes saisonnières :

Version simple : MAPE = 9.37%, RMSE = 3262.45 MW

Avec jours fériés : MAPE = 9.35%, RMSE = 3236.63 MW

L'amélioration avec les jours fériés est marginale mais visible sur les périodes spécifiques.



2.2.5 LSTM (Long Short-Term Memory)

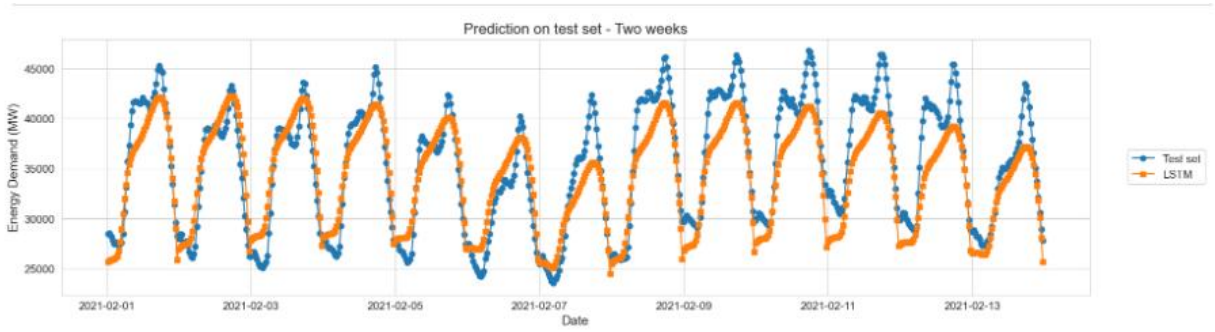
Architecture neuronale récurrente spécialisée pour les séquences temporelles :

```
# Create and compile neural network
model = tf.keras.Sequential()
model.add(tf.keras.layers.LSTM(256, input_shape=(X_train_keras.shape[1], X_train_keras.shape[2])))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(1))

model.compile(loss=root_mean_squared_error, optimizer="adam")
```

Résultats : MAPE = 7.59%, RMSE = 2785.54 MW

Meilleure capture des dépendances temporelles que les approches traditionnelles.



2.2.6 LSTM Profond

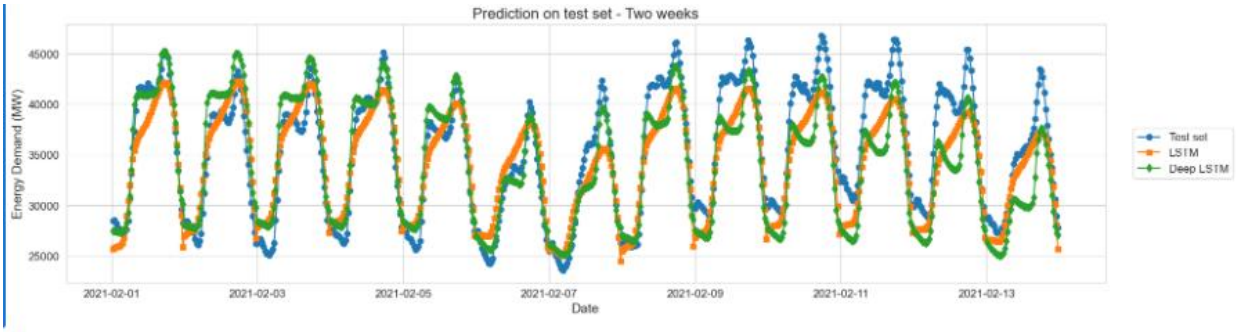
Architecture multi-couches pour une modélisation plus complexe :

```
# Create and compile neural network
model = Sequential()
model.add(LSTM(256, input_shape=(X_train_keras.shape[1], X_train_keras.shape[2]), return_sequences=True))
model.add(Dropout(0.5))
model.add(LSTM(128, return_sequences=True))
model.add(Dropout(0.5))
model.add(LSTM(32))
model.add(Dropout(0.5))

model.add(Dense(1))
model.compile(loss = root_mean_squared_error, optimizer="adam")
```

Résultats : MAPE = 7.37%, RMSE = 2648.19 MW

Performance optimale - meilleur modèle du benchmark.



2.2.7 GRU Profond

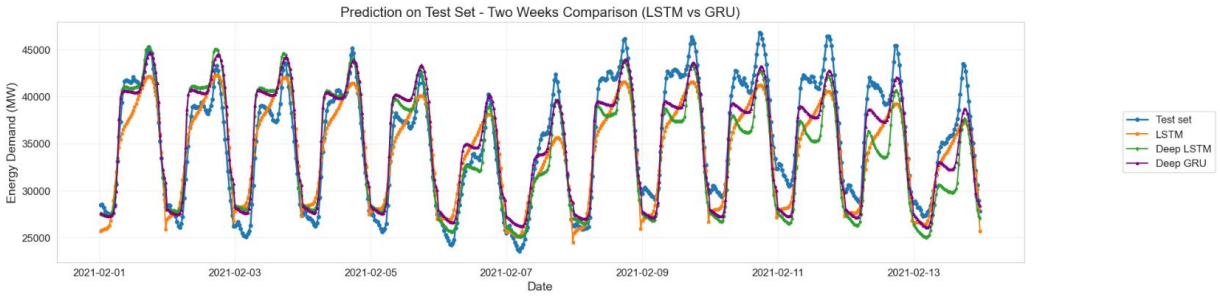
Variante des LSTM avec architecture simplifiée :

```
# Create and compile neural network GRU
model_gru = Sequential()
model_gru.add(GRU(256, input_shape=(X_train_keras.shape[1], X_train_keras.shape[2]), return_sequences=True))
model_gru.add(Dropout(0.5))
model_gru.add(GRU(128, return_sequences=True))
model_gru.add(Dropout(0.5))
model_gru.add(GRU(32))
model_gru.add(Dropout(0.5))

model_gru.add(Dense(1))
model_gru.compile(loss=root_mean_squared_error, optimizer="adam")
```

Résultats : MAPE = 7.62%, RMSE = 2724.78 MW

Performance très proche des LSTM avec entraînement potentiellement plus rapide.



3. Analyse Comparative et Résultats

3.1 Tableau Synthétique des Performances

Modèle	MAPE (%)	RMSE (MW)	Temps d'entraînement	Complexité
XGBoost Simple	11.29	3786.88	Faible	Faible
XGBoost Optimisé	7.43	2794.67	Moyen	Moyenne
Linear Boost	8.49	3045.82	Élevé	Moyenne
Prophet Simple	9.37	3262.45	Faible	Faible
Prophet avec Holidays	9.35	3236.63	Faible	Faible
LSTM	7.59	2785.54	Élevé	Élevée
LSTM Profond	**7.37**	**2648.19**	Très élevé	Très élevée
GRU Profond	7.62	2724.78	Élevé	Élevée

3.2 Principaux Enseignements

1. L'optimisation hyperparamétrique est cruciale : XGBoost optimisé gagne près de 4 points de MAPE par rapport à la version simple.

2. La profondeur architecturale paye : Les LSTM profonds surpassent toutes les autres approches, confirmant leur capacité à modéliser des dépendances temporelles complexes.

3. Le feature engineering temporel est essentiel : L'ajout des lags (décalages temporels) et des caractéristiques calendaires a significativement amélioré toutes les approches.

4. Les modèles spécialisés battent les généralistes : Prophet, bien que spécialisé, est surpassé par les approches neuronales adaptées.

5. Compromis performance/coût : XGBoost optimisé offre le meilleur rapport performance/coût computationnel.

4. Recommandations et Perspectives

4.1 Modèle Recommandé : LSTM Profond

Le LSTM profond émerge comme le choix optimal pour les prévisions de demande électrique, avec les avantages suivants :

- Meilleure précision globale (MAPE le plus bas)
- Excellente capture des patterns complexes (pics, creux, transitions)
- Robustesse aux variations saisonnières
- Adaptabilité aux changements de régime

4.2 Améliorations Possibles

Pour des performances encore supérieures :

1. Architectures hybrides : Combiner LSTM avec des mécanismes d'attention

2. Incorporation de données externes : Météo, prix de l'énergie, données économiques

3. Apprentissage par transfert : Utiliser des modèles pré-entraînés sur des données similaires

4. Optimisation avancée : Recherche hyperparamétrique plus poussée avec Bayesian Optimization

4.3 Applications Pratiques

Le modèle déployé peut servir à :

- Gestion en temps réel des réseaux électriques
- Optimisation économique de la production
- Planification maintenance des infrastructures
- Intégration optimale des énergies renouvelables

Conclusion

Ce projet démontre la supériorité des approches neuronales profondes, en particulier les LSTM, pour la prévision de demande électrique. Le LSTM profond atteint une précision remarquable avec seulement 7.37% d'erreur moyenne, tout en capturant finement la complexité des patterns temporels.

Cependant, le choix final du modèle doit considérer le contexte opérationnel: si la performance pure prime, le LSTM profond est indépassable ; si les contraintes computationnelles ou l'interprétabilité sont importantes, XGBoost optimisé constitue une excellente alternative.

Ces résultats ouvrent des perspectives prometteuses pour l'application de l'intelligence artificielle dans la gestion des réseaux électriques intelligents et la transition énergétique.