Note d'information : Dashboard de Veille Économique sur les Énergies Renouvelables

Synthèse Exécutive

Ce document présente une analyse détaillée du projet "Dashboard Veille Économique - Énergies Renouvelables Afrique Francophone". L'objectif principal est de fournir un outil interactif pour la collecte, la visualisation et l'analyse prédictive des données sur les énergies renouvelables. L'application, construite avec Streamlit, exploite l'API de la Banque Mondiale pour récupérer automatiquement des indicateurs clés pour plusieurs pays d'Afrique francophone. Son architecture intègre un modèle de régression linéaire multivariée bayésienne avec PyMC pour réaliser des prévisions futures, ainsi qu'un système d'alerte par email configurable via SMTP. L'interface utilisateur est dynamique, permettant des filtrages par pays, indicateurs et périodes, et offre des visualisations interactives via Plotly et Matplotlib. Le projet est conçu pour être extensible, avec des possibilités d'intégration de données supplémentaires via scrapping et de déploiement sur des plateformes cloud.

1. Présentation du Projet

- **Nom du Projet** : Dashboard Veille Économique - Énergies Renouvelables Afrique Francophone
- **Objectif** : Mettre à disposition un outil interactif pour automatiser la collecte, la visualisation et l'analyse bayésienne des données sur les énergies renouvelables. Les données proviennent de la Banque Mondiale et concernent plusieurs pays d'Afrique francophone.

2. Architecture et Composants Techniques

L'application est structurée autour de plusieurs modules fonctionnels clés, allant de l'acquisition des données à leur restitution à l'utilisateur.

2.1. Collecte des Données

- **Source** : L'API de la Banque Mondiale est utilisée pour la récupération automatique des indicateurs.
- **Indicateurs Clés** : Les principaux indicateurs collectés sont la "Part énergies renouvelables (%)" et la "Production électricité renouvelable (kWh)".
- **Périmètre** : Le système permet une récupération multi-pays et multi-indicateurs sur des plages temporelles configurables, avec une période par défaut fixée de 2000 à 2024.
- **Optimisation** : La fonction fetch_worldbank_data() utilise le système de cache de Streamlit (@st.cache) pour minimiser les appels API redondants et améliorer les performances.

2.2. Analyse Bayésienne

- **Modèle** : Une régression linéaire multivariée bayésienne est implémentée en utilisant la bibliothèque PyMC.
- **Prédictions** : Le modèle génère des prédictions sur un nombre d'années futures configurable par l'utilisateur (3 ans par défaut).
- **Validation du Modèle** : Des diagnostics MCMC (Monte Carlo par chaînes de Markov) sont disponibles pour évaluer la convergence et la qualité du modèle. Cela inclut des graphiques de trace (trace plots), le test de R-hat et l'analyse des autocorrélations.

2.3. Visualisation des Données

- **Graphiques Interactifs** : Des séries temporelles sont générées avec Plotly, permettant une exploration interactive des données par pays et par indicateur.
- **Résultats Bayésiens** : Les résultats de l'analyse prédictive, y compris les intervalles de crédibilité, sont visualisés à l'aide de graphiques Matplotlib.
- **Tableaux Synthétiques** : Un tableau récapitulatif présente de manière concise les données observées et les prévisions générées.

2.4. Système d'Alertes

- **Fonctionnalité** : Une option d'alerte par email est disponible pour notifier un utilisateur si un indicateur dépasse un seuil prédéfini.
- **Technologie** : L'envoi des emails est géré via le protocole SMTP, avec des paramètres configurables pour un compte Gmail.
- **Interface** : L'utilisateur peut activer ou désactiver les alertes, définir les seuils, spécifier l'email du destinataire et tester la configuration directement depuis l'interface.

2.5. Interface Utilisateur (UI)

- **Framework** : L'interface est entièrement construite avec la bibliothèque Streamlit.
- **Navigation** : Une barre latérale (sidebar) regroupe les filtres dynamiques, permettant à l'utilisateur de sélectionner les pays, les indicateurs, la période d'analyse et de configurer les alertes.
- **Apparence** : La mise en page est responsive et personnalisée avec du CSS pour inclure un thème vert et des animations simples.

3. Installation et Déploiement

3.1. Prérequis

- **Environnement** : Python 3.9 ou une version ultérieure.
- **Dépendances Python** : Les bibliothèques nécessaires peuvent être installées via pip à partir du fichier requirements.txt. La commande est la suivante : pip install streamlit pandas numpy requests pymc arviz matplotlib plotly
- **Configuration SMTP** : Un compte Gmail avec un mot de passe d'application est requis pour la fonctionnalité d'alerte par email.

3.2. Lancement de l'Application

1. Télécharger les fichiers du projet, notamment dashboard.py.
2. Configurer les identifiants d'envoi (email et mot de passe d'application) directement dans la fonction send_email().
3. Exécuter la commande suivante dans le terminal : streamlit run dashboard.py

4. Guide d'Utilisation

4.1. Configuration des Filtres

- **Pays** : Sélectionner un ou plusieurs pays d'Afrique francophone dans la liste.
- **Indicateurs** : Choisir entre "Part énergies renouvelables (%)" et "Production électricité renouvelable (kWh)".
- **Années** : Définir la plage temporelle pour l'analyse des données historiques.
- **Prévision** : Spécifier le nombre d'années futures pour lesquelles l'analyse bayésienne doit générer des prédictions.

4.2. Consultation des Données

- Les données brutes récupérées sont affichées dans des tableaux interactifs.
- Les graphiques temporels permettent de visualiser et de comparer les tendances entre les pays et les indicateurs sélectionnés.
- Un bouton "Export CSV" permet de télécharger les données affichées.

4.3. Interprétation de l'Analyse Bayésienne

- **Périmètre** : L'analyse est effectuée uniquement sur le premier pays et les deux premiers indicateurs sélectionnés par l'utilisateur.
- **Résultats** : L'application affiche les courbes prévisionnelles avec un intervalle de confiance de 95%. Un tableau synthétique résume les valeurs observées et prédites.
- **Diagnostics** : Des graphiques et statistiques de diagnostic sont fournis pour permettre à l'utilisateur de vérifier la qualité du modèle prédictif.

4.4. Gestion des Alertes Email

- L'activation se fait via la barre latérale.
- L'utilisateur doit choisir un pays, un indicateur et définir une valeur seuil.
- L'adresse email du destinataire doit être saisie.
- Un bouton de test permet de vérifier la configuration d'envoi. Si une valeur observée dépasse le seuil, un email est envoyé automatiquement.

5. Structure du Code et Personnalisation

Fichier	Description
dashboard.py	Script principal Streamlit contenant la logique de l'UI, la collecte, l'analyse, les visualisations et les alertes.
requirements.txt	Fichier listant toutes les dépendances Python requises pour le projet.

Les paramètres suivants doivent être personnalisés dans la fonction send_email() du fichier dashboard.py :

- from_email = "ton.email@gmail.com"
- password = "ton_mdp_app"

6. Évolutions et Extensions Possibles

Le projet est conçu avec une architecture permettant des améliorations futures :

- **Sources de Données** : Ajout de fonctionnalités de web scraping pour collecter des données complémentaires.
- **Périmètre d'Analyse** : Extension de l'application pour inclure davantage d'indicateurs et de pays.
- **Alertes Avancées** : Intégration d'un système d'alertes multiples et planifiées automatiquement (envoi régulier).
- **Exports** : Ajout de fonctionnalités d'export au format PDF.
- **Déploiement** : Hébergement de l'application sur une plateforme cloud (Heroku, AWS, etc.) pour garantir un accès 24/7.

7. Annexes et Ressources

- **Documentation PyMC** : https://docs.pymc.io
- **Documentation Streamlit** : https://docs.streamlit.io
- **Documentation API World Bank** : https://datahelpdesk.worldbank.org/knowledgebase/articles/889386-api-documentation