Méthodologie et Fonctionnement de notre Tableau de Bord de Suivi Forestier

Notre projet vise à fournir un tableau de bord interactif pour l'analyse et le suivi de la couverture végétale et de la déforestation au Sénégal, en s'appuyant sur des données raster GeoTIFF.

Origine et Adaptation du Projet

Pour démarrer ce développement, nous nous sommes initialement inspirés d'un exemple d'application trouvé au sein de la galerie d'applications Dash, une ressource riche accessible publiquement à l'adresse : https://github.com/InesRoque3/GroupV_project2 . Afin de maîtriser sa structure et de pouvoir l'adapter efficacement, nous avons entrepris de déconstruire cet exemple. Nous l'avons découpé en plusieurs scripts Python interdépendants dont les differents commits permettent de suivre le cheminement dans le dossier dashboard:

- app.py : Le point d'entrée principal, définissant la structure globale du layout avec Dash Bootstrap Components et initialisant l'application.
- components/ : Un répertoire contenant des modules pour créer les éléments visuels réutilisables (sidebar, sélecteurs, graphiques, zone d'affichage raster). Par exemple, selectors.py contient des fonctions comme create_forest_selector ou create_year_slider même si au fil des commits il y a beaucoup de changement.
- callbacks/ : Contient la logique interactive de l'application, notamment main_callback.py qui orchestre la mise à jour des composants.
- utils/ : Modules dédiés au chargement et au prétraitement des données (lecture depuis AWS S3, calculs raster) et aux constantes.Dans la suite pour faciliter l'importations et ne pas surcharger la path de la variable d'environnement (windows) les fichiers dedie au load de data sont directement integre dans le dashboard/

Noter que dans le dossier pretraitement nous avons entrepris des pretraitement sur nos donnes et nous avons aussi pris soin d'effectuer des testes unitaires et dans le dossier notebooks essentiellement c'est labas que nous avons tester la documentation de certains librairies dont nous nous sommes servis par la suite.

Cette modularité nous a permis de mieux comprendre la logique sous-jacente et de personnaliser l'application pour répondre à nos besoins : le suivi des forêts classées du Sénégal en utilisant nos propres données TIF et en intégrant des analyses temporelles et comparatives.

Fonctionnement Général de l'Application

Notre tableau de bord offre une interface utilisateur structurée :

1. **Sidebar (**components/siderbar.py**) :** Une barre latérale fixe présente l'application et fournit des informations contextuelles. Son style est défini dans assets/styles.css tout comme tous le reste des composantes bien que parfois le style css est directement definie dans la composante dash (souvent dash.html)
2. **Zone de Sélection (** components/selectors.py**) :** Située en haut de la zone de contenu principal, elle permet à l'utilisateur de définir les paramètres d'analyse :
    - **Mode d'Analyse :** "Instantané Annuel" pour visualiser une année spécifique ou "Comparaison Annuelle" pour comparer deux années.
    - **Forêt :** Sélection de la zone d'intérêt parmi celles disponibles (détectées depuis S3 pour la derniere version).
    - **Type de Vue :** Choix entre l'affichage du NDVI calculé ou une vue RGB (quelques soucis avec les couleurs ...).
    - **Année(s) :** Un slider (create_year_slider) sélectionne l'année principale (ou la première année en comparaison), et un dropdown (create_year_selectors) permet de choisir la seconde année pour le mode comparaison.
3. **Zone d'Affichage (**components/map_display.py**) :** C'est la zone principale où les résultats sont visualisés :

- **Affichage Raster (**px.imshow**) :** La partie centrale affiche une image du fichier TIF correspondant à la forêt, l'année et le type de vue sélectionnés. Plutôt qu'une carte interactive type Leaflet, nous utilisons plotly.express.imshow pour un affichage direct des données raster.nous avons chercher la documentation de mapbox mais son implementation (par clé tocken) s'avére payant .Nous nous sommes donc contenté d'un simple px.imshow d'autant plus que la gestion des callbacks pour px.imshow est largement plus simple.
- **Graphique Temporel (**create_secondary_chart_area**) :** Sur la gauche, un graphique linéaire montre l'évolution temporelle de la surface des différentes classes de végétation (calculées à partir des stats NDVI) pour la forêt sélectionnée. Un filtre permet de choisir les classes à afficher.
- **Statistiques et Distribution :** Sous l'affichage raster, un espace combine un graphique à barres montrant la distribution des surfaces par classe pour l'année sélectionnée (si vue NDVI) et un tableau récapitulatif de ces statistiques (surface en ha, % couverture). Ces statistiques sont calculées à la volée par des fonctions comme calcul_class_stats après classification du NDVI (via calcul_ndvi).
- **Informations Contextuelles (**tertiary-content**) :** En dessous, une zone affiche soit la légende des classes NDVI et les tendances générales d'évolution (en mode Instantané), soit un résumé des changements nets par classe (en mode Comparaison).

Gestion des Données et Déploiement AWS

Initialement, durant la phase de développement et de prototypage, nous avons travaillé avec les fichiers TIF stockés localement dans un dossier data/. Afin de pouvoir deployer vers Heroku nous avons mis le chemin du dossier dans constantes.py (bien que malheuresement cela n'ai pas servis pour AWS mais cela c'est avéré utule pour les seuls des NDVI, dont l'estimation a demandé beaucoup de documentation).

C'est aws_data_loader.py qui gère désormais linteraction avec les images . Il utilise la bibliothèque boto3 pour se connecter à notre bucket S3 (hackaton-

stat, préfixe Data_hackathon_stat_data_raster/). Nous avons toujours pris la peine de sauvegarder nos clef secret et d'acces dans un fichier .env afin de respecter les normes de securité.

Les fonctions de traitement comme calcul_ndvi et read_rgb_bands lisent désormais les objets TIF directement depuis S3 en mémoire avant de les traiter avec rasterio. Ce changement est transparent à peu de chose prés pour le reste de l'application, qui continue d'appeler les fonctions de aws_data_loader comme elle appelait celles de l'ancien data_loader local.

Le Rôle des Callbacks

L'interactivité dynamique de notre tableau de bord repose entièrement sur le système de callbacks de Dash. Le fichier callbacks/main_callback.py contient la fonction principale (update_dashboard) . Cette fonction est le "cerveau" de l'application :

- Elle est **déclenchée** par tout changement dans les composants définis comme Input (les sélecteurs, le slider).
- Elle **reçoit** les nouvelles valeurs de ces inputs.
- Elle exécute la logique nécessaire : lecture des données depuis S3 (via aws_data_loader), calculs NDVI/Stats, génération des figures Plotly, formatage des tableaux et des résumés HTML/Bootstrap.
- Elle **retourne** les nouvelles valeurs pour tous les composants définis comme Output (les figures des graphiques, le contenu des zones de texte, les options/valeurs des sélecteurs, etc.).

Pistes Futures (GeoJSON)

Au cours de notre développement, nous avons exploré d'autres approches. Nous avions notamment envisagé de travailler avec des données vectorielles au format GeoJSON, générées à partir de nos rasters classifiés. L'utilisation de GeoJSON, par exemple avec dash-leaflet, aurait pu ouvrir la voie à des fonctionnalités cartographiques plus avancées, comme une

interactivité accrue (clic sur polygone) ou un suivi plus sophistiqué de l'évolution pixel par pixel au fil du temps.

Cependant, face à la complexité de la vectorisation et de l'intégration de ces fonctionnalités, et afin de nous concentrer sur l'objectif principal d'afficher et d'analyser nos données raster TIF sources et leurs statistiques dérivées, nous avons décidé de reporter cette piste GeoJSON pour d'éventuelles évolutions futures.

En l'état actuel, notre application Dash fournit une plateforme fonctionnelle et informative pour explorer les données de couverture forestière au Sénégal stockées sur AWS S3. Elle permet la visualisation raster (NDVI/RGB), l'analyse temporelle et la comparaison entre années, offrant ainsi un outil précieux pour le suivi de la dégradation des forêts du Sénégal.