

PROJET PYTHON / ML
M2 SISE

OCTOBRE 2025



À propos

GreenTech Solutions est une société de service fictive qui développe des applications. Avec l'accélération du changement climatique et la hausse des prix de l'énergie, la sobriété énergétique est au cœur des préoccupations des Français.

C'est pourquoi Enedis nous sollicite en vue d'évaluer l'impact de la classe de Diagnostic de Performance Energétique (DPE) sur les consommations électriques de logements. Vous devez créer une application qui permet à des particuliers d'évaluer le DPE d'un logement et la consommation d'énergie.



Les livrables

Pour ce projet, le client attend plusieurs livrables. Il ne se contentera pas de l'application. Il est important pour lui d'être autonome sur la maintenance de l'application une fois celle-ci opérationnelle.



01

Déploiement de l'application

L'application doit-être déployée sur <https://render.com/> OU <https://shiny.posit.co/py/> OU <https://www.heroku.com/students;>

02

Captation vidéo de l'application

Réaliser une démo enregistrée de l'application pour présenter son fonctionnement et publier en privé sur YouTube.

03

Repository GitHub avec

- tous les scripts utilisés (avec commentaire)
- un README.md pour présenter les informations importantes
- documentation technique (orientée pour développeur) avec schéma de l'architecture dessiné avec [Draw.io](#) - 2 pages max en markdown
- documentation fonctionnelle (orientée utilisateur) - 2 pages max en markdown.

04

Rédaction d'un rapport sur les modèles ML

Ce rapport doit permettre de présenter le contexte et la méthodologie de la construction des modèles retenus. Il est souhaité également d'avoir les métriques et les interprétation des modèles. Le document est rédigé au format Markdown. 4 à 6 pages sont attendues.



DEADLINE

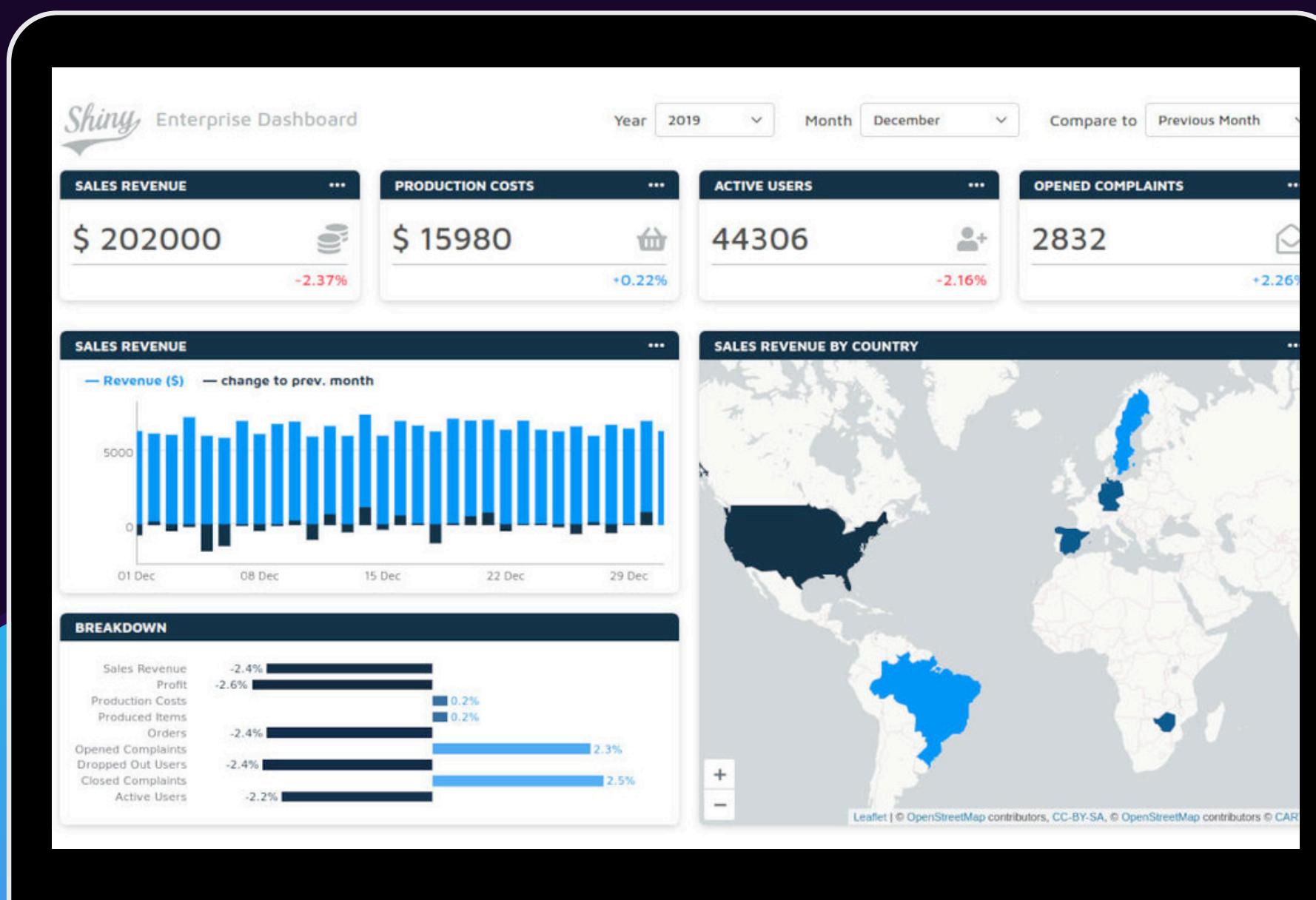
Pour réaliser ce projet, votre équipe a :

- 7 x 3H

Le projet est à rendre au plus tard le :
02/11/2025. -1 pt de pénalité par jour retard supplémentaire



Les technologies et outils à utiliser



- Dash / Shiny
- Python
- Jupyter/Colab
- VS Code
- Git
- ChatGPT
- Azure DevOps

Votre équipe

Pour ce projet, vous serez en équipe de 3. Les équipes seront tirées au sort. Vous aurez plusieurs rôles* à jouer !



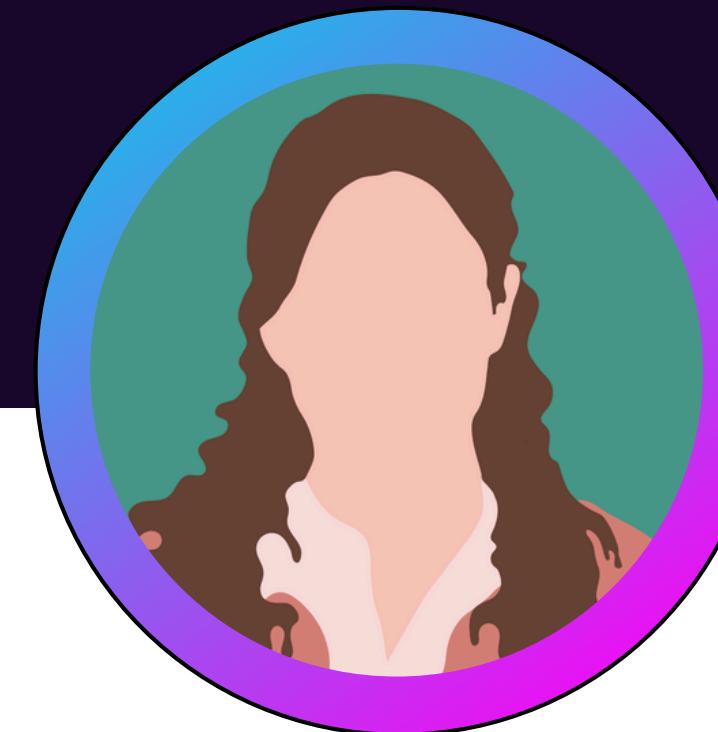
Anthony
Client



Rôle 1
Chef de projet



Rôle 2
Développeur



Rôle 3
Data Scientist

*Chaque étudiant doit pouvoir jouer les 3 rôles durant le projet.



Les rôles

CHEF DE PROJET

est chargé de planifier, coordonner et gérer toutes les activités liées au projet afin de fournir une solution efficace, fonctionnelle et répondant aux besoins du client / des utilisateurs.

DÉVELOPPEUR

joue un rôle essentiel dans le développement et le déploiement de la solution. Il veille à effectuer des tests pour éviter les bugs. Versionner et documenter son code. Il garantit le bon fonctionnement de l'application (back-end / front-end).

DATA SCIENTIST

joue un rôle essentiel dans l'analyse des données, la préparation des données et la création de visualisations pertinentes afin de fournir des informations significatives et exploitables aux utilisateurs finaux. Il est également chargé de construire et d'interpréter des modèles.



Les fonctionnalités



Standard

- au moins 3 onglets/pages différentes
- images et icônes dans l'application
- une cartographie interactive
- page "Contexte" avec les données disponibles
- L'utilisateur peut interagir avec l'application avec des filtres pour actualiser des tableaux et visualisations
- au moins 4 types de graphiques différents (ex : histogramme, boîte à moustache, diagramme, nuage de point)



Intermédiaire

- Fonctionnalités Standard
- Bouton pour exporter les graphiques en .png
- Une page "Prédiction" pour prédire l'étiquette DPE et la consommation
- Enrichir l'apprentissage du modèle avec des données d'OpenData
- Déploiement de l'application sur le web



Expert

- Fonctionnalités Intermédiaire
- L'utilisateur peut rafraîchir les données actuelles avec les nouvelles via l'API
- L'utilisateur peut réentrainer le modèle à partir de nouvelle données
- Les modèles sont accessibles via une API (Application Programming Interface)
- Conteneuriser l'application dans une image docker



Les étapes



Rappels du langage Python



Création d'un repos GitHub et d'un environnement virtuel Python



Construction de visuels et cartographie



Extraction des données du projet avec l'API



Modèle de classification des Etiquettes



Modèle de régression pour prédire la consommation



Développement des composants de l'application Dash



Finalisation du modèle et développement de la page "prédiction" de l'application



Implémentation du module de réentrainement dans l'application et le développement de l'API



LIENS UTILES

- Cours Kaggle : <https://www.kaggle.com/learn>
- Crédation de l'environnement virtuel : <https://python.land/virtual-environments/virtualenv>
- Crédation d'une API : <https://towardsdatascience.com/deploy-a-machine-learning-model-using-flask-da580f84e60c>
- Déployer l'application : <https://render.com/> OU <https://shiny.posit.co/py/docs/deploy.html>
- Dessiner des schémas : <https://www.drawio.com/>
- Apprendre Shiny : <https://shiny.posit.co/py/>
- Cours de Python : <https://asardell.github.io/programmation-python/>
- Exemple de graphique en Python : <https://python-graph-gallery.com/>
- Contexte du Challenge ENEDIS : <https://defis.data.gouv.fr/defis/65b76f15d7874915c8e41298>
- Initiation au RMarkdown : <https://rmarkdown.rstudio.com/lesson-1.html>
- Initiation à GitHub Desktop : <https://docs.github.com/fr/desktop/overview/>
- API des logements neufs : <https://data.ademe.fr/datasets/dpe-v2-logements-neufs/api-doc>
- API des logements existants : <https://data.ademe.fr/datasets/dpe-v2-logements-existants/api-doc>
- API - GET query with range parameters from ElasticSearch :
https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html#_ranges
- Base Adresse Nationale au format .csv : <https://adresse.data.gouv.fr/donnees-nationales>

Cahier des charges 1/2

-  : Non réalisé
-  : Inachevé
-  : Réalisé



Rapport d'étude Markdown (6 pages max)

Mise en forme, rédaction et export au format markdown

KPI pertinents

Méthodologie, métriques et interprétation des modèles

Repository GitHub

Création d'un repository public appelé "m2_enedis"

Tous les scripts (avec commentaires) utilisés dans le projet

Un fichier README.md pour présenter les informations importantes du repository

Une documentation technique de l'application en markdown (2 pages max) :

- avec un schéma de l'architecture
- qui explique comment installer l'application sur son poste avec docker
- qui présente les packages nécessaires
- mise en forme, rédaction

Une documentation fonctionnelle de l'application en markdown (2 pages max) :

- qui présente l'intérêt de chaque page de l'application
- qui présente les fonctionnalités majeures de l'application
- mise en forme, rédaction

Captation vidéo de l'application (5 min max)

La vidéo explique comment installer l'application en local et présente les fonctionnalités majeures de l'application

Cahier des charges 2/2

- ✗ : Non réalisé
- 🚧 : Inachevé
- ✓ : Fonctionnel



Pack "Standard"

L'application dispose d'au moins 3 onglets/pages différentes

Il y a des images et des icônes dans l'application

L'application dispose d'au moins une cartographie interactive avec des markers

Il y a une page "Contexte" qui permet de présenter et visualiser les données disponibles

L'utilisateur peut interagir avec l'application et les données via les widgets : select, checkbox, sliders et radio buttons

L'utilisateur peut filtrer dynamiquement les données pour actualiser des tableaux et visualisations

L'application propose plusieurs KPI et au moins 4 types de graphiques différents

Suivi du projet et des tickets sur Azure DevOps

Pack "Intermédiaire"

Il y a des boutons pour exporter les graphiques en .png et exporter les données sélectionnées en .csv

Une page "prédiction" pour prédire l'étiquette DPE et la consommation énergétique

L'application est déployée sur internet et disponible via son url

Enrichir l'apprentissage du modèle avec des données d'OpenData (ex : température)

Pack "Expert"

L'utilisateur peut rafraîchir les données actuelles avec les nouvelles via l'API et la date de réception du DPE

L'utilisateur peut réentraîner le modèle à partir de nouvelles données

Les modèles sont accessibles via une API (Application Programming Interface)

Conteneurisation de l'application dans une image docker stockée sur le repos

THANK YOU