

# Projet Python : Développement d'une WebApp de Prédiction avec Streamlit

## Objectif

Vous devez réaliser une **application web en Python**, en local avec **Streamlit**, capable de :

- **Faire une prédiction automatique** grâce à un **modèle de machine learning** que vous aurez entraîné
- **Recevoir des données utilisateurs** via des champs interactifs (listes déroulantes, champs de texte, etc.)
- **Afficher le résultat de manière claire et intuitive**

Un **travail personnel** et **cohérent** sera attendu. Ce projet vous permettra d'apprendre à utiliser Python dans un environnement plus complet, intégrant **le traitement des données, l'entraînement de modèles et le développement d'applications web simples**.

## Consignes détaillées

### 1. Choix du Jeu de Données

- Choisissez **un dataset libre** (Kaggle, OpenML, etc.) **sur un sujet qui vous intéresse** (sport, santé, environnement, immobilier, finance, etc.).
- Votre projet doit être **un problème de prédiction supervisée** :
  - **Classification** (*exemple : prédire si un client achète ou pas*),
  - ou **Régression** (*exemple : prédire le prix d'une maison*).

#### ⚠ Attention :

Le choix du problème doit être **pertinent** et **cohérent**.

Exemples :

- Prédire le prix d'une maison en fonction de ses caractéristiques ✓
- Prédire la surface d'une maison à partir de son prix ✗ (peu utile)

## 2. Modèle de Machine Learning

- Utilisez **scikit-learn** (ou Pytorch si vous souhaitez faire du deep learning) pour entraîner un modèle.
- Le modèle devra :
  - Être **entraîné (ou fine-tuné) sur votre dataset**,
  - Être **enregistré** (par exemple avec `joblib` ou `pickle`) pour être utilisé dans votre application.

## 3. Création de la WebApp (avec Streamlit)

Votre application Streamlit devra comporter :

- **Des inputs utilisateur** :
  - Des **listes déroulantes** pour les variables catégorielles,
  - Des **champs de texte** ou  **curseurs** pour les variables numériques,
- **Un bouton** pour lancer la prédiction,
- **L'affichage clair** du résultat de la prédiction.

### Conseil :

Proposez une **interface simple et ergonomique**.

## 4. Démonstration Vidéo

- Vous devrez réaliser une **capture d'écran vidéo** (2 à 4 minutes) :
  - Montrant **l'utilisation** de votre WebApp,
  - Présentant brièvement **les fonctionnalités** et **le but** du projet.
  - En plus de la capture d'écran, vous expliquerez à voix haute ce que vous faites durant votre démonstration pour aider à la compréhension.
- La vidéo servira de **preuve de bon fonctionnement** de votre application.

# Éléments attendus pour la remise

À rendre :

Sur Github

- Le **code source complet**
- Un **README** expliquant :
  - L'objectif de la webapp
  - Le choix du dataset,
  - Le choix du modèle,
  - Le fonctionnement global de l'application,

Sur un Drive (Google ou équivalent):

- Les **données** ou lien pour les télécharger.
- La **vidéo de démonstration**.

## Critères d'évaluation

Critères
Pertinence du choix de projet et des données
Qualité du modèle (préparation, choix, performance raisonnable)
Fonctionnement correct de la WebApp (inputs, prédition)
Qualité de l'interface et de l'expérience utilisateur
Clarté du README et de la vidéo de démonstration
Respect des consignes générales et rendu complet

## Remarques importantes

- **Date de rendu** mercredi 18 mai 22h

- Vous devez former des **groupes de 3 personnes** (seul 1 groupe est autorisé à avoir une composition différente - seul ou à deux).
- Suivi projets le 5 mai
- **Pas besoin de déployer en ligne** : l'application doit fonctionner **localement**.
- L'utilisation de ChatGPT ou d'autres assistants est évidemment nécessaire pour ce projet, **mais** la cohérence de l'ensemble et des méthodes utilisées dans votre code permettront de déduire votre degré de compréhension, qui sera bien-sûr évalué.