Projet Don de Sang

Aperçu Général

Le Projet Don de Sang est une application web développée avec le framework Django pour gérer une base de données de donneurs de sang. L'objectif est de faciliter le suivi des donneurs, d'évaluer leur éligibilité à donner du sang via un modèle d'apprentissage automatique, et de fournir des analyses statistiques sur les données collectées. L'interface utilisateur est moderne, responsive et suit un thème visuel cohérent basé sur une palette rouge et grise, avec des icônes Font Awesome pour une expérience utilisateur améliorée.

Objectifs

Centraliser les informations des donneurs (âge, genre, profession, localisation, santé, dons).
Prédire l'éligibilité des donneurs à l'aide d'un modèle machine learning.
Offrir une interface intuitive pour ajouter, modifier, supprimer et analyser les donneurs.
Intégrer la géolocalisation pour remplir automatiquement les champs de localisation.

Prérequis

Pour exécuter ce projet localement, vous devez disposer des éléments suivants :

Logiciels

Python : Version 3.9.6 ou supérieure (testé avec 3.9.6).
Django : Version 4.2.16.
Navigateur web : Chrome, Firefox, Edge ou Safari (pour tester l'interface).
Git : Pour cloner le projet (optionnel).

Dépendances Python

Installez les bibliothèques suivantes avec pip :

bash

Réduire

Envelopper

Copier

```
pip install django==4.2.16 pandas numpy scikit-learn joblib textblob requests
```

django : Framework web principal.
pandas : Manipulation des données pour la prédiction.
numpy : Calculs numériques.
scikit-learn : Modèle ML et LabelEncoder.
joblib : Chargement du modèle ML.
textblob : Analyse de sentiment (utilisé dans certaines vues analytiques).
requests : Appels API (si utilisé ailleurs).

Ressources externes

Font Awesome : Inclus via CDN dans base.html pour les icônes :

html

Réduire

Envelopper

Copier

API Nominatim : Utilisée pour la géolocalisation (aucune clé API requise, mais nécessite une connexion internet). Installation Suivez ces étapes pour configurer et lancer le projet localement :

Cloner le projet (si hébergé dans un dépôt Git) :
bash

Réduire

Envelopper

Copier
git clone <URL-du-dépôt>
cd Don
Sinon, téléchargez et décompressez le code source.
Créer un environnement virtuel (optionnel mais recommandé) :
bash

Réduire

Envelopper

Copier
python -m venv venv
source venv/bin/activate # Sur Windows : venv\Scripts\activate
Installer les dépendances :
bash

Réduire

Envelopper

Copier
pip install -r requirements.txt
Si vous n’avez pas de requirements.txt, utilisez la commande listée dans "Dépendances Python".
Configurer Django :
Ouvrez DonDeSang/settings.py et vérifiez :
DEBUG = True (pour le développement).
DATABASES (par défaut, SQLite est utilisé avec db.sqlite3).
Ajoutez vos applications dans INSTALLED_APPS si nécessaire (ex. 'campagne').
Appliquer les migrations :
bash

Réduire

Envelopper

Copier
python manage.py makemigrations
python manage.py migrate

Créer un superutilisateur :
bash

Réduire

Envelopper

Copier
python manage.py createsuperuser
Suivez les instructions pour définir un nom d'utilisateur, email et mot de passe.
Placer les fichiers ML :
Copiez eligibility_model.pkl dans campagne/ml/.
Si disponibles, ajoutez le_genre.pkl, le_profession.pkl, et le_quartier.pkl dans le même dossier. Sinon, les encodeurs seront initialisés dynamiquement.
Lancer le serveur :
bash

Réduire

Envelopper

Copier
python manage.py runserver
Accédez à http://127.0.0.1:8000/ dans votre navigateur.
Structure du projet
Voici la structure complète du projet :

text

Réduire

Envelopper

Copier
Don/
├── campagne/
│ ├── ml/
│ │ ├── eligibility_model.pkl # Modèle ML pour prédire l'éligibilité
│ │ ├── le_genre.pkl # Encodeur pour le genre (optionnel)
│ │ ├── le_profession.pkl # Encodeur pour la profession (optionnel)
│ │ └── le_quartier.pkl # Encodeur pour le quartier (optionnel)
│ ├── templates/
│ │ ├── add_donor.html # Ajouter un donneur
│ │ ├── confirm_delete.html # Confirmer la suppression
│ │ ├── donors.html # Liste des donneurs
│ │ ├── prediction.html # Prédiction d'éligibilité
│ │ ├── update_donor.html # Modifier un donneur
│ │ ├── base.html # Template de base
│ │ ├── dashboard.html # Tableau de bord (statistiques)

```
│ │ ├── geo.html # Analyse géographique
│ │ ├── health.html # Analyse santé
│ │ ├── profiling.html # Profilage des donneurs
│ │ ├── efficiency.html # Efficacité des campagnes
│ │ ├── loyalty.html # Fidélité des donneurs
│ │ └── sentiment.html # Analyse de sentiment
│ ├── migrations/ # Fichiers de migration Django
│ ├── __init__.py
│ ├── admin.py # Configuration de l'admin Django
│ ├── apps.py # Configuration de l'application
│ ├── models.py # Modèle Donor
│ ├── serializers.py # Sérialiseurs REST (API)
│ ├── tests.py # Tests unitaires
│ └── views.py # Logique des vues
├── DonDeSang/
│ ├── __init__.py
│ ├── asgi.py # Configuration ASGI (optionnel)
│ ├── settings.py # Configuration Django
│ ├── urls.py # Routage principal
│ └── wsgi.py # Configuration WSGI
├── static/ # Fichiers statiques (CSS, JS, images)
│ └── (à créer si nécessaire)
├── media/ # Fichiers uploadés
├── manage.py # Script de gestion Django
├── db.sqlite3 # Base de données SQLite
└── README.md # Ce fichier
```

Modèle de données

Le modèle principal est Donor (défini dans campagne/models.py). Voici ses champs probables (basés sur les templates) :

python

Réduire

Envelopper

Copier

```
from django.db import models
from django.contrib.auth.models import User

class Donor(models.Model):
user = models.ForeignKey(User, on_delete=models.CASCADE, null=True) # Utilisateur associé
date_remplissage = models.DateField(auto_now_add=True) # Date de création
date_naissance = models.DateField() # Date de naissance
genre = models.CharField(max_length=10, choices=[('Homme', 'Homme'), ('Femme', 'Femme')])
profession = models.CharField(max_length=100, blank=True)
arrondissement = models.CharField(max_length=100, blank=True)
```

quartier = models.CharField(max_length=100, blank=True)
latitude = models.FloatField(null=True, blank=True) # Coordonnées GPS
longitude = models.FloatField(null=True, blank=True)
hypertension = models.BooleanField(default=False) # Conditions de santé
diabete = models.BooleanField(default=False)
asthme = models.BooleanField(default=False)
hiv_hbs_hcv = models.BooleanField(default=False)
tatoue = models.BooleanField(default=False)
feedback = models.TextField(blank=True) # Commentaires
nombre_dons = models.PositiveIntegerField(default=0) # Nombre de dons

```
def age(self):
    from datetime import date
    today = date.today()
    return today.year - self.date_naissance.year - ((today.month, today.day) <
(self.date_naissance.month, self.date_naissance.day))

def __str__(self):
    return f"Donneur #{self.id} ({self.profession}, {self.quartier})"
```

Remarque : Ajustez ce modèle selon votre fichier models.py réel.

Vues et URLs
Voici les vues principales et leurs URLs associées (déduites des templates) :

views.py
python

Réduire

Envelopper

Copier
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from .models import Donor
import joblib
from sklearn.preprocessing import LabelEncoder
import pandas as pd

Initialisation des encodeurs ML

model = joblib.load('campagne/ml/eligibility_model.pkl')
le_genre, le_profession, le_quartier = LabelEncoder(), LabelEncoder(), LabelEncoder()

```
def initialize_encoders():
donors = Donor.objects.all()
le_genre.fit(list(set(d.genre for d in donors if d.genre)) + ['Inconnu'])
le_profession.fit(list(set(d.profession for d in donors if d.profession)) + ['Inconnu'])
le_quartier.fit(list(set(d.quartier for d in donors if d.quartier)) + ['Inconnu'])

initialize_encoders()

@login_required
def donors(request):
donor_list = Donor.objects.all()
return render(request, 'donors.html', {'donor_list': donor_list})

@login_required
def add_donor(request):
if request.method == 'POST':
donor = Donor(
user=request.user,
date_naissance=request.POST['date_naissance'],
genre=request.POST['genre'],
profession=request.POST['profession'],
arrondissement=request.POST['arrondissement'],
quartier=request.POST['quartier'],
latitude=request.POST.get('latitude'),
longitude=request.POST.get('longitude'),
hypertension='hypertension' in request.POST,
diabete='diabete' in request.POST,
asthme='asthme' in request.POST,
hiv_hbs_hcv='hiv_hbs_hcv' in request.POST,
tatoue='tatoue' in request.POST,
feedback=request.POST.get('feedback', ''),
nombre_dons=int(request.POST['nombre_dons'])
)
donor.save()
messages.success(request, "Donneur ajouté avec succès.")
return redirect('donors')
return render(request, 'add_donor.html')

@login_required
def update_donor(request, donor_id):
donor = get_object_or_404(Donor, id=donor_id)
if request.method == 'POST':
donor.date_naissance = request.POST.get('date_naissance', donor.date_naissance)
donor.genre = request.POST.get('genre', donor.genre)
donor.profession = request.POST.get('profession', donor.profession)
donor.arrondissement = request.POST.get('arrondissement', donor.arrondissement)
donor.quartier = request.POST.get('quartier', donor.quartier)
```

```
donor.hypertension = 'hypertension' in request.POST
donor.diabete = 'diabete' in request.POST
donor.asthme = 'asthme' in request.POST
donor.hiv_hbs_hcv = 'hiv_hbs_hcv' in request.POST
donor.tatoue = 'tatoue' in request.POST
donor.feedback = request.POST.get('feedback', donor.feedback)
donor.nombre_dons = int(request.POST.get('nombre_dons', donor.nombre_dons))
donor.save()
messages.success(request, f"Donneur #{donor_id} mis à jour.")
return redirect('donors')
return render(request, 'update_donor.html', {'donor': donor})

@login_required
def delete_donor(request, donor_id):
donor = get_object_or_404(Donor, id=donor_id)
if request.method == 'POST':
donor.delete()
messages.success(request, f"Donneur #{donor_id} supprimé.")
return redirect('donors')
return render(request, 'confirm_delete.html', {'donor': donor})

@login_required
def prediction(request):
# Logique de prédiction (extrait simplifié)
if request.method == 'POST':
data = {
'age': int(request.POST.get('age', 0)),
'genre': request.POST.get('genre', 'Inconnu'),
'profession': request.POST.get('profession', 'Inconnu'),
'quartier': 'Inconnu',
'hypertension': request.POST.get('hypertension') == 'on',
'diabete': request.POST.get('diabete') == 'on',
'asthme': request.POST.get('asthme') == 'on',
'hiv_hbs_hcv': request.POST.get('hiv_hbs_hcv') == 'on',
'tatoue': request.POST.get('tatoue') == 'on',
}
df = pd.DataFrame([data])
df['genre'] = le_genre.transform([df['genre'][0]])[0]
df['profession'] = le_profession.transform([df['profession'][0]])[0]
df['quartier'] = le_quartier.transform([df['quartier'][0]])[0]
prediction = model.predict(df)[0]
return render(request, 'prediction.html', {'prediction_result': {'ml': 'Éligible' if prediction else 'Non éligible'}})
return render(request, 'prediction.html')
urls.py (dans DonDeSang/urls.py)
python

Réduire
```

Envelopper

Copier

```
from django.contrib import admin
from django.urls import path
from campagne import views

urlpatterns = [
path('admin/', admin.site.urls),
path('donors/', views.donors, name='donors'),
path('add_donor/', views.add_donor, name='add_donor'),
path('update_donor/int:donor_id/', views.update_donor, name='update_donor'),
path('delete_donor/int:donor_id/', views.delete_donor, name='delete_donor'),
path('prediction/', views.prediction, name='prediction'),
]
```

Fonctionnalités détaillées

1. Liste des donneurs (donors.html)
   URL : /donors/
   Description : Affiche tous les donneurs dans une grille de cartes avec leurs détails.
   Champs affichés : ID, âge, genre, profession, quartier, conditions de santé, nombre de dons.
   Actions : Modifier, supprimer, ajouter un nouveau donneur.
2. Ajouter un donneur (add_donor.html)
   URL : /add_donor/
   Description : Formulaire avec géolocalisation automatique via JavaScript et Nominatim.
   Champs : Date de naissance, genre, profession, arrondissement, quartier, conditions de santé, feedback, nombre de dons.
3. Modifier un donneur (update_donor.html)
   URL : /update_donor/<donor_id>/
   Description : Formulaire pré-rempli pour éditer les informations d'un donneur.
   Champs : Identiques à add_donor.html.
4. Supprimer un donneur (confirm_delete.html)
   URL : /delete_donor/<donor_id>/
   Description : Page de confirmation avant suppression définitive.
   Champs affichés : ID, profession, quartier.
5. Prédiction d'éligibilité (prediction.html)
   URL : /prediction/
   Description : Prédit si un donneur est éligible en utilisant un modèle ML et une vérification manuelle.
   Entrées : Âge, genre, profession, quartier, conditions de santé.
   Design et CSS
   Palette de couleurs :
   --primary-color: #B22222 (rouge principal).
   --secondary-color: #8B0000 (rouge sombre).
   --background-light: #F8F9FA (fond clair).
   --text-dark: #333333 (texte principal).
   --text-muted: #6C757D (texte secondaire).

--border-light: #E9ECEF (bordures).
--warning-color: #DC3545 (avertissements).
Caractéristiques visuelles :
Cartes avec box-shadow et border-radius.
Transitions au survol (transform: translateY(-2px)).
Grilles responsives avec grid-template-columns.
Icônes Font Awesome pour chaque champ/action.
Responsive : Media queries pour tablettes (768px) et mobiles (576px).
Tests
Tests manuels
Liste des donneurs :
Visitez /donors/ et vérifiez que les cartes s'affichent.
Cliquez sur "Ajouter un Donneur" et remplissez le formulaire.
Ajout :
Testez la géolocalisation (autorisez l'accès dans le navigateur).
Modification :
Modifiez un donneur et vérifiez les changements dans /donors/.
Suppression :
Confirmez la suppression et assurez-vous que le donneur disparaît.
Prédiction :
Testez avec différentes combinaisons de données.
Tests unitaires
Ajoutez des tests dans campagne/tests.py (exemple) :

python

Réduire

Envelopper

Copier
from django.test import TestCase
from .models import Donor

class DonorTestCase(TestCase):
def setUp(self):
Donor.objects.create(
date_naissance='1990-01-01',
genre='Homme',
profession='Professeur',
quartier='Inconnu',
nombre_dons=2
)

```
def test_donor_creation(self):
    donor = Donor.objects.get(profession='Professeur')
    self.assertEqual(donor.genre, 'Homme')
```

Exécutez les tests :

bash

Réduire

Envelopper

Copier
python manage.py test
Déploiement
Pour un environnement de production :

Configurer settings.py :
DEBUG = False
ALLOWED_HOSTS = ['votre-domaine.com', 'votre-ip']
Utilisez PostgreSQL : pip install psycopg2
Collecter les fichiers statiques :
bash

Réduire

Envelopper

Copier
python manage.py collectstatic
Servir avec Gunicorn :
bash

Réduire

Envelopper

Copier
pip install gunicorn
gunicorn DonDeSang.wsgi:application --bind 0.0.0.0:8000
Configurer un reverse proxy (ex. Nginx) :
Consultez la documentation Django pour plus de détails.
Exemples d'utilisation
Ajouter un donneur
Accédez à /add_donor/.
Remplissez :
Date de naissance : 2000-05-15
Genre : Homme
Profession : Étudiant
Autorisez la géolocalisation.
Soumettez et vérifiez dans /donors/.
Prédire l'éligibilité

Accédez à /prediction/.
Entrez :
Âge : 25
Genre : Femme
Profession : Infirmière
Cochez hypertension.
Vérifiez le résultat (ex. "Non éligible").
Problèmes connus
Erreur NotFittedError :
Cause : Encodeurs ML non entraînés.
Solution : initialize_encoders() dans views.py.
Géolocalisation échoue :
Cause : Permission refusée ou réseau.
Solution : Alertes JavaScript avec valeurs par défaut.
Contribution
Forkez le projet.
Créez une branche :
bash

Réduire

Envelopper

Copier
git checkout -b feature/
Commitez vos changements :
bash

Réduire

Envelopper

Copier
git commit -m "Description des changements"
Soumettez une pull request.
Licence
Ce projet est sous licence MIT (ou choisissez une autre licence selon vos préférences).

Crédits
Développeur principal : CodeStorm
Assistance IA : Grok (xAI).
Technologies : Django, scikit-learn, Nominatim API, Font Awesome.