**Modele de Inteligența Artificială în Schimbarea Climatică**

Descrierea tehnologiilor folosite:

* **React**

React este o bibliotecă JavaScript foarte populară, folosită pentru dezvoltarea interfețelor utilizator la nivel de frontend. Creată de către Facebook, React a revoluționat modul în care dezvoltatorii construiesc aplicații web, oferindu-le posibilitatea de a crea interfețe interactive și eficiente.

Una dintre cele mai importante caracteristici ale React este conceptul de "componente". O componentă reprezintă o bucată de cod reutilizabilă, care poate fi compusă și combinată pentru a construi interfețe complexe. Fiecare componentă are propriul său comportament, starea și vizualizare, ceea ce facilitează dezvoltarea și întreținerea aplicațiilor.

React utilizează un model de actualizare eficient, numit "Virtual DOM" (Document Object Model virtual). Acesta este un reprezentare în memorie a structurii interfeței utilizator, care este actualizată doar în cazul modificărilor necesare, ceea ce duce la o performanță îmbunătățită a aplicațiilor. Astfel, React face schimburile minime în DOM real, ceea ce duce la o reprezentare rapidă și interactivă a interfeței.

React oferă, de asemenea, suport pentru un flux de date unidirecțional, cunoscut sub numele de "flux de date descendente". Aceasta înseamnă că datele sunt transmise de la componenta părinte la componenta copil, asigurând o structură clară și ușor de gestionat a aplicației.

Datorită popularității și comunității active de dezvoltatori, React dispune de o mulțime de resurse și biblioteci suplimentare care facilitează dezvoltarea aplicațiilor web. De asemenea, React poate fi utilizat în combinație cu alte tehnologii, cum ar fi Redux sau React Router, pentru a crea aplicații mai complexe și scalabile.

Cu toate acestea, este important de menționat că React este o bibliotecă de frontend, ceea ce înseamnă că se concentrează pe partea de vizualizare a aplicației. Pentru a dezvolta o aplicație completă, este necesară integrarea React cu un server și o bază de date, pentru a gestiona logica și stocarea datelor.

În concluzie, React este o bibliotecă puternică și flexibilă, care permite dezvoltatorilor să creeze interfețe interactive și performante pentru aplicații web. Cu conceptul său de componente și abordarea sa eficientă a actualizărilor DOM, React a devenit o alegere populară pentru dezvoltarea aplicațiilor moderne.

* **Flask**

Flask este un framework web minimalistic și flexibil pentru limbajul de programare Python. Creat de către Armin Ronacher, Flask este proiectat pentru a fi simplu de înțeles și de utilizat, în timp ce oferă totuși funcționalități puternice pentru construirea aplicațiilor web.

Una dintre caracteristicile cheie ale Flask este abordarea sa minimalistă. Framework-ul oferă un set de funcționalități de bază, permițând dezvoltatorilor să-și construiască propriul lor stack tehnologic și să-și personalizeze aplicațiile conform nevoilor lor specifice. Acest lucru face ca Flask să fie ușor de învățat și de utilizat, în special pentru cei care sunt noi în dezvoltarea web.

Flask utilizează conceptul de rute pentru a gestiona cererile primite de la clienți. Prin intermediul decoratorilor Python, dezvoltatorii pot defini rute pentru diferite URL-uri, specificând funcțiile care vor fi apelate atunci când se accesează acele URL-uri. Această abordare face ca definirea endpoint-urilor și gestionarea cererilor să fie simplă și intuitivă.

Descrierea etapelor necesare pentru integrare:

**Server**

1. Am încărcat modelul pe server, folosind un checkpoint*. Toarch.load* este o metoda care citește starea unui model (weights-urile), ce au fost salvate după antrenare prin metoda *torch.save(state, filename)*

model = TempChangeNN()  
FILENAME = '../notebooks/checkpoint.pth.tar'  
load\_checkpoint(torch.load(FILENAME), model)  
model.eval()

2. Am creat un singur endpoint necesar pentru a face predicții. Expune un serviciu de tip GET, primind ca parametrii 2 numere, ce reprezintă luna și anul pentru care se cere schimbarea de temperatură.

@app.route('/predict', methods=['GET'])  
def predict(): *# put application's code here* month = float(request.args.get('month')) - 1  
 year = float(request.args.get('year'))  
 min\_train\_input = 1961.  
 max\_train\_input = 2019.  
 year = (year - min\_train\_input) / (max\_train\_input - min\_train\_input)  
 x = torch.tensor([month, year ])  
 with torch.no\_grad():  
 out = model(x)  
 return str(out.numpy()[0])

**Client**

1. Avem o singură pagină, reprezentată de componenta App
2. const root = ReactDOM.*createRoot*(document.getElementById('root'));  
   root.render(  
    <React.StrictMode>  
    <App />  
    </React.StrictMode>  
   );
3. function *App*() {  
     
    const [month, setMonth] = *useState*(1)  
    const [year, setYear] = *useState*(1950)  
    const [prediction, setPrediction] = *useState*(undefined);  
     
    const handlePredictButton = () => {  
    axios.get(`${baseURL}?month=${month}&year=${year}`)  
    .then(response => setPrediction(response.data))  
    .catch(err => *alert*(err))  
    }... }
4. Utilizând funcția de mai sus facem un request la server, iar ca response primim numărul ce reprezinta schimbarea de temeperatură.
5. prediction ? (<div  
    style={{display: "flex", flexDirection: "column", justifyContent: "center", alignItems: "center"}}>  
     
    Temperature change for Romania is {prediction}  
    {  
    prediction < 0 ? (<img src={Romania\_COLD} width={"20%"} height={"20%"} alt={"COLD"}/>)  
    :  
    (<img src={Romania\_HOT} width={"20%"} height={"20%"} alt={"COLD"}/>)  
    }  
     
   </div>) : (<></>)
6. Dacă schimbarea de temperatură este negativă afișăm o imagine cu o hartă a României afectată de temperaturi joase, iar daca este pozitivă afișăm o hartă a unei canicule.

GitHubRepo: https://github.com/StefanButacu/Air\_Trees\_Water\_Animals

Server: <https://github.com/StefanButacu/Air_Trees_Water_Animals/tree/master/API>

Client: https://github.com/StefanButacu/Air\_Trees\_Water\_Animals/tree/master/my-app