import streamlit as st

import pandas as pd

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import train\_test\_split , GridSearchCV

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

import pickle

import base64

# Load your dataset or replace this with your data loading code

# Assuming your data is in a DataFrame called 'df'

file\_path = 'Datafile\_Ml\_1.csv'

df = pd.read\_csv(file\_path)

# Step 1: Data Preprocessing

# Select the features and target variable

selected\_features = ['secchi', 'do\_mg\_l\_s', 'temp\_s', 'sal\_ppt\_s', 'turb\_s', 'N/P']

X = df[selected\_features] # Independent variables

y = df['chla\_n'] # Target variable

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Step 2: Train a Random Forest Regressor model

rf\_model = RandomForestRegressor(random\_state=42)

rf\_model.fit(X\_train, y\_train)

feature\_names = X\_train.columns.tolist()

# Function to make predictions

def predict\_chla\_n(input\_data):

predictions = rf\_model.predict(input\_data)

return predictions

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Step 2: Train a Random Forest Regressor model

rf\_model = RandomForestRegressor(random\_state=42)

rf\_model.fit(X\_train, y\_train)

# Function to make predictions

def predict\_chla\_n(input\_data, model):

predictions = model.predict(input\_data)

return predictions

# Save the trained model to a file using pickle

with open("rf\_model.pkl", "wb") as model\_file:

pickle.dump(rf\_model, model\_file)

# Function to predict Chl-a for user data (multiple as csv)

def predict\_chla\_n\_multiple(user\_data, model):

# Ensure that the user data columns match the model's input features

expected\_columns = ['secchi', 'do\_mg\_l\_s', 'temp\_s', 'sal\_ppt\_s', 'turb\_s', 'N/P']

if not set(expected\_columns).issubset(user\_data.columns):

st.error("The uploaded CSV file does not have the expected columns.")

return None

# Make predictions using the loaded model

predictions = model.predict(user\_data[expected\_columns])

return predictions

import streamlit as st

# Set a background image

background\_image = "ImageApala.png"

background\_css = f"""

<style>

body {{

background-image: url("{background\_image}");

background-size: cover;

}}

</style>

"""

st.markdown(background\_css, unsafe\_allow\_html=True)

# Add your content here

st.sidebar.image("ImageApala.png", use\_column\_width=True)

st.sidebar.header("Chlorophyll-a Estimation Tool for Bay-Estuary")

st.sidebar.markdown("This tool was developed training Machine Learning models to allow user to estimate Chl-a levels in bay estuaries based on physical water quality parameters with or without meteorological parameters")

# Streamlit app

#st.sidebar.image()

#st.sidebar.image()

#st.title("Bay\_Estuary\_Chl\_A")

# Create two options for the user to choose estimation method

estimation\_method = st.radio('Estimate Chl-a with:', ['Physical Water Quality Parameters', 'Physical Water Quality & Meteorological Parameters'])

if estimation\_method == 'Physical Water Quality Parameters':

st.write("Please provide Physical Water Quality Parameters:")

# Create an empty table for user input

input\_data = []

for feature\_name in feature\_names:

user\_input = st.text\_input(f"Enter {feature\_name}:", key=feature\_name)

input\_data.append(user\_input)

if st.button("Estimate Chl-a"):

# Convert user input to a DataFrame

user\_input\_data = pd.DataFrame({feature\_names[i]: [input\_data[i]] for i in range(len(feature\_names))})

# After receiving input data, load the model

with open("rf\_model.pkl", "rb") as model\_file:

loaded\_model = pickle.load(model\_file)

# Call the prediction function

predictions = predict\_chla\_n(user\_input\_data, loaded\_model)

# Display predictions to the user

st.write("Predictions for entered data:")

st.write(predictions)

uploaded\_file = st.file\_uploader("Upload a CSV file", type=["csv"])

if uploaded\_file is not None:

# Read the uploaded CSV file

user\_data = pd.read\_csv(uploaded\_file)

# Make predictions using the loaded model

with open("rf\_model.pkl", "rb") as model\_file:

loaded\_model = pickle.load(model\_file)

predictions = predict\_chla\_n\_multiple(user\_data, loaded\_model)

# Add a new column for estimated Chl-a

user\_data['Estimated\_Chl-a'] = predictions

# Display predictions to the user

st.write("Predictions for uploaded data:")

st.write(user\_data)

# Option to download the CSV file with estimated Chl-a

csv\_file\_with\_predictions = user\_data.to\_csv(index=False)

b64 = base64.b64encode(csv\_file\_with\_predictions.encode()).decode()

st.markdown(f'\*\*[Download CSV with Estimated Chl-a](data:file/csv;base64,{b64})\*\*')

elif estimation\_method == 'Meteorological Parameters':

st.write("Please provide Meteorological Parameters:")

# Create an empty table for user input

input\_data = []

for feature\_name in feature\_names:

user\_input = st.text\_input(f"Enter {feature\_name}:", key=feature\_name)

input\_data.append(user\_input)

uploaded\_file = st.file\_uploader("Upload a CSV file", type=["csv"])

if uploaded\_file is not None:

# Read the uploaded CSV file

user\_data = pd.read\_csv(uploaded\_file)

# Make predictions using the loaded model

predictions = predict\_chla\_n(user\_data, loaded\_model)

# Add a new column for estimated Chl-a

user\_data['Estimated\_Chl-a'] = predictions

# Display predictions to the user

st.write("Predictions for uploaded data:")

st.write(user\_data)

# Option to download the CSV file with estimated Chl-a

csv\_file\_with\_predictions = user\_data.to\_csv(index=False)

b64 = base64.b64encode(csv\_file\_with\_predictions.encode()).decode()

st.markdown(f'\*\*[Download CSV with Estimated Chl-a](data:file/csv;base64,{b64})\*\*')

if st.button("Estimate Chl-a"):

# Convert user input to a DataFrame

user\_input\_data = pd.DataFrame({feature\_names[i]: [input\_data[i]] for i in range(len(feature\_names))})

# After receiving input data, load the model

with open("rf\_model.pkl", "rb") as model\_file:

loaded\_model = pickle.load(model\_file)

# Call the prediction function

predictions = predict\_chla\_n(user\_input\_data, loaded\_model)

# Display predictions to the user

st.write("Predictions for entered data:")

st.write(predictions)

# Optionally, you can add visualizations and additional information her