Water portability checker

Model_train.py:

```
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from imblearn.over_sampling import SMOTE
import pickle

df = pd.read_csv('dataset (1).csv')
df = df.fillna(df.mean())

X = df.drop('Potability', axis=1)
y = df['Potability']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, stratify=y, random_state=42
)

smote = SMOTE(random_state=42)
X_train, y_train = smote.fit_resample(X_train, y_train)
```

```
model = xgb.XGBClassifier(
   n_estimators=500,
   max_depth=10,
   learning_rate=0.01,
   subsample=0.8,
   colsample_bytree=0.8,
   use_label_encoder=False,
   eval_metric='logloss'
)

model.fit(X_train, y_train)

preds = model.predict(X_test)
acc = accuracy_score(y_test, preds)
print(f'Accuracy: {acc}')

with open('water_model.pkl', 'wb') as f:
   pickle.dump(model, f)

with open('scaler.pkl', 'wb') as f:
   pickle.dump(scaler, f)
```

Main.py:

```
from fastapi import FastAPI
from pydantic import BaseModel
```

```
import pickle
import numpy as np

with open('water_model.pkl', 'rb') as f:
    model = pickle.load(f)

with open('scaler.pkl', 'rb') as f:
    scaler = pickle.load(f)

app = FastAPI()

class WaterInput(BaseModel):
    ph: float
    Hardness: float
    Solids: float
    Chloramines: float
    Sulfate: float
    Conductivity: float
    Organic_carbon: float
    Trihalomethanes: float
    Turbidity: float

@app.post("/predict")
def predict(data: WaterInput):
    input_data = np.array([[
        data.ph, data.Hardness, data.Solids, data.Chloramines,
        data.Sulfate, data.Conductivity, data.Organic_carbon,
        data.Trihalomethanes, data.Turbidity
    ]])
    input_scaled = scaler.transform(input_data)
    pred = model.predict(input_scaled)[0]
    return {"Potable": bool(pred)}
```

App.py

```
import streamlit as st
import pandas as pd
import numpy as np
import pickle
import plotly.express as px

st.set_page_config(
    page_title="Water Quality Analysis",
    page_icon="💧",
    layout="wide"
)

st.markdown("""
<style>
    .main {
        padding: 20px;
    }
    .stButton>button {
        width: 100%;
        background-color: #4CAF50;
        color: white;
        height: 3em;
        margin-top: 20px;
        border-radius: 10px;
        font-size: 18px;
        font-weight: bold;
    }
    .stButton>button:hover {
        background-color: #45a049;
    }
    .stSlider {
        padding-top: 10px;
        padding-bottom: 10px;
    }
    .title {
        font-size: 48px;
        font-weight: bold;
        color: #1e88e5;
        text-align: center;
        margin-bottom: 30px;
        text-shadow: 2px 2px 4px rgba(0,0,0,0.1);
    }
    .subtitle {
        font-size: 24px;
        color: #424242;
```

```
        margin-bottom: 20px;
        text-align: center;
    }
    .result-box {
        padding: 20px;
        border-radius: 10px;
        margin: 20px 0;
        text-align: center;
        font-size: 24px;
        font-weight: bold;
    }
    .info-box {
        background-color: #f8f9fa;
        padding: 20px;
        border-radius: 10px;
        margin-bottom: 20px;
    }
</style>
""", unsafe_allow_html=True)

@st.cache_resource
def load_model():
    with open('water_model.pkl', 'rb') as f:
        model = pickle.load(f)
    with open('scaler.pkl', 'rb') as f:
        scaler = pickle.load(f)
    return model, scaler


st.markdown("<h1 class='title'>💧 Water Potability Analysis</h1>", unsafe_allow_html=True)
st.markdown("""
<div class='info-box'>
    <p class='subtitle'>
        Analyze your water quality parameters to determine if the water is safe for drinking.
        This AI-powered tool uses advanced machine learning to assess water potability based on various chemical and physical properties.
    </p>
</div>
""", unsafe_allow_html=True)


try:
    model, scaler = load_model()


    col1, col2, col3 = st.columns(3)


    with col1:
        st.markdown("<h3 style='text-align: center; color: #1e88e5;'>Chemical Properties</h3>", unsafe_allow_html=True)
        ph = st.slider('pH Level', min_value=0.0, max_value=14.0, value=7.0)
        sulfate = st.slider('Sulfate', min_value=0.0, max_value=500.0, value=250.0)
        chloramines = st.slider('Chloramines', min_value=0.0, max_value=15.0, value=7.0)


    with col2:
        st.markdown("<h3 style='text-align: center; color: #1e88e5;'>Physical Properties</h3>", unsafe_allow_html=True)
        hardness = st.slider('Hardness', min_value=0.0, max_value=500.0, value=200.0)
        solids = st.slider('Total Dissolved Solids', min_value=0.0, max_value=1000.0, value=500.0)
        conductivity = st.slider('Conductivity', min_value=0.0, max_value=1000.0, value=500.0)

    with col3:
        st.markdown("<h3 style='text-align: center; color: #1e88e5;'>Organic Properties</h3>", unsafe_allow_html=True)
        organic_carbon = st.slider('Organic Carbon', min_value=0.0, max_value=30.0, value=15.0)
        trihalomethanes = st.slider('Trihalomethanes', min_value=0.0, max_value=150.0, value=75.0)
        turbidity = st.slider('Turbidity', min_value=0.0, max_value=10.0, value=5.0)

    input_data = pd.DataFrame({
        'ph': [ph],
        'Hardness': [hardness],
        'Solids': [solids],
        'Chloramines': [chloramines],
        'Sulfate': [sulfate],
        'Conductivity': [conductivity],
        'Organic_carbon': [organic_carbon],
        'Trihalomethanes': [trihalomethanes],
        'Turbidity': [turbidity]
    })

    if st.button('Analyze Water Quality'):
```

```
with st.spinner('Analyzing water quality...'):
    input_scaled = scaler.transform(input_data)
    prediction = model.predict(input_scaled)
    probability = model.predict_proba(input_scaled)

    st.markdown("<br>", unsafe_allow_html=True)

    result_col, chart_col = st.columns(2)

    with result_col:
        if prediction[0] == 1:
            st.markdown(f"""
                <div class='result-box' style='background-color: #e8f5e9; color: #2e7d32;'>
                    🎉 POTABLE WATER<br>
                    Confidence: {probability[0][1]:.1%}
                </div>
            """, unsafe_allow_html=True)
        else:
            st.markdown(f"""
                <div class='result-box' style='background-color: #ffebee; color: #c62828;'>
                    ⚠️ NON-POTABLE WATER<br>
                    Confidence: {probability[0][0]:.1%}
                </div>
            """, unsafe_allow_html=True)

    with chart_col:
            fig = px.pie(values=[probability[0][0], probability[0][1]],
                       names=['Non-Potable', 'Potable'],
                       title='Prediction Confidence',
                       hole=0.6,
                       color_discrete_sequence=['#ef5350', '#66bb6a'])
            fig.update_layout(
                title_x=0.5,
                title_font_size=20,
                showlegend=True,
                legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="center", x=0.5)
            )
            st.plotly_chart(fig)

        st.markdown("<h3 style='text-align: center; color: #1e88e5;'>Water Quality Parameters</h3>", unsafe_allow_html=True)

        param_cols = st.columns(3)

        param_ranges = {
            'pH': {'min': 0, 'max': 14, 'optimal_min': 6.5, 'optimal_max': 8.5},
            'Hardness': {'min': 0, 'max': 500, 'optimal_min': 50, 'optimal_max': 300},
            'TDS': {'min': 0, 'max': 1000, 'optimal_min': 100, 'optimal_max': 500}
        }

        with param_cols[0]:
            fig = px.bar([{'Parameter': 'pH', 'Value': ph}],
                x='Parameter', y='Value',
                title='pH Level',
                color_discrete_sequence=['#1e88e5'])
        fig.add_hline(y=6.5, line_dash="dash", line_color="green", annotation_text="Min Safe")
        fig.add_hline(y=8.5, line_dash="dash", line_color="red", annotation_text="Max Safe")
        fig.update_layout(title_x=0.5)
        st.plotly_chart(fig)

    with param_cols[1]:
        fig = px.bar([{'Parameter': 'TDS', 'Value': solids}],
                x='Parameter', y='Value',
                title='Total Dissolved Solids',
                color_discrete_sequence=['#1e88e5'])
        fig.add_hline(y=500, line_dash="dash", line_color="red", annotation_text="Max Safe")
        fig.update_layout(title_x=0.5)
        st.plotly_chart(fig)

    with param_cols[2]:
        fig = px.bar([{'Parameter': 'Hardness', 'Value': hardness}],
                x='Parameter', y='Value',
                title='Water Hardness',
                color_discrete_sequence=['#1e88e5'])
        fig.add_hline(y=300, line_dash="dash", line_color="red", annotation_text="Max Safe")
        fig.update_layout(title_x=0.5)
        st.plotly_chart(fig)
```

```
except Exception as e:
    st.error(f"An error occurred: {str(e)}")
    st.info("Please make sure the model files (water_model.pkl and scaler.pkl) are present in the same directory.")
```