Project Name: Diabetes Prediction Web App

Description:

A Streamlit-based interactive web application to predict the likelihood of diabetes based on health input parameters. The app uses Logistic Regression, Decision Tree, and Random Forest models trained on a dataset (likely the PIMA Indian Diabetes dataset).

Features:

- User input for 8 clinical features.
- Model selection from 3 popular classifiers.
- Immediate diabetes prediction.
- Model performance metrics shown in the sidebar.
- Confusion matrix heatmap visualization.

Modules Required:

- streamlit
- scikit-learn
- matplotlib
- seaborn
- Custom module: backend.py with load_data(), train_models(), and predict().

```
import streamlit as st
from sklearn.metrics import accuracy_score, precision_score, confusion_matrix, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
from backend import load_data, train_models, predict
from sklearn.model_selection import train_test_split


def main():
    st.title('Diabetes Prediction Web App')
    st.markdown('Enter the patient details to predict the likelihood of Diabetes.')

    # Inputs for the model
    with st.form(key="input_form"):
```

```
    Pregnancies = st.number_input('Pregnancies', min_value=0)
    Glucose = st.number_input('Glucose Level', min_value=0)
    BloodPressure = st.number_input('Blood Pressure', min_value=0)
    SkinThickness = st.number_input('Skin Thickness', min_value=0)
    Insulin = st.number_input('Insulin Level', min_value=0)
    BMI = st.number_input('BMI', min_value=0.0, format="%.2f")
    DiabetesPedigreeFunction = st.number_input('Diabetes Pedigree Function', min_value=0.0, format="%.3f")
    Age = st.number_input('Age', min_value=0)

    model_choice = st.selectbox('Choose Model', ['Logistic Regression', 'Decision Tree', 'Random Forest'])

    submit_button = st.form_submit_button("Predict")

  if submit_button:
    # Load data and models only once
    X, Y, scaler = load_data()
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
    models = train_models(X_train, Y_train)

    input_data = [Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age]
    model = models[model_choice]
    prediction = predict(model, scaler, input_data)

    if prediction == 1:
      st.error('The person is likely to have Diabetes. 🚨')
    else:
      st.success('The person is unlikely to have Diabetes. ✅')

    # Displaying model performance in the sidebar
```

```
    st.sidebar.header('Model Performance')
    for name, model in models.items():
        y_pred = model.predict(X_test)
        acc_score = accuracy_score(Y_test, y_pred)
        prec_score = precision_score(Y_test, y_pred)
        roc_score = roc_auc_score(Y_test, y_pred)
        st.sidebar.write(f"{name} - Accuracy: {acc_score:.2f}, Precision: {prec_score:.2f}, ROC AUC: {roc_score:.2f}")

    # Display confusion matrix for the selected model
    st.subheader(f'Confusion Matrix for {model_choice}')
    cm = confusion_matrix(Y_test, selected_model.predict(X_test))
    fig, ax = plt.subplots()
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=ax)
    ax.set_xlabel('Predicted')
    ax.set_ylabel('Actual')
    st.pyplot(fig)
    plt.close(fig)

# Run the app
if __name__ == "__main__":
    main()
```