```
import streamlit as st import joblib import pandas as pd import numpy as np # Load the saved scaler and logistic regression model try: scaler = joblib.load('scaler.pkl') model = joblib.load('logistic_regression_model.pkl') except FileNotFoundError: st.error("Scaler or model file not found. Please ensure 'scaler.pkl' and 'logistic_regression_model.pkl' are in the same directory.") st.stop() # Set the title of the Streamlit application st.title("Customer Churn Prediction") st.write("Enter customer details to predict churn.") # Create input fields for each feature credit_score = st.number_input("Credit Score", min_value=350, max_value=850, value=650) country = st.selectbox("Country", ['France', 'Germany', 'Spain']) gender = st.selectbox("Gender", ['Female', 'Male']) age = st.number_input("Age", min_value=18, max_value=92, value=38) tenure = st.number_input("Tenure (years)", min_value=0, max_value=10, value=5) balance = st.number_input("Balance", min_value=0.0, value=70000.0) products_number = st.number_input("Number of Products", min_value=1, max_value=4, value=1) credit_card = st.selectbox("Has Credit Card?", [0, 1], format_func=lambda x: 'Yes' if x == 1 else 'No') active_member = st.selectbox("Is Active Member?", [0, 1], format_func=lambda x: 'Yes' if x == 1 else 'No') estimated_salary = st.number_input("Estimated Salary", min_value=0.0, value=100000.0) # Create a button to trigger the prediction if st.button("Predict Churn"): # Create a pandas DataFrame from the user input input_data = pd.DataFrame([[credit_score, country, gender, age, tenure, balance, products_number, credit_card, active_member, estimated_salary]], columns=['credit_score', 'country', 'gender', 'age', 'tenure', 'balance', 'products_number', 'credit_card', 'active_member', 'estimated_salary']) # Apply one-hot encoding to 'country' and 'gender' input_data = pd.get_dummies(input_data, columns=['country', 'gender'], drop_first=True) # Define the columns that were present during training, including the dummy variables training_columns = ['credit_score', 'age', 'tenure', 'balance', 'products_number', 'credit_card', 'active_member', 'estimated_salary', 'country_Germany', 'country_Spain', 'gender_Male'] # Reindex the input DataFrame to match the columns of the training data # Fill any missing columns with 0 input_data = input_data.reindex(columns=training_columns, fill_value=0) # Scale the preprocessed input data input_scaled = scaler.transform(input_data) # Make a prediction prediction = model.predict(input_scaled) prediction_proba = model.predict_proba(input_scaled)[:, 1] # Display the prediction result if prediction[0] == 1: st.error(f"The customer is likely to churn (Probability: {prediction_proba[0]:.2f})") else: st.success(f"The customer is unlikely to churn (Probability: {prediction_proba[0]:.2f})")
```