```
import streamlit as st
import pickle
import pandas as pd
import warnings

warnings.filterwarnings("ignore", category=
UserWarning, module='sklearn')
warnings.filterwarnings("ignore", category=
FutureWarning, module='sklearn')

st.header('Country Clustering Prediction')

# Load the models
with open('kmeans.pkl', 'rb') as f:
    kmeans = pickle.load(f)

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

# Load your existing dataset (a subset of it) for
batching
existing_data = pd.read_csv(r'C:\Users\Acer\
PycharmProjects\pythonProject4\env\Country-data.csv'
)  # Replace with your actual dataset path

# Ensure that the existing data only contains
numerical values
existing_data_numeric = existing_data.select_dtypes(
include=[float, int])

# Extract the country names
country_names = existing_data['country'].tolist()

# Create a dropdown for country selection
selected_country = st.selectbox('Select a country to
auto-fill fields:', country_names)

# Autofill the input fields based on the selected
country
if selected_country:
    country_data = existing_data[existing_data['
```

```
32 country'] == selected_country].iloc[0]
33     child_mort = st.number_input('Child Mortality',
   value=country_data['child_mort'])
34     exports = st.number_input('Exports', value=
   country_data['exports'])
35     health = st.number_input('Health Spending', value
   =country_data['health'])
36     imports = st.number_input('Imports', value=
   country_data['imports'])
37     income = st.number_input('Income', value=
   country_data['income'])
38     inflation = st.number_input('Inflation', value=
   country_data['inflation'])
39     life_expec = st.number_input('Life Expectancy',
   value=country_data['life_expec'])
40     total_fer = st.number_input('Total Fertility',
   value=country_data['total_fer'])
41     gdpp = st.number_input('GDP per capita', value=
   country_data['gdpp'])
42 else:
43     child_mort = st.number_input('Child Mortality')
44     exports = st.number_input('Exports')
45     health = st.number_input('Health Spending')
46     imports = st.number_input('Imports')
47     income = st.number_input('Income')
48     inflation = st.number_input('Inflation')
49     life_expec = st.number_input('Life Expectancy')
50     total_fer = st.number_input('Total Fertility')
51     gdpp = st.number_input('GDP per capita')
52
53 # Button to predict and visualize clusters
54 if st.button('Predict and Visualize Clusters'):
55     input_data = {
56         'child_mort': [child_mort],
57         'exports': [exports],
58         'health': [health],
59         'imports': [imports],
60         'income': [income],
61         'inflation': [inflation],
62         'life_expec': [life_expec],
63         'total_fer': [total_fer],
```

```
64          'gdpp': [gdpp]
65      }
66
67      input_df = pd.DataFrame(input_data)
68
69      # Combine input data with the existing dataset
70      combined_df = pd.concat([existing_data_numeric,
    input_df], ignore_index=True)
71
72      # Predict the cluster using hierarchical
    clustering (AgglomerativeClustering)
73      hierarchical_clusters = hierarchical.fit_predict
    (combined_df)
74      hierarchical_predicted_cluster =
    hierarchical_clusters[-1]  # The cluster of the new
    input
75
76      st.write(f'Predicted Hierarchical Cluster: {
    hierarchical_predicted_cluster}')
77
78      # Refit the KMeans model on the combined dataset
     to update the clusters
79      kmeans.fit(combined_df)
80      kmeans_clusters = kmeans.predict(combined_df)
81      kmeans_predicted_cluster = kmeans_clusters[-1]
    # The cluster of the new input
82
83      st.write(f'Predicted KMeans Cluster: {
    kmeans_predicted_cluster}')
84
```