```python

import streamlit as st
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# def is_valid_email(email):
#     allowed_domains = ["gmail.com", "cloudmail.com"]
#     return "@" in email and email.split("@")[-1] in allowed_domains

# # --- Login Page ---
# if "logged_in" not in st.session_state:
#     st.session_state.logged_in = False

# if not st.session_state.logged_in:
#     st.title("Login Page")
#     st.write("Please log in with a Gmail or Cloudmail email address to access the app.")

#     email = st.text_input("Enter your email address", key="email")

#     if st.button("Login"):
#         if is_valid_email(email):
#             st.session_state.logged_in = True
#             st.success("Login successful! Redirecting...")
#             st.experimental_rerun()
#         else:
#             st.error("Invalid email address. Only Gmail or Cloudmail addresses are allowed.")
# else:
#     # --- Main App ---
#     st.sidebar.button("Logout", on_click=lambda: st.session_state.update({"logged_in": False}))


# Set up page configuration
st.set_page_config(
    page_title="Weather Impact on Crop Yield",
    page_icon="🌾",
    layout="wide"
)

st.markdown(
    """
    <style>
    .stApp {
        background: url('https://media.istockphoto.com/id/965148388/photo/green-ripening-soybean-field-agricultural-landscape.jpg?s=612x612&w=0&k=20&c=cEVP3uj34-5obt-Jf_WI3O9qfP6tVrFaQIv1rBvvpzc=') no-repeat center center fixed;
        background-size: cover;
        background-blend-mode: multiply;
        background-color: rgba(0, 0, 0, 0.5); /* Darken with black at 50% opacity */

    }
    .sidebar .sidebar-content {
        background: rgba(300, 255, 255, 0.8); /* Light sidebar for readability */
    }
```

```
    </style>
    """,
    unsafe_allow_html=True
)

# Title and description
st.title("Weather Impact on Crop Yield")
st.markdown("""
This application allows you to:
- Explore weather data and crop yield
- Analyze the impact of temperature, humidity, and rainfall
- Predict future crop performance using machine learning
""")

# Sidebar for user inputs
st.sidebar.header("User Inputs")
uploaded_file = st.sidebar.file_uploader("Upload your dataset (CSV format)", type=["csv"])

if uploaded_file:
    # Load dataset
    df = pd.read_csv(uploaded_file)
    st.sidebar.success("File uploaded successfully!")

    # Show dataset preview
    st.subheader("Uploaded Dataset")
    st.dataframe(df)

    # Sidebar filters
    state = st.sidebar.selectbox("Select State", df['State'].unique())
    crop = st.sidebar.selectbox("Select Crop", df['Main_Crop'].unique())
    year = st.sidebar.slider("Select Year Range", int(df['Year'].min()), int(df['Year'].max()),
                             (int(df['Year'].min()), int(df['Year'].max())))

    # Filtered data
    filtered_data = df[
        (df['State'] == state) &
        (df['Main_Crop'] == crop) &
        (df['Year'] >= year[0]) &
        (df['Year'] <= year[1])
    ]

    st.subheader(f"Filtered Data for {crop} in {state} ({year[0]} - {year[1]})")
    st.dataframe(filtered_data)

    # Visualizations
    st.subheader("Crop Yield Analysis")

    # Bar plot for Temperature vs. Yield
    st.write("### Temperature Impact on Crop Yield")
    if not filtered_data.empty:
        fig, ax = plt.subplots(figsize=(10, 6))
        sns.barplot(
            data=filtered_data, x=filtered_data['Temperature_C'].round(2), y='Yield_ton_per_hectare',
            palette="coolwarm", hue=filtered_data['Temperature_C'].round(2), dodge=False, legend=False
        )
        plt.title(f"Impact of Temperature on {crop} Yield in {state}")
```

```
        plt.xlabel("Temperature (°C)")
        plt.ylabel("Yield (tons/hectare)")
        plt.legend([], [], frameon=False)  # Remove extra legend
        st.pyplot(fig)

    # Bar plot for Humidity vs. Yield
    st.write("### Humidity Impact on Crop Yield")
    if not filtered_data.empty:
        fig, ax = plt.subplots(figsize=(10, 6))
        sns.barplot(
            data=filtered_data, x=filtered_data['Humidity_%'].round(2), y='Yield_ton_per_hectare',
            palette="viridis", hue=filtered_data['Humidity_%'].round(2), dodge=False, legend=False
        )
        plt.title(f"Impact of Humidity on {crop} Yield in {state}")
        plt.xlabel("Humidity (%)")
        plt.ylabel("Yield (tons/hectare)")
        plt.legend([], [], frameon=False)  # Remove extra legend
        st.pyplot(fig)

    # Bar plot for Rainfall vs. Yield
    st.write("### Rainfall Impact on Crop Yield")
    if not filtered_data.empty:
        fig, ax = plt.subplots(figsize=(10, 6))
        sns.barplot(
            data=filtered_data, x=filtered_data['Rainfall_mm'].round(2), y='Yield_ton_per_hectare',
            palette="viridis", hue=filtered_data['Rainfall_mm'].round(2), dodge=False, legend=False
        )
        plt.title(f"Impact of Rainfall on {crop} Yield in {state}")
        plt.xlabel("Rainfall (mm)")
        plt.ylabel("Yield (tons/hectare)")
        plt.legend([], [], frameon=False)  # Remove extra legend
        st.pyplot(fig)

    # Machine Learning Prediction Section
    st.subheader("Future Crop Yield Prediction")
    st.write("""
    Predict crop yield based on weather conditions using a machine learning model.
    """)

    # ML Dataset Preparation
    features = ['Temperature_C', 'Rainfall_mm', 'Humidity_%']
    target = 'Yield_ton_per_hectare'

    if all(col in df.columns for col in features + [target]):
        # Split dataset into features and target
        X = df[features]
        y = df[target]

        # Split into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

        # Train a Random Forest Regressor
        model = RandomForestRegressor(random_state=42)
        model.fit(X_train, y_train)

```

```
        # # Predict and evaluate the model
        # y_pred = model.predict(X_test)
        # mse = mean_squared_error(y_test, y_pred)
        # r2 = r2_score(y_test, y_pred)

        # st.write(f"Model Performance:")
        # st.write(f"- Mean Squared Error: {mse:.2f}")
        # st.write(f"- R-Squared: {r2:.2f}")

        # User input for prediction
        st.write("### Predict Crop Yield")
        temp_input = st.number_input("Enter Temperature (°C)", min_value=0.0, max_value=50.0, value=30.0, step=0.1)
        rainfall_input = st.number_input("Enter Rainfall (mm)", min_value=0.0, max_value=2000.0, value=1000.0, step=10.0)
        humidity_input = st.number_input("Enter Humidity (%)", min_value=0.0, max_value=100.0, value=50.0, step=1.0)

        # Prediction
        user_input = pd.DataFrame({
            'Temperature_C': [temp_input],
            'Rainfall_mm': [rainfall_input],
            'Humidity_%': [humidity_input]
        })

        predicted_yield = model.predict(user_input)[0]
        st.write(f"### Predicted Crop Yield: {predicted_yield:.2f} tons/hectare")
    else:
        st.warning("The dataset must contain 'Temperature_C', 'Rainfall_mm', 'Humidity_%', and 'Yield_ton_per_hectare' columns.")

else:
    st.warning("Please upload a dataset to proceed.")

# Footer
st.markdown("---")
st.markdown("**Developed by [Rohit,Abhishek,Aman,Anurag]** | Weather Impact on Crop Yield Analysis Tool 🌾")

```