Crop Yield Prediction Project with Streamlit and XGBoost

This is a fully explained Python project file for interview/demo purposes.
It uses a trained XGBoost model to predict crop yield based on crop type, state, and year.

Libraries Used:

- **streamlit: for building the web interface**
- **pickle: to load the trained model and feature list**
- **pandas & numpy: for data manipulation**

Model: XGBoost Regressor

XGBoost (Extreme Gradient Boosting) is a powerful ensemble ML algorithm that builds multiple decision trees sequentially.
It minimizes error by focusing on the mistakes of previous trees using gradient descent.

Code Walkthrough:

```
# Crop Yield Prediction Web App using Streamlit and XGBoost

import streamlit as st                    # Web app framework
import pickle                             # To load saved model and features
import numpy as np                        # For numerical operations
import pandas as pd                       # For structured data handling

# Load the trained ML model and feature list
with open("xgb_crop_yield_model.pkl", "rb") as f:
    model = pickle.load(f)  # Trained XGBoost model

with open("features_list.pkl", "rb") as f:
    feature_list = pickle.load(f)  # One-hot encoded column names

# Extract state and crop names for dropdowns
states = sorted([col for col in feature_list if col.startswith("State_Name_")])
```

```
crops = sorted([col for col in feature_list if col.startswith("Crop_")])
years = list(range(1997, 2016))  # Crop Year range

# App UI
st.title("Crop Yield Prediction App")
st.write("Predict yield based on crop, state, and year using XGBoost.")

# Dropdown inputs
selected_state = st.selectbox("Select State", [s.replace("State_Name_", "") for s in states])
selected_crop = st.selectbox("Select Crop", [c.replace("Crop_", "") for c in crops])
selected_year = st.selectbox("Select Year", years)

# Create input vector of same shape as training data
input_data = pd.DataFrame([np.zeros(len(feature_list))], columns=feature_list)
input_data[f"State_Name_{selected_state}"] = 1
input_data[f"Crop_{selected_crop}"] = 1
input_data["Crop_Year"] = selected_year

# Predict using the model
log_pred = model.predict(input_data)[0]
predicted_yield = np.expm1(log_pred)  # Reverse log1p if used

# Output result
st.subheader("Predicted Yield")
st.success(f"Estimated Yield: {predicted_yield:.2f} units")
```