In [ ]:
```python
import streamlit as st
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error, r2_score

@st.cache_data
def load_data():
    try:
        st.write("Attempting to load dataset...")
        data = pd.read_csv(r'C:\Users\DELL\OneDrive\Desktop\Unified Mentor Project\Crop Production data.csv')
        st.write("Dataset loaded successfully")
        return data
    except Exception as e:
        st.error(f"Error loading data: {e}")
        return None

data = load_data()

if data is not None:
    try:
        st.write("Dataset preview:")
        st.write(data.head())

        st.write("Preprocessing data...")
        data = data.dropna()
        label_encoders = {}
        categorical_columns = ['State_Name', 'District_Name', 'Season', 'Crop']

        for col in categorical_columns:
            label_encoders[col] = LabelEncoder()
            data[col] = label_encoders[col].fit_transform(data[col])

        X = data.drop('Production', axis=1)
        y = data['Production']
        st.write("Data preprocessing complete")

        st.write("Splitting data 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)
        st.write("Data splitting complete")

        st.write("Performing hyperparameter optimization with Grid Search...")
        param_grid = {
            'n_estimators': [50, 100, 200],
            'max_depth': [None, 10, 20],
            'min_samples_split': [2, 5, 10]
        }

        model = RandomForestRegressor(random_state=42)
        grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='r2')
        grid_search.fit(X_train, y_train)
        st.write("Grid Search complete")

        best_params = grid_search.best_params_
        best_model = RandomForestRegressor(**best_params, random_state=42)

        st.write("Training the model with the best parameters...")
        best_model.fit(X_train, y_train)
        st.write("Model training complete")

        st.write("Model trained successfully with the best parameters:")
        st.write(best_params)

        st.write("Evaluating the model on the test set...")
        y_pred = best_model.predict(X_test)
        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)

        st.write(f'Test Set Mean Squared Error: {mse:.2f}')
        st.write(f'Test Set R-squared: {r2:.2f}')

        st.title('Live Crop Production Prediction')

        st.subheader('Test Set Predictions')
        results = pd.DataFrame({
            'Actual': y_test.reset_index(drop=True),
            'Predicted': y_pred
        })

        st.write("Test set predictions:")
        st.write(results)

        st.subheader('Predicted vs Actual')
        st.line_chart(results)

        st.sidebar.title('Make a Prediction')
        state = st.sidebar.selectbox('Select State', label_encoders['State_Name'].classes_)
        district = st.sidebar.selectbox('Select District', label_encoders['District_Name'].classes_)
        season = st.sidebar.selectbox('Select Season', label_encoders['Season'].classes_)
        crop = st.sidebar.selectbox('Select Crop', label_encoders['Crop'].classes_)
        area = st.sidebar.number_input('Enter Area')
        crop_year = st.sidebar.number_input('Enter Crop Year', min_value=int(data['Crop_Year'].min()), max_value=int(data['Crop_Year'].max()), step=1)

        input_data = pd.DataFrame({
            'State_Name': [state],
            'District_Name': [district],
            'Season': [season],
            'Crop': [crop],
            'Area': [area],
            'Crop_Year': [crop_year]
        })

        for col in categorical_columns:
            input_data[col] = label_encoders[col].transform(input_data[col])

        input_data = input_data[X_train.columns]

        st.sidebar.write("Encoded input data:")
        st.sidebar.write(input_data)

        st.write("Performing prediction on user input...")
        user_prediction = best_model.predict(input_data)

        st.sidebar.subheader('Prediction')
        st.sidebar.write(f"Predicted Production: {user_prediction[0]}")
    except Exception as e:
        st.error(f"An error occurred during preprocessing or model prediction: {e}")
else:
    st.error("Failed to load data. Please check the file path and format.")
```