In [ ]:

```python
import streamlit as st
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import altair as alt

if st.button('Clear Cache and Rerun'):
    st.cache_data.clear()
    st.experimental_rerun()

@st.cache_data
def load_data():
    try:
        st.write("Attempting to load dataset...")
        data = pd.read_csv('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:
    st.write("Dataset preview:")
    st.write(data.head())

    try:
        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("Training the model...")
        model = RandomForestRegressor(n_estimators=100, random_state=42)
        model.fit(X_train, y_train)
        st.write("Model training complete")

        st.title('Live Crop Production Prediction')

        st.subheader('Test Set Predictions')
        predictions = model.predict(X_test)
        results = pd.DataFrame({
            'index': range(len(y_test)),
            'Actual': y_test.reset_index(drop=True),
            'Predicted': predictions
        })

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

        st.subheader('Predicted vs Actual')

        melted_results = pd.melt(results, id_vars=['index'], value_vars=['Actual', 'Predicted'], var_name='Type', value_name='Value')

        chart = alt.Chart(melted_results).mark_line().encode(
            x='index',
            y=alt.Y('Value', scale=alt.Scale(zero=False)),
            color=alt.Color('Type', scale=alt.Scale(domain=['Actual', 'Predicted'], range=['orange', 'white']))
        ).properties(
            width=600,
            height=400
        ).interactive()

        st.altair_chart(chart)

        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.write("Performing prediction on user input...")
        user_prediction = 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.")
```