CROP PREDICTION SYSTEM

presented by:

Akash Vinod
Amith VA
Anna Geo
Albin Jickoo

Introduction

The Crop Prediction System predicts suitable crops using soil, weather, and historical data, helping farmers optimize yields and adopt sustainable practices.

objectives

1. Enhance Agricultural Decision-Making

Assist farmers in choosing the most suitable crops based on environmental and soil conditions.

2. Efficient Resource Utilization

Minimize resource wastage (water, fertilizer, seeds) by optimizing crop selection.

3. Boost Agricultural Productivity

Increase yield by providing data-driven crop recommendations.

objectives

4. Leverage Data for Accuracy

Use historical and real-time datasets (e.g., soil type, rainfall, temperature) to make precise predictions.

5. Support Sustainable Farming

Promote sustainable agricultural practices by aligning crop choices with ecological conditions.

6. Simplify User Experience

Design a system that is user-friendly for farmers with minimal technical expertise.

04

objectives

7. Improve Profitability

Help farmers maximize profits by predicting high-demand crops suited to the region.

8. Enable Real-Time Analysis

Incorporate live data updates (e.g., weather conditions) for dynamic recommendations.

9. Address Regional Challenges

Tackle issues such as drought, over-irrigation, or pest infestations through predictive insights.

10. Promote Smart Agriculture

Integrate the system into broader smart agriculture initiatives for long-term benefits.

DATASET

	A	B	C	D	E	F	G	H	I	J
1	Crop	Crop_Year	Season	State	Area	Production	Annual_Rainfall	Fertilizer	Pesticide	Yield
2	Arecanut	1997	Whole Yea	Assam	73814	56708	2051.4	7024878	22882.34	0.796087
3	Arhar/Tur	1997	Kharif	Assam	6637	4685	2051.4	631643.3	2057.47	0.710435
4	Castor see	1997	Kharif	Assam	796	22	2051.4	75755.32	246.76	0.238333
5	Coconut	1997	Whole Yea	Assam	19656	126905000	2051.4	1870662	6093.36	5238.052

LOADING AND EXPLORING CROP YIELD DATA

```
data=pd.read_csv('/content/crop_yield.csv')
print(data.head())
```

```
           Crop  Crop_Year       Season  State     Area  Production  \
0      Arecanut       1997  Whole Year   Assam  73814.0       56708
1     Arhar/Tur       1997  Kharif       Assam   6637.0        4685
2   Castor seed       1997  Kharif       Assam    796.0          22
3      Coconut        1997  Whole Year   Assam  19656.0   126905000
4  Cotton(lint)       1997  Kharif       Assam   1739.0         794

   Annual_Rainfall  Fertilizer  Pesticide         Yield
0           2051.4  7024878.38   22882.34      0.796087
1           2051.4   631643.29    2057.47      0.710435
2           2051.4    75755.32     246.76      0.238333
3           2051.4  1870661.52    6093.36  5238.051739
4           2051.4   165500.63     539.09      0.420909
```

IINSPECTNG THE LAST FEW ROWS OF DATA

```
print(data.tail())
```

```
                Crop  Crop_Year       Season                State      Area  \
19684  Small millets       1998  Kharif                Nagaland    4000.0
19685          Wheat       1998  Rabi                  Nagaland    1000.0
19686          Maize       1997  Kharif        Jammu and Kashmir  310883.0
19687           Rice       1997  Kharif        Jammu and Kashmir  275746.0
19688          Wheat       1997  Rabi          Jammu and Kashmir  239344.0

       Production  Annual_Rainfall   Fertilizer  Pesticide     Yield
19684        2000           1498.0    395200.00    1160.00  0.500000
19685        3000           1498.0     98800.00     290.00  3.000000
19686      440900           1356.2  29586735.11   96373.73  1.285000
19687        5488           1356.2  26242746.82   85481.26  0.016667
19688      392160           1356.2  22778368.48   74196.64  1.261818
```

```
[32] print(data.shape)
```

```
(19689, 10)
```

GETTING DETAILED INFORMATION

```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19689 entries, 0 to 19688
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Crop             19689 non-null  object 
 1   Crop_Year        19689 non-null  int64  
 2   Season           19689 non-null  object 
 3   State            19689 non-null  object 
 4   Area             19689 non-null  float64
 5   Production       19689 non-null  int64  
 6   Annual_Rainfall  19689 non-null  float64
 7   Fertilizer       19689 non-null  float64
 8   Pesticide        19689 non-null  float64
 9   Yield            19689 non-null  float64
dtypes: float64(5), int64(2), object(3)
memory usage: 1.5+ MB
None
```

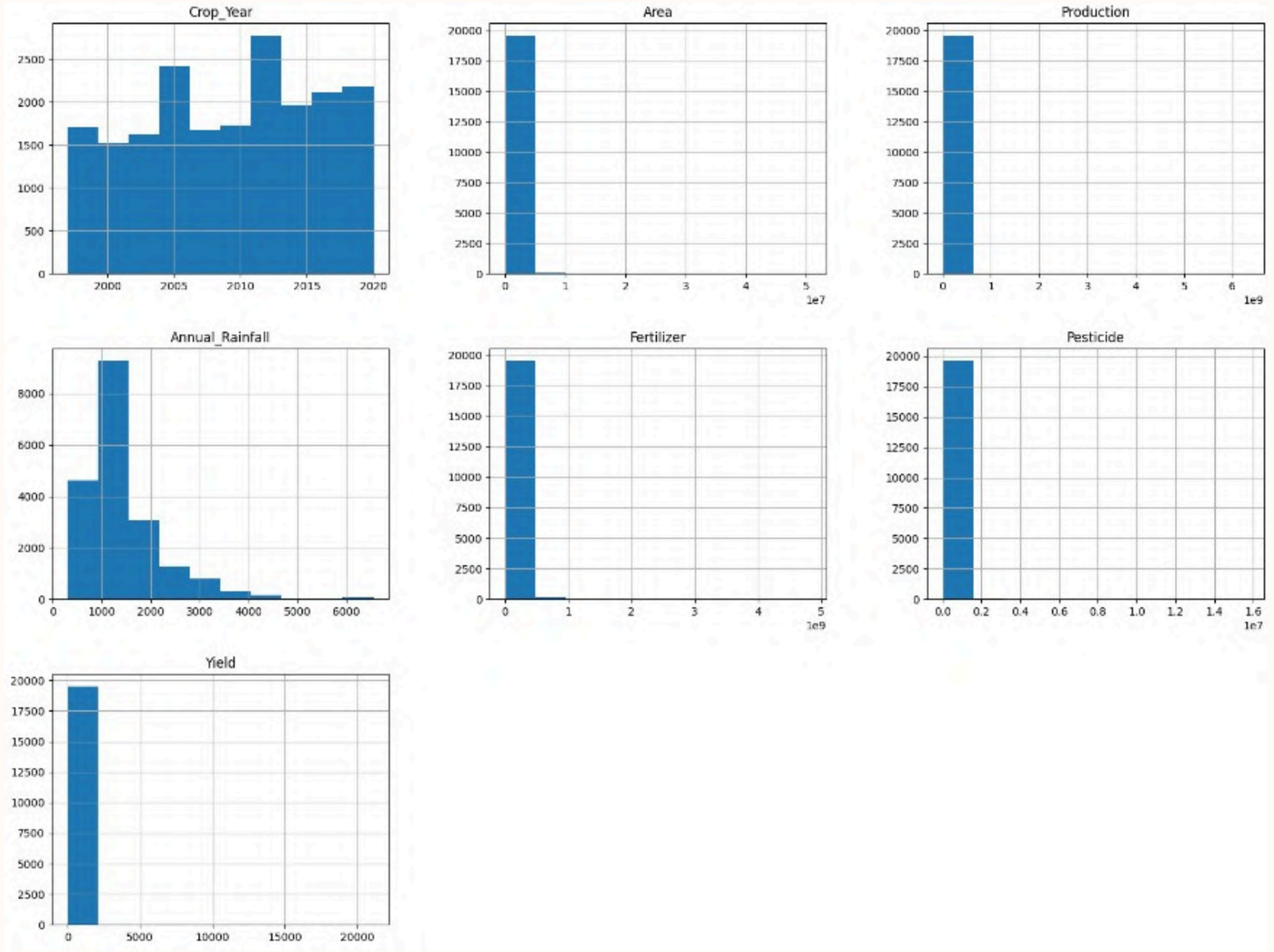
CHECKING FOR MISSING VALUES

```
print(data.isna().sum())
```

```
Crop                0
Crop_Year           0
Season              0
State               0
Area                0
Production          0
Annual_Rainfall     0
Fertilizer          0
Pesticide           0
Yield               0
dtype: int64
```

HISTOGRAM

```
import matplotlib.pyplot as plt
numeric_cols = data.select_dtypes(include=['int64', 'float64'])
numeric_cols.hist(figsize=(20, 15))
plt.show()
```


Crop_Year
Area
Production
Annual_Rainfall
Fertilizer
Pesticide
Yield
1e7
1e9
1e9
1e7

SCALING

```python
from sklearn.preprocessing import MinMaxScaler

# Assuming 'data' is your dataframe
scaler = MinMaxScaler()

# Select columns to scale based on your dataset
columns_to_scale = ['Area', 'Production', 'Annual_Rainfall', 'Fertilizer', 'Pesticide']

# Scale and transform the data
data[columns_to_scale] = scaler.fit_transform(data[columns_to_scale])

# View the scaled data
print(data.head())
```

```
           Crop  Crop_Year       Season  State      Area    Production  \
0      Arecanut       1997  Whole Year   Assam  0.001453  8.964274e-06
1     Arhar/Tur       1997  Kharif       Assam  0.000131  7.405944e-07
2   Castor seed       1997  Kharif       Assam  0.000016  3.477711e-09
3      Coconut        1997  Whole Year   Assam  0.000387  2.006086e-02
4  Cotton(lint)       1997  Kharif       Assam  0.000034  1.255138e-07

   Annual_Rainfall  Fertilizer  Pesticide        Yield
0         0.279953    0.001453   0.001453     0.796087
1         0.279953    0.000131   0.000131     0.710435
2         0.279953    0.000016   0.000016     0.238333
3         0.279953    0.000387   0.000387  5238.051739
4         0.279953    0.000034   0.000034     0.420909
```

ENCODING

```
[38] from sklearn.preprocessing import LabelEncoder

     # Create encoder instance
     encoder = LabelEncoder()

     # Specify categorical columns in your dataset
     categorical_cols = ['Crop', 'Season', 'State']  # Update with your actual categorical column names

     # Encode each categorical column
     for col in categorical_cols:
         data[col] = encoder.fit_transform(data[col])

     # View the transformed data
     print(data.head())
```

```
   Crop  Crop_Year  Season  State      Area    Production  Annual_Rainfall  \
0     0       1997       4      2  0.001453  8.964274e-06         0.279953
1     1       1997       1      2  0.000131  7.405944e-07         0.279953
2     8       1997       1      2  0.000016  3.477711e-09         0.279953
3     9       1997       4      2  0.000387  2.006086e-02         0.279953
4    11       1997       1      2  0.000034  1.255138e-07         0.279953

   Fertilizer  Pesticide        Yield
0    0.001453   0.001453     0.796087
1    0.000131   0.000131     0.710435
2    0.000016   0.000016     0.238333
3    0.000387   0.000387  5238.051739
4    0.000034   0.000034     0.420909
```

CREATING YIELD CATEGORIES

```
[44] # Create categories for Yield (e.g., low, medium, high)
     bins = [0, 2, 5, 10]  # Adjust based on the distribution of your target variable
     labels = ['Low', 'Medium', 'High']
     data['Yield_Category'] = pd.cut(data['Yield'], bins=bins, labels=labels)
```

MODEL TRAINING AND EVALUATION

```python
# Define features and target
X = data.drop(['Yield', 'Yield_Category'], axis=1)  # Features
y = data['Yield_Category']  # Target (categorical 'Yield_Category')

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

# Train the model
dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(X_train, y_train)

# Make predictions
y_pred = dt_classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.9310344827586207
              precision    recall  f1-score   support

        High       0.85      0.87      0.86       119
         Low       0.96      0.97      0.97      1387
      Medium       0.79      0.76      0.78       263

    accuracy                           0.93      1769
   macro avg       0.87      0.87      0.87      1769
weighted avg       0.93      0.93      0.93      1769
```

MAKING PREDICTIONS

```
new_input = pd.DataFrame({
    'Crop': [1],                # Encoded value for 'Wheat'
    'Crop_Year': [2022],        # Year of the crop
    'Production': [5000],       # Production value
    'Area': [73814],            # Area of the farm
    'Annual_Rainfall': [2051.4],  # Annual Rainfall
    'Fertilizer': [7024878],    # Fertilizer value
    'Pesticide': [22882.34],    # Pesticide value
    'Season': [1],              # Encoded value for 'Kharif'
    'State': [2]                # Encoded value for 'Maharashtra'
}, index=[0])  # Adding an index

# Get the columns used during training
training_columns = X_train.columns

# Reorder the new_input columns to match the training data
new_input = new_input[training_columns]

# Now, use new_input for predictions:
new_prediction = dt_classifier.predict(new_input)
print(f"Prediction for the new input: {new_prediction[0]}")
```

```
Prediction for the new input: Low
```

09

Conclusion

The Crop Prediction System uses AI and ML to help farmers select suitable crops, improve yields, and promote sustainable farming. It addresses agricultural challenges and supports climate-resilient practices, enhancing productivity and food security.

Thank You!