FEYNN LABS

CROP PREDICTION SYSTEM

BY

Uma Mahesh Yenamala

Emie Ann Varghese

Suvrojyoti Biswas

Rupal Meshram

Rehan Khan

Feasibility:

ML and DL techniques, like CNNs, are being successfully used for predicting the crop based on values of the Nitrogen, phosphorus, potassium, and pH of soil. For this to work, we need a lot of data that contains the values of Nitrogen, phosphorus, potassium, pH, humidity, and rainfall. There are now many online databases where we can find these values, and people are also helping by contributing to these collections. However, running these models requires powerful computers with special hardware like GPUs, and they also need to be set up on cloud platforms for them to work in real time. So, to make crop prediction with ML and DL feasible, we need good technology, lots of data, and the right hardware and software setup.

Viability:

ML and DL-based crop prediction to be viable, the systems need to accurately and reliably predict a crop, even in different environments. This accuracy ensures that farmers can trust the system's recommendations for crops effectively. Additionally, the system should be scalable to handle large deployments across different agricultural settings and adaptable to changes in prediction and crop varieties over time. User-friendly interfaces, such as easy-to-use dashboards and mobile apps, are crucial for farmers and agronomists to access and interact with the system effortlessly, enhancing its overall viability and usability.

Monetization:

To make money from crop prediction services, there are a few ways to do it. One way is to charge farmers a regular fee for using the service, kind of like a subscription. This fee would give them access to all the features, like predicting a crop based on soil and advice on how to deal with the problems. Another option is to charge farmers each time they use the service, like paying for each analysis or consultation. You could also offer extra-special features and support for a higher fee, giving farmers more advanced crop prediction and personalized advice. Lastly, by collecting and analyzing all the data from the soil with electronic devices, there's an opportunity to make money by sharing valuable insights with researchers, businesses, and government agencies who are interested in using the information for things like studying agriculture trends and cultivating a right crop for better profits based on soil.

1. Problem Statement :

Agriculture plays a crucial role in our economy, and farmers often face challenges in predicting the optimal crops to cultivate for a given season. Factors such as soil composition, weather conditions, and historical data influence crop yields. A crop prediction system can provide valuable insights to farmers, helping them make informed decisions about crop selection, resource allocation, and overall farm management.

2 Customer Need Assessment :

i. **Soil challenges**: Every crop requires specific nutrition in the soil. There are three main nutrients Nitrogen (N), Phosphorus (P), and Potassium (K) required in soil. The deficiency in nutrients can lead to poor quality of crops.

ii. **Climate challenges**: In agriculture, climatic factors such as rainfall, temperature, and humidity play an important role.

2.1. Market/Business Need Assessment :

In this assessment, we find out what the agriculture market needs. It defines the gaps that are preventing agriculture from reaching its desired goals. It also contains the strategy to make this business perfect or up to the mark

1) Recommend the type of crop the customer can cultivate that would best suit the respective conditions.
2) Recommend the type of fertilizer best suited for the particular soil and the recommended crop.

3. Target Specification :

Using the problem statement and the knowledge gathered from the customer needs, this system/service will provide them with some techniques so that the quality of the soil can be improved and it would give better yields of crops and even to some new customers/farmers who want to start with and if they might not have the idea of which crop to sow. The service which will be provided here can be beneficial for them as after testing the soil they could know which crop will be best to grow.

4. External Search

This section includes information gathered from numerous sources about the design problem and the product, process, or system that is the center of the design problem.

1) https://www.irjet.net/archives/V7/i2/IRJET-V7I2163.pdf

2) https://www.mdpi.com/2076-3417/13/16/9288

3) https://www.javatpoint.com/crop-yield-prediction-using-machine-learning

4) https://ieeexplore.ieee.org/abstract/document/9987366

5. Benchmarking

The comparison table between services in agriculture with or without machine learning.

1) **Soil Composition:**

   Without ML: In the early days farmers didn't know the importance of the composition of soil in crop harvesting due to which the harvest doesn't give more profit.

   With ML: Nowadays with the help of ML after testing the soil the machine learning models will tell us which crop to harvest according to our soil so there are more chances of high profit.

2) **Quality of Soil:**

   Without ML: Farmers don't have the chance to know which nutrient is less in the soil due to which the crop is damaging.

   With ML: But after testing we can know which nutrient to add to make our yield better.

6. Applicable Patents

1) Predictive models:

   Patents related to novel machine learning models and algorithms specifically designed for crop prediction.

2) Data Integration and Feature Engineering:

   Patents that focus on methods for integrating diverse data sources, such as soil data, weather data, satellite imagery, and historical crop yield data. This may also include inventive approaches to feature engineering for better model performance.

7. Applicable Regulations

1) Environmental:

If the system has environmental sustainability or involved the use of data related to soil health, water management, or other environmental regulations is important.

2) Intellectual property:

We will have to ensure that the technology we are using in our system does not infringe on existing patents or intellectual property rights.

3) Cybersecurity:

The sensitivity of agricultural data, compliance with cybersecurity regulations is important to protect against data breaches and ensure the confidentiality of the information.

8. Applicable Constraints

1) Limited availability or poor quality of data, especially in certain regions or for specific crops.

2) Varying levels of technological literacy among farmers.

3) Financial constraints among farmers who may be unwilling or unable to invest in new technologies.

4) Concerns about data security and privacy.

9. Business Model

1) Crop prediction as a service:

Offer a subscription-based service that provides farmers with accurate crop yield predictions, personalized recommendations, and decision support tools.

2) Customized solutions for different crops:

Specialize in providing crop prediction system tailored for specific crops, considering the unique requirements of each crop type.

3) Mobile apps for customers:

Develop user-friendly mobile applications that deliver crop predictions, weather forecasts, and actionable insights to farmers.

4) Data analytics and insight services:

Offer data analytics services to analyze and interpret agricultural data, providing valuable insights to farmers, agribusiness, and government agencies.

5) Government and NGO's Partnership:

Collaborate with government agencies and non-governmental organizations to deploy crop prediction systems for wider adoption.

6) Weather Risk Insurance :

Partner with insurance companies to develop weather risk insurance products based on accurate crop predictions.

7) Collaboration with Agribusiness:

Collaborate with agribusinesses, cooperatives, and supply chain stakeholders to integrate crop prediction systems into their operations.

10. Concept Generation

Throughout the concept generation process, it's essential to prioritize user needs, feasibility, and market viability. Collaborate with stakeholders, gather insights, and iterate on concepts to develop a robust foundation for the Crop Prediction System.

i) Divergent thinking techniques:

Divergent thinking techniques such as mapping, brainstorming, and lateral thinking can be used to explore a wide range of ideas. Encouraging participants to think beyond conventional solutions.

ii) Cross-industry inspiration:

We can also look for inspiration other than the agriculture domain. Exploring concepts and technologies from other industries that can be adapted or applied to improve crop prediction systems.

iii) Emerging technologies:

Consider emerging technologies such as blockchain, edge computing, or advanced sensors. Explore how these technologies could enhance the accuracy and efficiency of the Crop Prediction System.

iv) Data integration strategies:

Explore different strategies for integrating diverse data sources, including satellite imagery, soil sensors, weather data, and historical records. Consider innovative approaches for handling and analyzing big data.

v) Feedback loops:

Implement feedback loops to continuously improve the system. Explore concepts for gathering user feedback, monitoring system performance, and adapting to changing agricultural dynamics.

vi) Cost-effective solution:

Develop concepts that are cost-effective for farmers. Explore innovative business models, subscription plans, or partnerships to make the system financially accessible.

11. Concept Development

Concept development involves refining and elaborating on the ideas generated during the concept generation phase. This phase aims to turn promising concepts into well-defined and detailed proposals. Throughout the concept development phase, collaboration among interdisciplinary teams, including developers, designers, domain experts, and potential users, is crucial. Regular reviews, feedback loops, and a commitment to user-centric design principles contribute to the successful refinement and development of the Crop Prediction System.

Deliverables:

1) A functional Crop Yield Prediction System with a user interface.

2) Documentation outlining the model architecture, data sources, and instructions for users.

3) Training materials to educate farmers on interpreting predictions and utilizing the system effectively.

Success Criteria:

1) The system should demonstrate accurate predictions, with validation metrics meeting predefined thresholds.

2) Positive feedback and adoption from farmers in the target region.

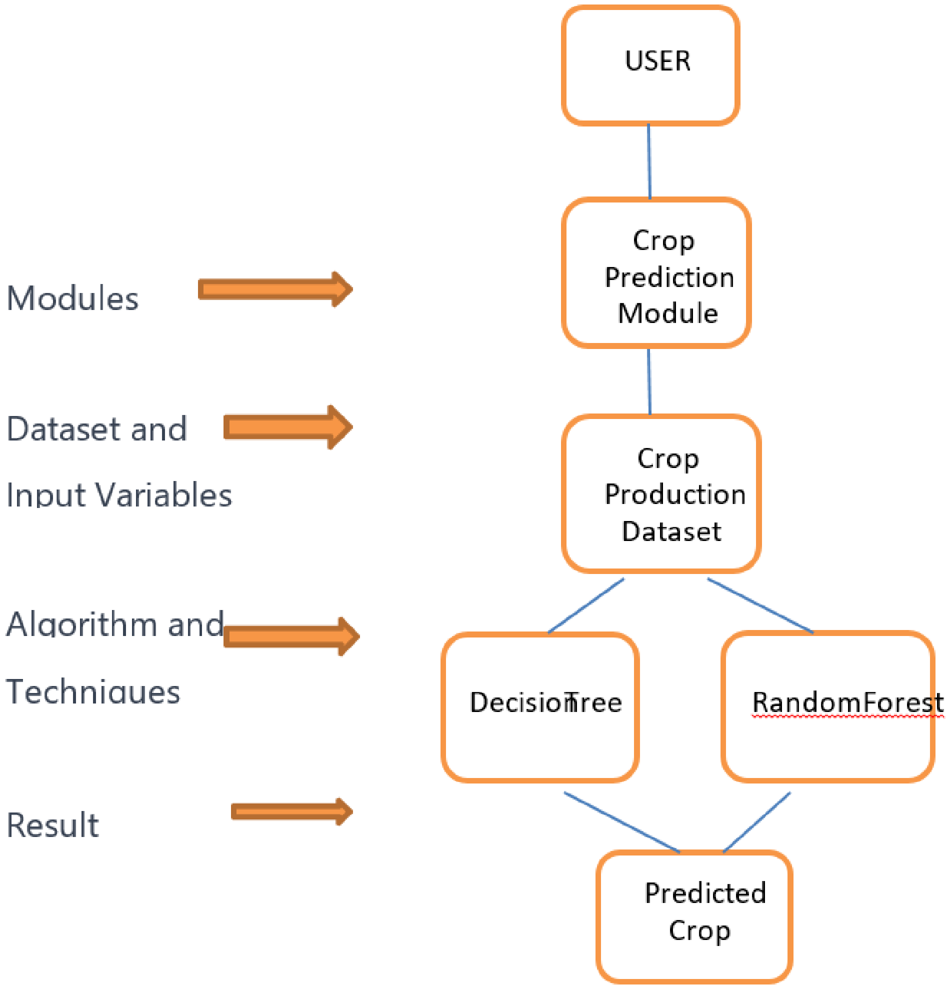
3) Improved decision-making and resource utilization by farmers based on the predictions provided by the system

12. Final Product Prototype

Abstract description:

Agriculture plays a crucial role in our economy, and farmers often face challenges in predicting the optimal crops to cultivate for a given season. Factors such as soil composition, weather conditions, and historical data influence crop yields. A crop prediction system can provide valuable insights to farmers, helping them make informed decisions about crop selection, resource allocation, and overall farm management. Developing a machine learning-based Crop Yield Prediction System that predicts the expected yield of various crops based on historical data, soil characteristics, and weather conditions.

Schematic Diagram Overview:



13. Product Details

How It Works?

1) Data Input: Farmers input the data of the soil nutrients, temperature, and rainfall and pH level of soil in the form which will be displayed in web interface.

2) Crop Prediction: After filling out all the information when we click on the predict button our system provides the result of which crop would be useful to harvest.

Frontend Development:

1) Design: Simple and very easy to fill information in the form.

2) Technologies: Built with HTML, CSS, and JavaScript.

Backend Development:

A lot of manual supervised machine learning has been performed to optimize the automated tasks.

1) Data Collection:

Gather historical data on crop yields, soil composition, and weather conditions for the target region. This data will be used to train and validate the machine-learning models.

2) Feature engineering:

Identify relevant features that influence crop yields, such as soil nutrients,
temperature, rainfall, humidity, and other environmental factors

3) Model Development:

Build machine learning models (e.g., regression models, ensemble methods) to predict crop yields based on the selected features.

Market Segmentation Analysis

In [1]:
```
import pandas as pd
df = pd.read_csv("indiancrop_dataset.csv")
df.head()
```

Out[1]:

	N_SOIL	P_SOIL	K_SOIL	TEMPERATURE	HUMIDITY	ph	RAINFALL	STATE	CROP_PRICE	CROP
0	90	42	43	20.879744	82.002744	6.502985	202.935536	Andaman and Nicobar	7000	Rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	Andaman and Nicobar	5000	Rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	Andaman and Nicobar	7000	Rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	Andaman and Nicobar	7000	Rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	Andaman and Nicobar	120000	Rice

In [2]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   N_SOIL       2200 non-null   int64
 1   P_SOIL       2200 non-null   int64
 2   K_SOIL       2200 non-null   int64
 3   TEMPERATURE  2200 non-null   float64
 4   HUMIDITY     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   RAINFALL     2200 non-null   float64
 7   STATE        2200 non-null   object
 8   CROP_PRICE   2200 non-null   int64
 9   CROP         2200 non-null   object
dtypes: float64(4), int64(4), object(2)
memory usage: 172.0+ KB
```


In [3]:
```
df.isnull().sum()
```

Out[3]:
```
N_SOIL         0
P_SOIL         0
K_SOIL         0
TEMPERATURE    0
HUMIDITY       0
ph             0
RAINFALL       0
STATE          0
CROP_PRICE     0
CROP           0
dtype: int64
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['N_SOIL', 'P_SOIL', 'K_SOIL', 'TEMPERATURE', 'HUMIDITY', 'ph',
       'RAINFALL', 'STATE', 'CROP_PRICE', 'CROP'],
      dtype='object')
```

In [5]:
```
df.shape
```

Out[5]:
```
(2200, 10)
```

In [6]:
```
df.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
N_SOIL	2200.0	50.551818	36.917334	0.000000	21.000000	37.000000	84.250000	140.000000
P_SOIL	2200.0	53.362727	32.985883	5.000000	28.000000	51.000000	68.000000	145.000000
K_SOIL	2200.0	48.149091	50.647931	5.000000	20.000000	32.000000	49.000000	205.000000
TEMPERATURE	2200.0	25.616244	5.063749	8.825675	22.769375	25.598693	28.561654	43.675493
HUMIDITY	2200.0	71.481779	22.263812	14.258040	60.261953	80.473146	89.948771	99.981876
ph	2200.0	6.469480	0.773938	3.504752	5.971693	6.425045	6.923643	9.935091
RAINFALL	2200.0	103.463655	54.958389	20.211267	64.551686	94.867624	124.267508	298.560117
CROP_PRICE	2200.0	2689.228182	3710.361267	2.000000	950.000000	1825.000000	3500.000000	120000.000000

In [8]:
```
import pandas as pd

# Assuming you already have a DataFrame named 'df' with integer columns
def check_outliers(df):
  int_cols = df.select_dtypes(include="int")
  outliers_info = pd.DataFrame(columns=["Column", "outlier vals","Outlier Count"])

  q1 = int_cols.quantile(0.25)
  q3 = int_cols.quantile(0.75)
  outlier_columns =[]

  iqr = q3 - q1
  upper_limit = q3 + (1.5 * iqr)
  lower_limit = q1 - (1.5 * iqr)
  print(lower_limit)

  for col in int_cols.columns:
      # Check for outliers in each column
      outlier_vals = ((df[col] < lower_limit[col]) | (df[col] > upper_limit[col]))
      outlier_count = ((df[col] < lower_limit[col]) | (df[col] > upper_limit[col])).sum()

      # If there are outliers, add the column and count to the DataFrame
      if outlier_count > 0:
          outlier_columns.append(col)
          outliers_info = outliers_info.append({"Column": col, "Outlier Count": outlier_count,"outlier vals": outlier_vals}, ignc

  # Display DataFrame with columns containing outliers and their counts

  #print("columns with outliers = ",outlier_columns)
  return outlier_columns,outliers_info,lower_limit,upper_limit
```

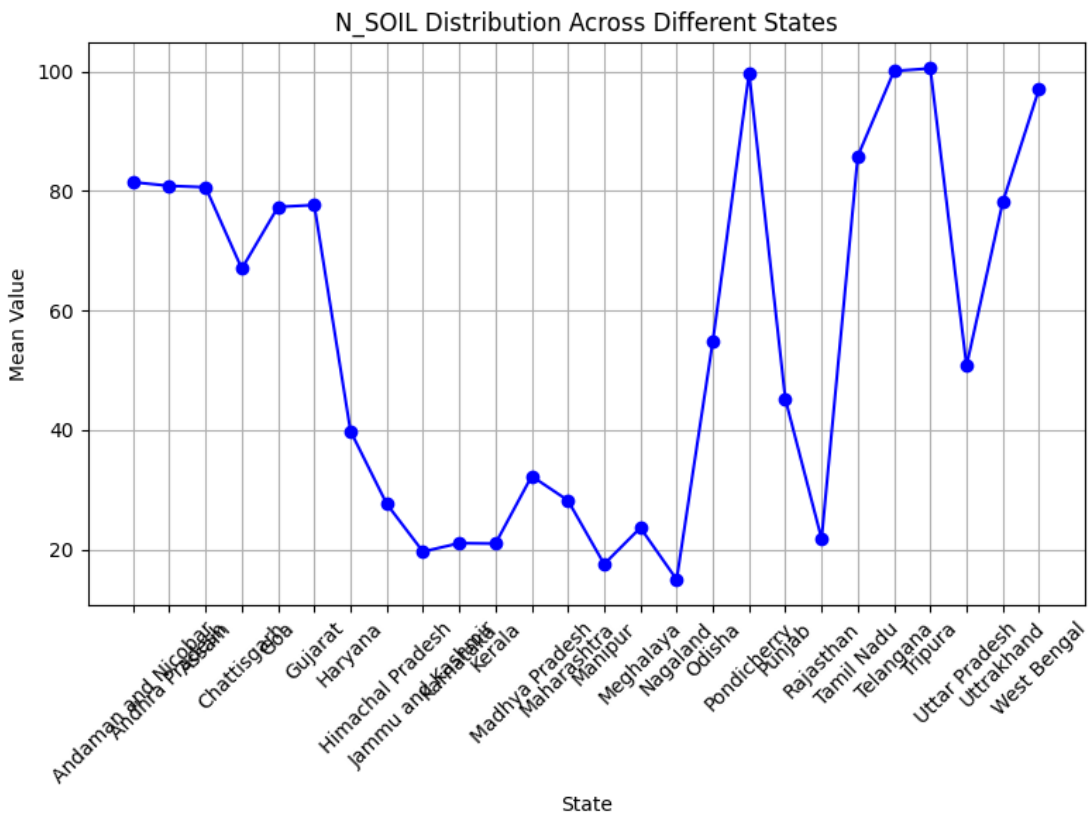
In [9]:
```
import numpy as np

def winsorize_column(df, col):
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3 - q1
    upper_limit = q3 + (1.5 * iqr)
    lower_limit = q1 - (1.5 * iqr)
    df[col] = np.where(df[col] <= lower_limit, lower_limit, df[col])
    df[col] = np.where(df[col] >= upper_limit, upper_limit, df[col])

def handle_outliers(df, outlier_columns):
    for col in outlier_columns:
        winsorize_column(df, col)

# Assuming df is your DataFrame

# Call the function to handle outliers
outlier_columns = ['N_SOIL', 'P_SOIL','K_SOIL','TEMPERATURE', 'HUMIDITY', 'ph', 'RAINFALL', 'CROP_PRICE']
handle_outliers(df, outlier_columns)

# Check for outliers after winsorization
outlier_columns, outliers_df, lower_limit, upper_limit = check_outliers(df)
print(outliers_df)
print(outlier_columns)
```

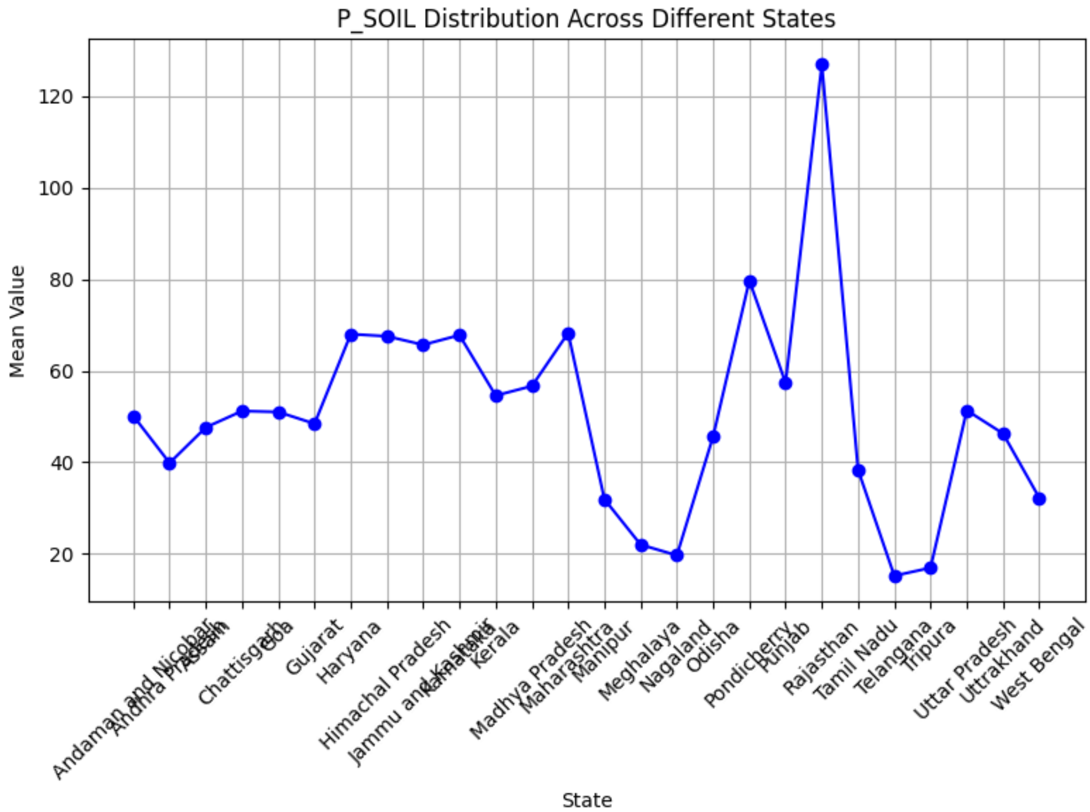
```
Series([], dtype: float64)
Empty DataFrame
Columns: [Column, outlier vals, Outlier Count]
Index: []
[]
```

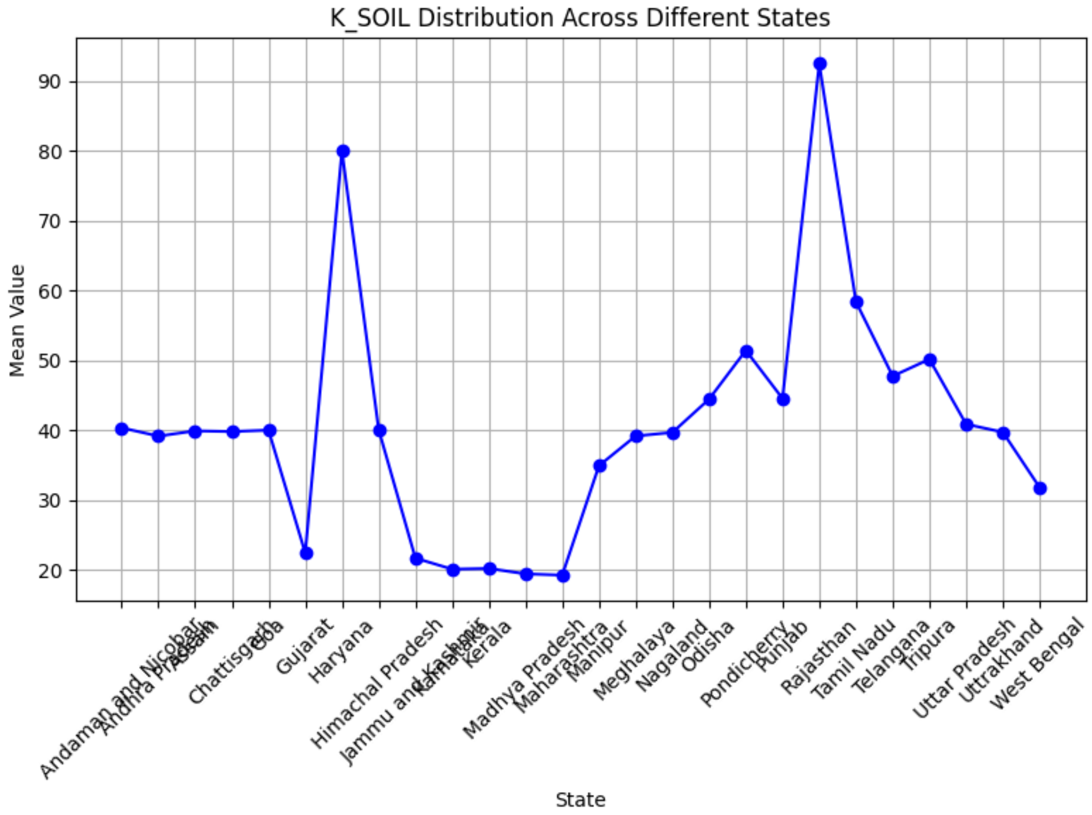
In [10]:
```
import matplotlib.pyplot as plt
numerical_cols = ['N_SOIL', 'P_SOIL', 'K_SOIL', 'TEMPERATURE', 'HUMIDITY', 'ph', 'RAINFALL', 'CROP_PRICE']

# Plot box plots for each numerical column individually
for col in numerical_cols:
    plt.figure(figsize=(8, 6))
    df.boxplot(column=col)
    plt.title(f'Boxplot of {col}')
    plt.ylabel('Values')
    plt.show()
```

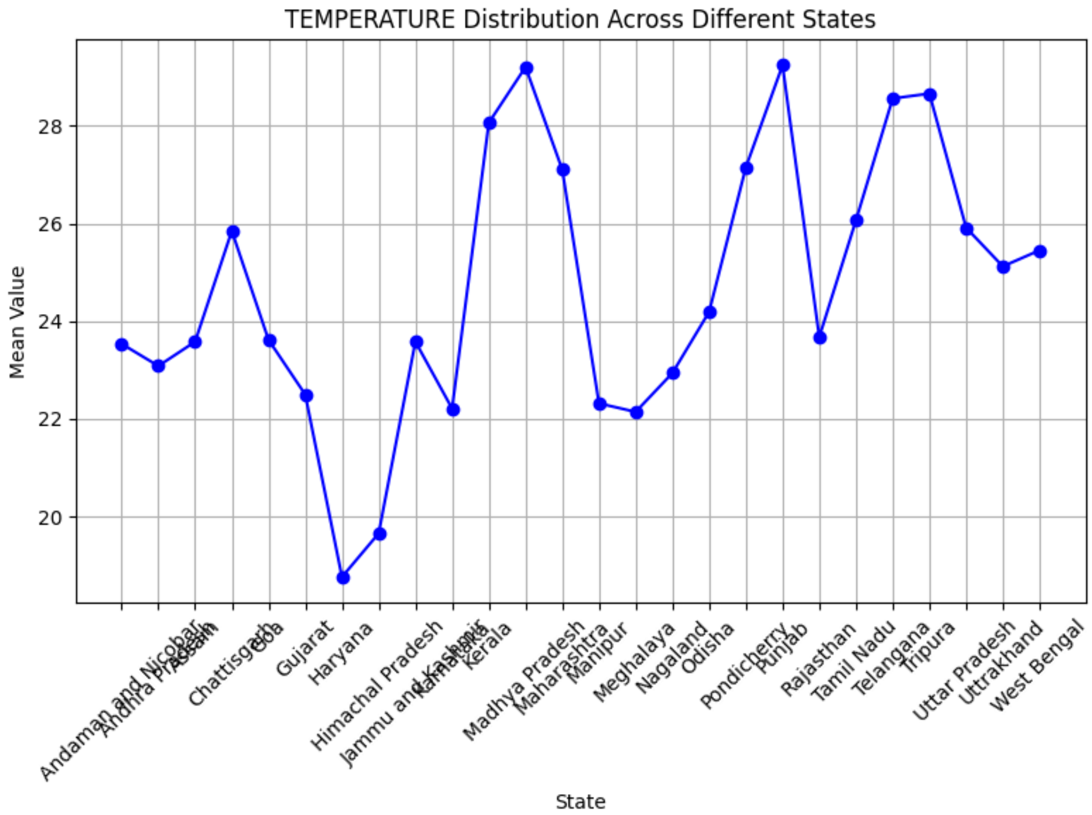
EDA1: Agricultural Diversity Across States

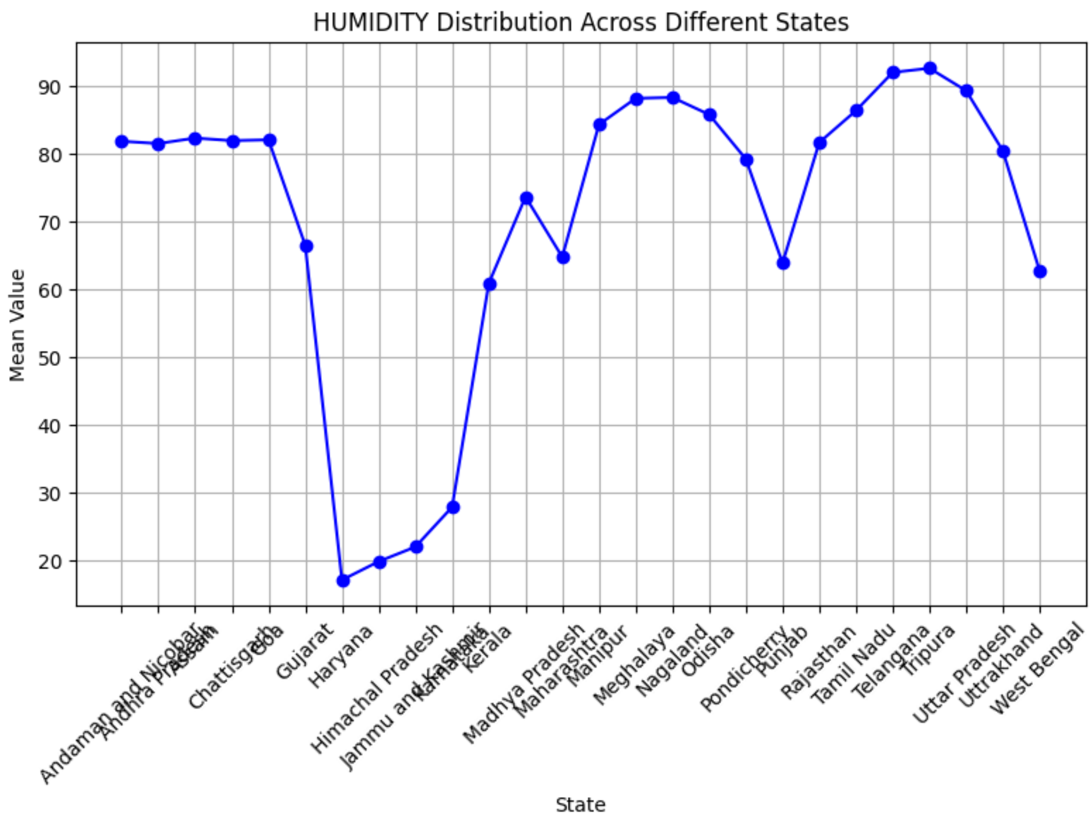
In [11]:
```
statewise_mean = df.groupby('STATE').mean()

# Plotting individual line plots for each feature
for feature in statewise_mean.columns:
    plt.figure(figsize=(8, 6))
    plt.plot(statewise_mean.index, statewise_mean[feature], marker='o', color='blue')
    plt.title(f'{feature} Distribution Across Different States')
    plt.xlabel('State')
    plt.ylabel('Mean Value')
    plt.xticks(rotation=45)
    plt.grid(True)
    plt.tight_layout()
    plt.show()

# Displaying the corresponding dataframe
print("Statewise Mean Dataframe:")
print(statewise_mean)
```

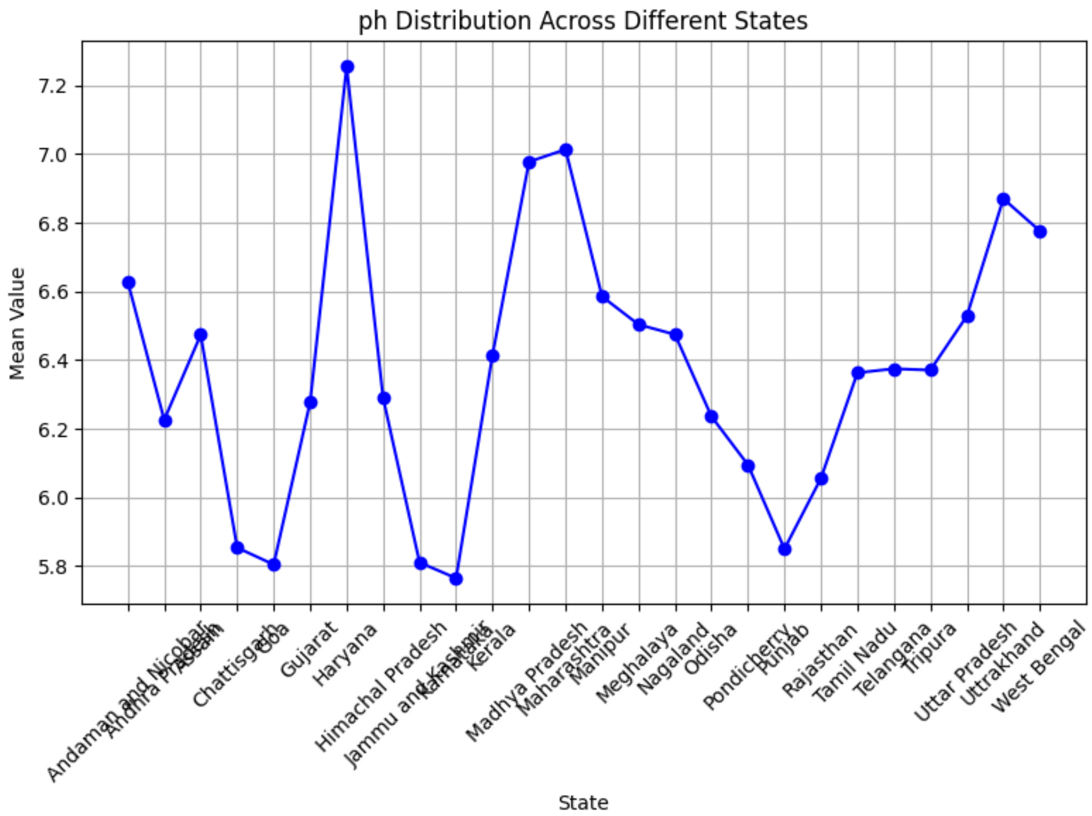

N_SOIL Distribution Across Different States
Mean Value
100
80
60
40
20
State
Andaman and Nicobar
Andhra Pradesh
Assam
Chattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Nagaland
Odisha
Pondicherry
Punjab
Rajasthan
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttrakhand
West Bengal

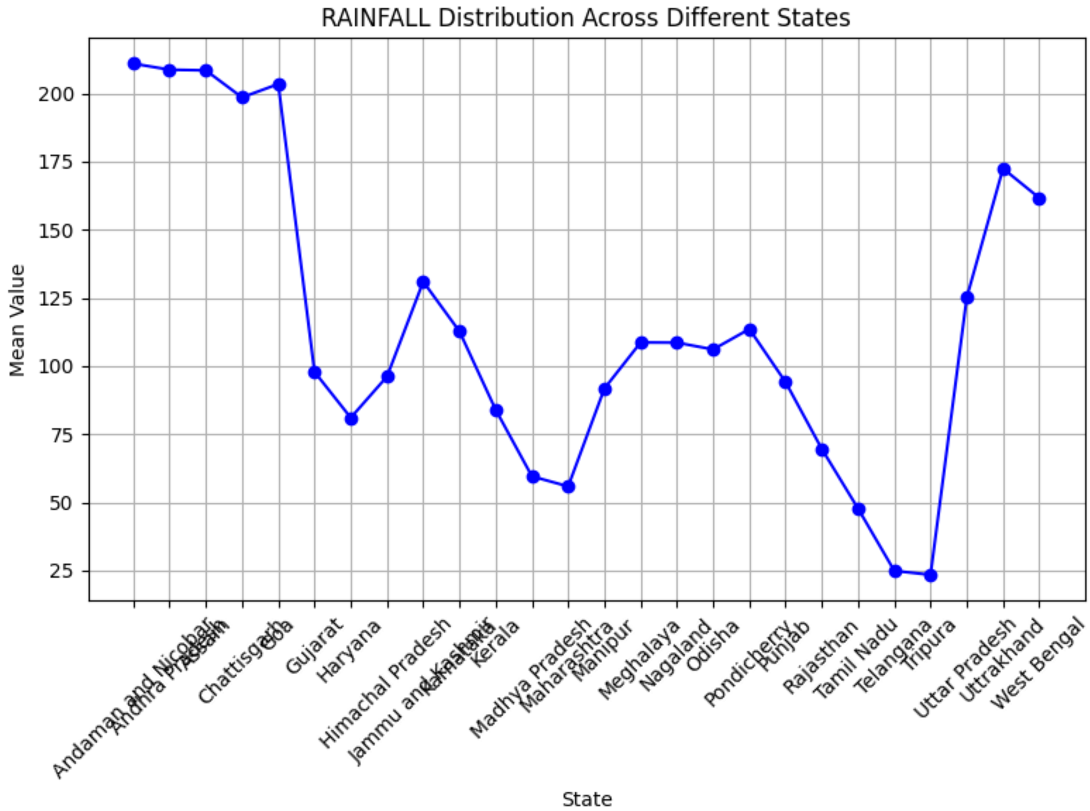

P_SOIL Distribution Across Different States
Mean Value
120
100
80
60
40
20
State
Andaman and Nicobar
Andhra Pradesh
Assam
Chattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Nagaland
Odisha
Pondicherry
Punjab
Rajasthan
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttrakhand
West Bengal

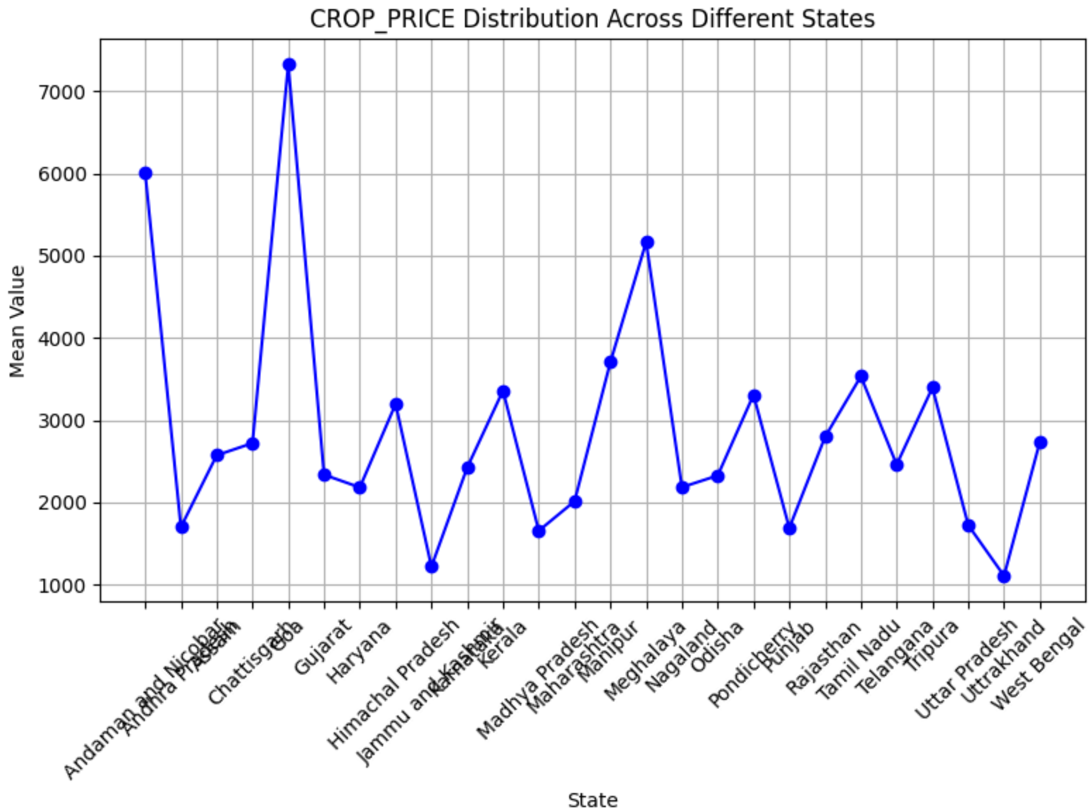

K_SOIL Distribution Across Different States
Mean Value
State
Andaman and Nicobar
Andhra Pradesh
Assam
Chattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Nagaland
Odisha
Pondicherry
Punjab
Rajasthan
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttrakhand
West Bengal


TEMPERATURE Distribution Across Different States
Mean Value
State
Andaman and Nicobar
Andhra Pradesh
Assam
Chattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Nagaland
Odisha
Pondicherry
Punjab
Rajasthan
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttrakhand
West Bengal


HUMIDITY Distribution Across Different States
Mean Value
State
Andaman and Nicobar
Andhra Pradesh
Assam
Chattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Nagaland
Odisha
Pondicherry
Punjab
Rajasthan
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttrakhand
West Bengal


ph Distribution Across Different States
Mean Value
State
Andaman and Nicobar
Andhra Pradesh
Assam
Chattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Nagaland
Odisha
Pondicherry
Punjab
Rajasthan
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttrakhand
West Bengal


RAINFALL Distribution Across Different States
Mean Value
200
175
150
125
100
75
50
25
State
Andaman and Nicobar
Andhra Pradesh
Assam
Chattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Nagaland
Odisha
Pondicherry
Punjab
Rajasthan
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttrakhand
West Bengal


CROP_PRICE Distribution Across Different States
Mean Value
7000
6000
5000
4000
3000
2000
1000
State
Andaman and Nicobar
Andhra Pradesh
Assam
Chattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Nagaland
Odisha
Pondicherry
Punjab
Rajasthan
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttrakhand
West Bengal

State

Statewise Mean Dataframe:

STATE	N_SOIL	P_SOIL	K_SOIL	TEMPERATURE
Andaman and Nicobar	81.466667	50.133333	40.333333	23.536551
Andhra Pradesh	80.857143	39.857143	39.142857	23.084331
Assam	80.620690	47.620690	39.896552	23.581132
Chattisgarh	67.000000	51.200000	39.800000	25.849626
Goa	77.333333	51.000000	40.000000	23.619286
Gujarat	77.646018	48.424779	22.398230	22.499366
Haryana	39.830769	68.000000	80.015385	18.763236
Himachal Pradesh	27.647059	67.539216	40.078431	19.656499
Jammu and Kashmir	19.666667	65.666667	21.666667	23.582200
Karnataka	21.083333	67.777778	20.083333	22.211717
Kerala	21.010949	54.572993	20.182482	28.064220
Madhya Pradesh	32.295455	56.659091	19.409091	29.205291
Maharashtra	28.228395	68.259259	19.228395	27.105562
Manipur	17.576923	31.923077	34.903846	22.318109
Meghalaya	23.666667	22.000000	39.166667	22.144096
Nagaland	15.000000	19.666667	39.666667	22.938963
Odisha	54.930233	45.662791	44.441860	24.191453
Pondicherry	99.714286	79.714286	51.428571	27.149553
Punjab	45.194444	57.355556	44.533333	29.242229
Rajasthan	21.804878	126.878049	92.500000	23.676312
Tamil Nadu	85.841530	38.349727	58.379781	26.067595
Telangana	100.066667	15.200000	47.733333	28.561532
Tripura	100.500000	16.944444	50.166667	28.661707
Uttar Pradesh	50.768293	51.339721	40.853659	25.915344
Uttrakhand	78.142857	46.285714	39.714286	25.123533
West Bengal	97.040323	32.241935	31.806452	25.452349

\

STATE	HUMIDITY	ph	RAINFALL	CROP_PRICE
Andaman and Nicobar	81.890278	6.628980	210.926315	6015.000000
Andhra Pradesh	81.554087	6.226104	208.709266	1698.571429
Assam	82.360072	6.473817	208.460805	2571.896552
Chattisgarh	81.979156	5.853723	198.539593	2720.000000
Goa	82.103312	5.804503	203.445806	7325.000000
Gujarat	66.559053	6.277621	97.841790	2335.646018
Haryana	17.192642	7.254993	81.027750	2181.169231
Himachal Pradesh	19.917225	6.289806	96.130018	3196.764706
Jammu and Kashmir	22.078571	5.810429	130.946567	1216.666667
Karnataka	27.944109	5.764943	112.706079	2423.388889
Kerala	60.944048	6.415968	83.929736	3358.302920
Madhya Pradesh	73.700232	6.978073	59.522094	1653.750000
Maharashtra	64.822194	7.013929	55.830781	2012.901235
Manipur	84.358748	6.585476	91.748464	3704.326923
Meghalaya	88.217547	6.503811	108.647579	5166.666667
Nagaland	88.355434	6.474680	108.583974	2183.333333
Odisha	85.828090	6.235814	106.076634	2325.755814
Pondicherry	79.276878	6.095029	113.565653	3307.000000
Punjab	63.933192	5.850205	94.086207	1687.444444
Rajasthan	81.706933	6.055711	69.548357	2796.512195
Tamil Nadu	86.404819	6.363186	47.699478	3532.737705
Telangana	92.041883	6.375065	24.858913	2459.000000
Tripura	92.656773	6.370987	23.402672	3395.833333
Uttar Pradesh	89.350102	6.528581	125.163188	1721.670732
Uttrakhand	80.487850	6.870214	172.455477	1105.952381
West Bengal	62.822172	6.776599	161.658207	2733.387097

Soil Nutrients (N_SOIL, P_SOIL, K_SOIL): The levels of nitrogen (N_SOIL), phosphorus (P_SOIL), and potassium (K_SOIL) vary considerably among states. For instance, Punjab shows high levels of potassium, while Haryana exhibits higher phosphorus content.

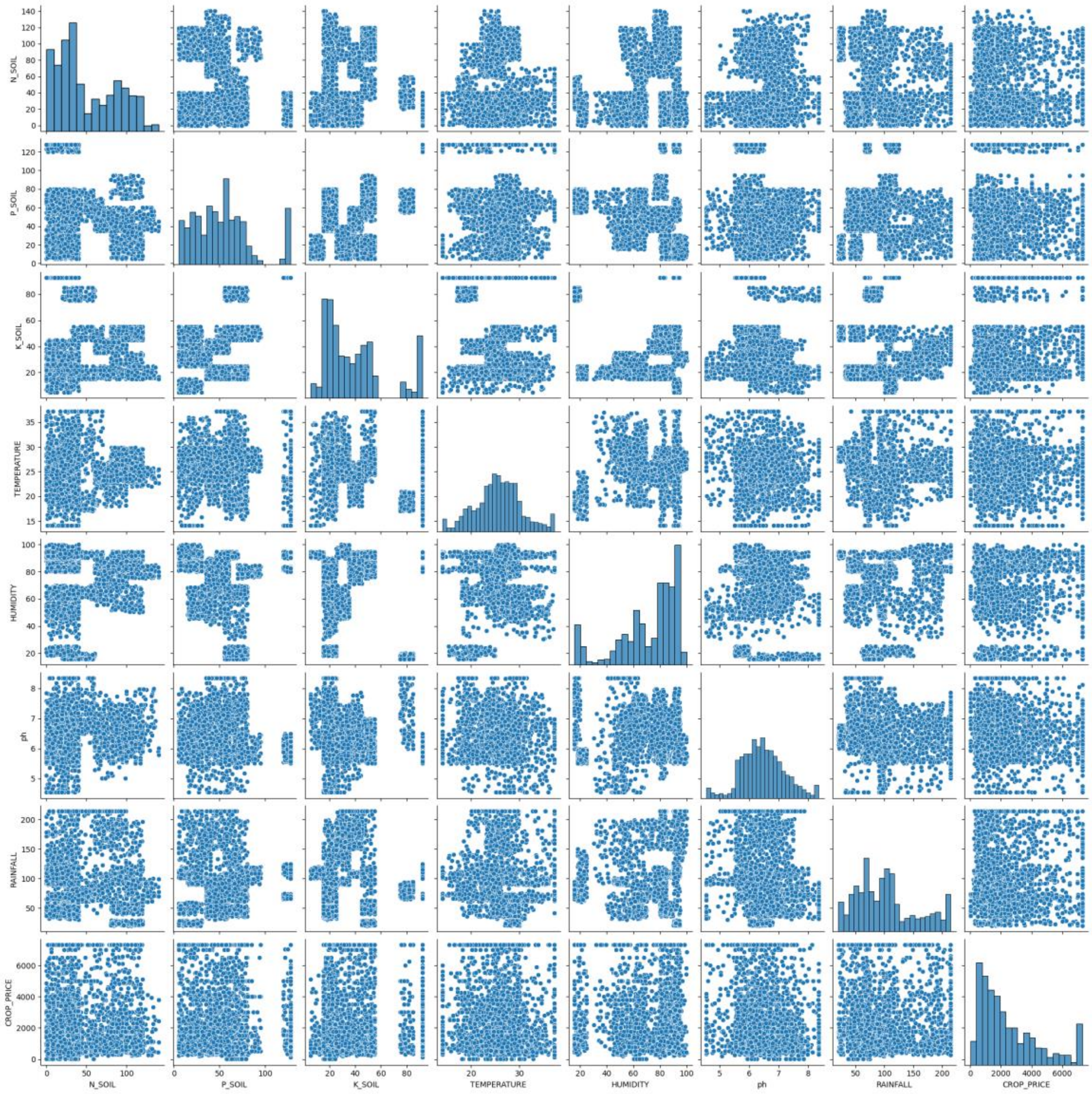
Environmental Factors (TEMPERATURE, HUMIDITY, RAINFALL): Temperature, humidity, and rainfall demonstrate diverse patterns across states. Southern states like Kerala and Tamil Nadu typically have higher temperatures and rainfall, whereas northern states like Rajasthan and Haryana have lower humidity levels.

pH Levels:pH levels vary slightly across states but generally fall within the optimal range for most crops, indicating favorable conditions for cultivation.

EDA2:Is there any relationship between temperature, humidity, and rainfall?

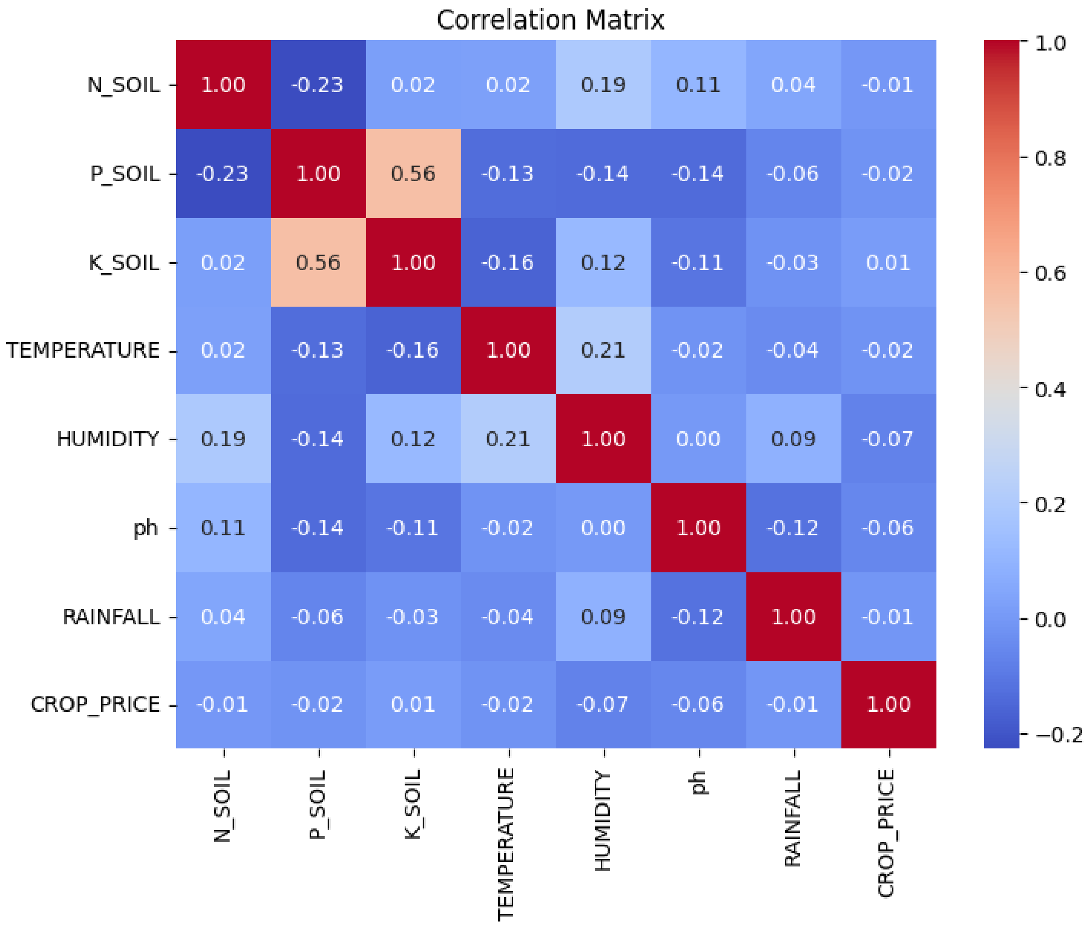
In [12]:
```
# Visualizing pairwise relationships using scatter plot matrix
import seaborn as sns
sns.pairplot(df)
plt.suptitle('Pairwise Relationships between Temperature, Humidity, and Rainfall', y=1.02)
plt.show()
```

Pairwise Relationships between Temperature, Humidity, and Rainfall



In [13]:
```python
correlation_matrix = df.corr()

# Visualizing correlation matrix using heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix ')
plt.show()

# Displaying the correlation matrix dataframe
print("Correlation Matrix:")
print(correlation_matrix)
```

```
<ipython-input-13-4484acdb289b>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a futur
e version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  correlation_matrix = df.corr()
```



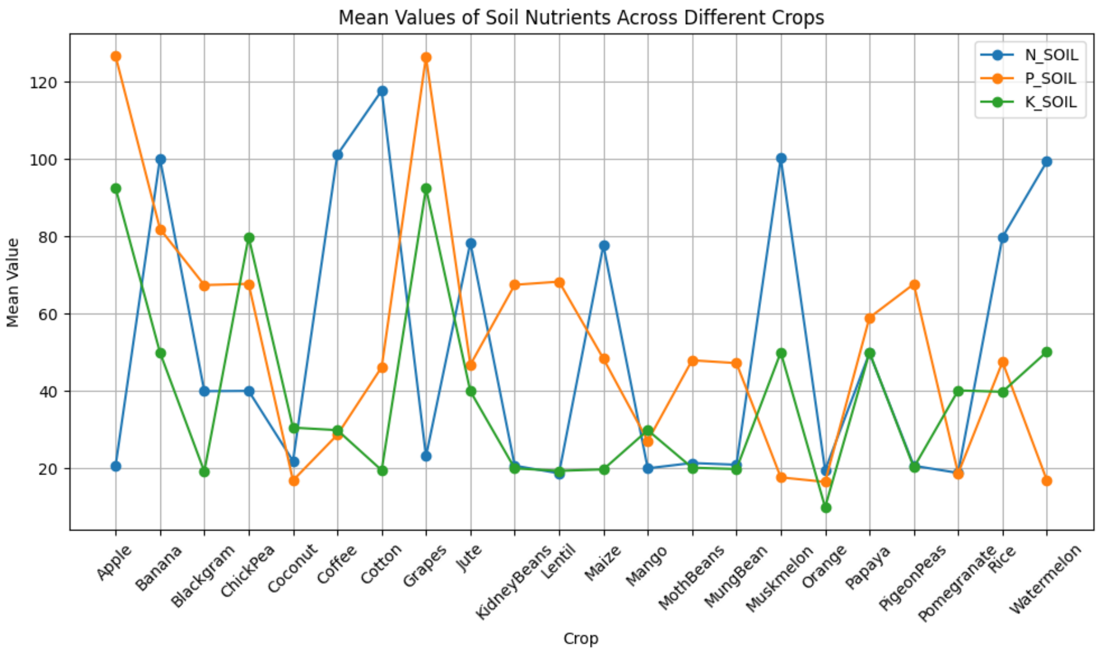

```
Correlation Matrix:
               N_SOIL    P_SOIL    K_SOIL  TEMPERATURE  HUMIDITY        ph  \
N_SOIL       1.000000 -0.227834  0.019000     0.023850  0.190746  0.106321
P_SOIL      -0.227834  1.000000  0.561850    -0.132569 -0.138850 -0.140243
K_SOIL       0.019000  0.561850  1.000000    -0.164312  0.120765 -0.107937
TEMPERATURE  0.023850 -0.132569 -0.164312     1.000000  0.212362 -0.020906
HUMIDITY     0.190746 -0.138850  0.120765     0.212362  1.000000  0.000323
ph           0.106321 -0.140243 -0.107937    -0.020906  0.000323  1.000000
RAINFALL     0.044041 -0.063986 -0.026358    -0.041047  0.085163 -0.119116
CROP_PRICE  -0.005032 -0.021370  0.008500    -0.017688 -0.073569 -0.056628

             RAINFALL  CROP_PRICE
N_SOIL       0.044041   -0.005032
P_SOIL      -0.063986   -0.021370
K_SOIL      -0.026358    0.008500
TEMPERATURE -0.041047   -0.017688
HUMIDITY     0.085163   -0.073569
ph          -0.119116   -0.056628
RAINFALL     1.000000   -0.011362
CROP_PRICE  -0.011362    1.000000
```

there is a slight positive correlation between temperature and humidity(0.21), there is no significant linear relationship between temperature, humidity, and rainfall based on the correlation coefficients calculated.

This indicates that as temperature increases, humidity tends to slightly increase as well.

EDA3: Are there any significant differences in soil nutrients (N_SOIL, P_SOIL, K_SOIL) between different crops?

In [14]:
```
# Calculate mean values of soil nutrients for each crop
mean_values = df.groupby('CROP').mean()

# Plotting line plots for mean values of soil nutrients
plt.figure(figsize=(10, 6))

# Plot for N_SOIL
plt.plot(mean_values.index, mean_values['N_SOIL'], marker='o', label='N_SOIL')

# Plot for P_SOIL
plt.plot(mean_values.index, mean_values['P_SOIL'], marker='o', label='P_SOIL')

# Plot for K_SOIL
plt.plot(mean_values.index, mean_values['K_SOIL'], marker='o', label='K_SOIL')

plt.title('Mean Values of Soil Nutrients Across Different Crops')
plt.xlabel('Crop')
plt.ylabel('Mean Value')
plt.xticks(rotation=45)
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Displaying the dataframe of mean values
print("Mean Values of Soil Nutrients Across Different Crops:")
print(mean_values)
```



1. **Nitrogen (N_SOIL):**
   - Nitrogen levels vary widely across crops, ranging from as low as 18.77 for Lentil to as high as 117.77 for Cotton. This indicates diverse nitrogen requirements among different crops.
2. **Phosphorus (P_SOIL):**
   - Phosphorus levels also exhibit considerable variation across crops, with values ranging from 16.55 for Orange to 126.66 for Apple. Such disparities highlight the importance of phosphorus management tailored to specific crop needs.
3. **Potassium (K_SOIL):**
   - Potassium levels display notable differences across crops, with values spanning from 10.01 for Orange to 92.50 for several crops including Apple and Grapes. Understanding these variations is crucial for optimizing potassium fertilization strategies.

Overall, these findings underscore the necessity of crop-specific soil nutrient management practices to ensure optimal growth, yield, and overall crop health. Farmers and agricultural practitioners should consider these variations in soil nutrient levels when formulating fertilization plans and crop management strategies to maximize agricultural productivity and sustainability.

In [15]:
```
state_crop_counts = df.groupby('STATE')['CROP'].value_counts().reset_index(name='COUNT')

# Get the most commonly grown crop for each state
most_common_crops = state_crop_counts.groupby('STATE').first().reset_index()


# Displaying dataframe
print("Most Commonly Grown Crops in Each State:")
print(most_common_crops)

plt.tight_layout()
plt.show()
```

```
Most Commonly Grown Crops in Each State:
                 STATE         CROP  COUNT
0   Andaman and Nicobar         Rice     15
1        Andhra Pradesh         Rice      7
2                 Assam         Rice     58
3          Chattisgarh          Rice      5
4                   Goa         Rice      3
5               Gujarat        Maize    100
6               Haryana     ChickPea     65
7      Himachal Pradesh  KidneyBeans     68
8     Jammu and Kashmir  KidneyBeans      3
9             Karnataka  KidneyBeans     29
10               Kerala    MothBeans    100
11       Madhya Pradesh    Blackgram     25
12          Maharashtra       Lentil     87
13              Manipur  Pomegranate     39
14            Meghalaya  Pomegranate      6
15             Nagaland  Pomegranate      6
16               Odisha  Pomegranate     49
17          Pondicherry       Banana      7
18               Punjab        Mango    100
19            Rajasthan       Grapes     41
20           Tamil Nadu   Watermelon    100
21            Telangana    Muskmelon     15
22              Tripura    Muskmelon     18
23        Uttar Pradesh        Apple    100
24           Uttrakhand         Jute     21
25          West Bengal       Coffee    100

<Figure size 640x480 with 0 Axes>
```

Rice dominates the agricultural landscape in states like Assam, Andaman and Nicobar, and Andhra Pradesh. States like Gujarat, Kerala, Punjab, Tamil Nadu, and Uttar Pradesh show a high prevalence of specific crops such as Maize, MothBeans, Mango, Watermelon, and Apple respectively. Other states exhibit a variety of dominant crops, including Pomegranate in Manipur, Grapes in Rajasthan, and ChickPea in Haryana.

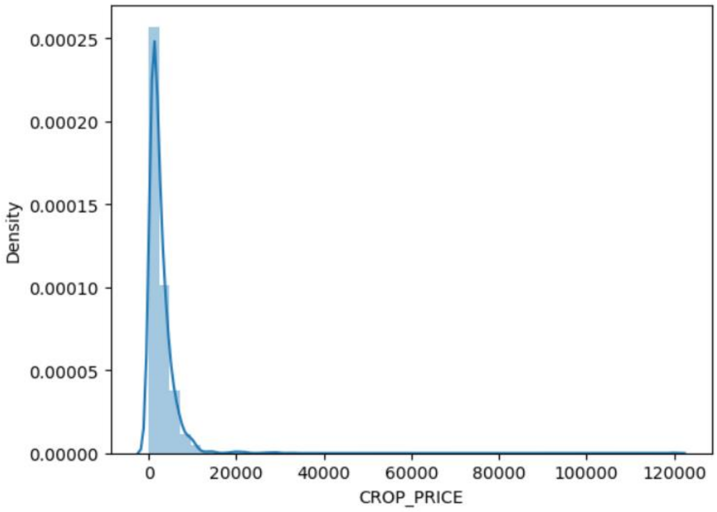
In [3]:
```
import seaborn as sns
```

In [4]:
```
sns.distplot(df['CROP_PRICE'])
```

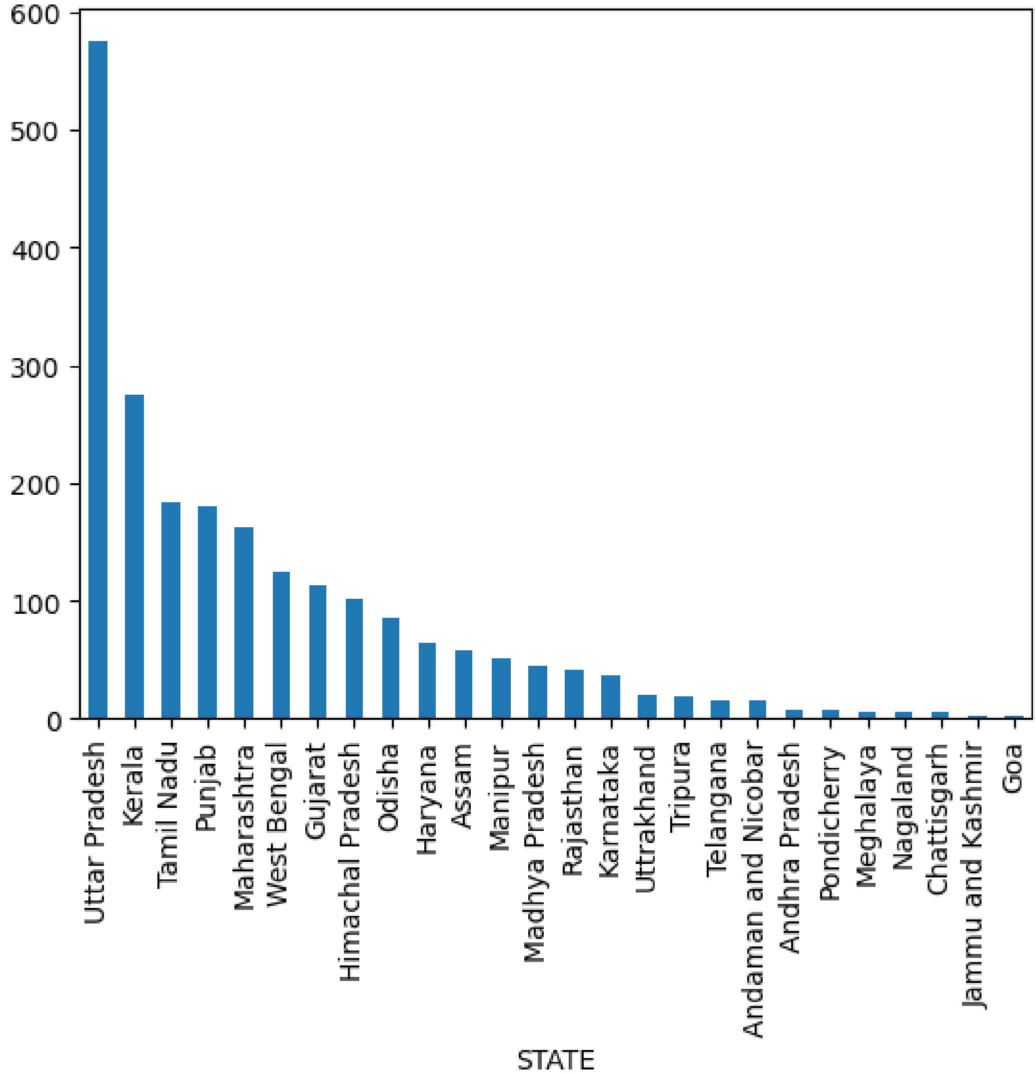
```
C:\Users\Shubham\AppData\Local\Temp\ipykernel_18768\3652019946.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['CROP_PRICE'])
```

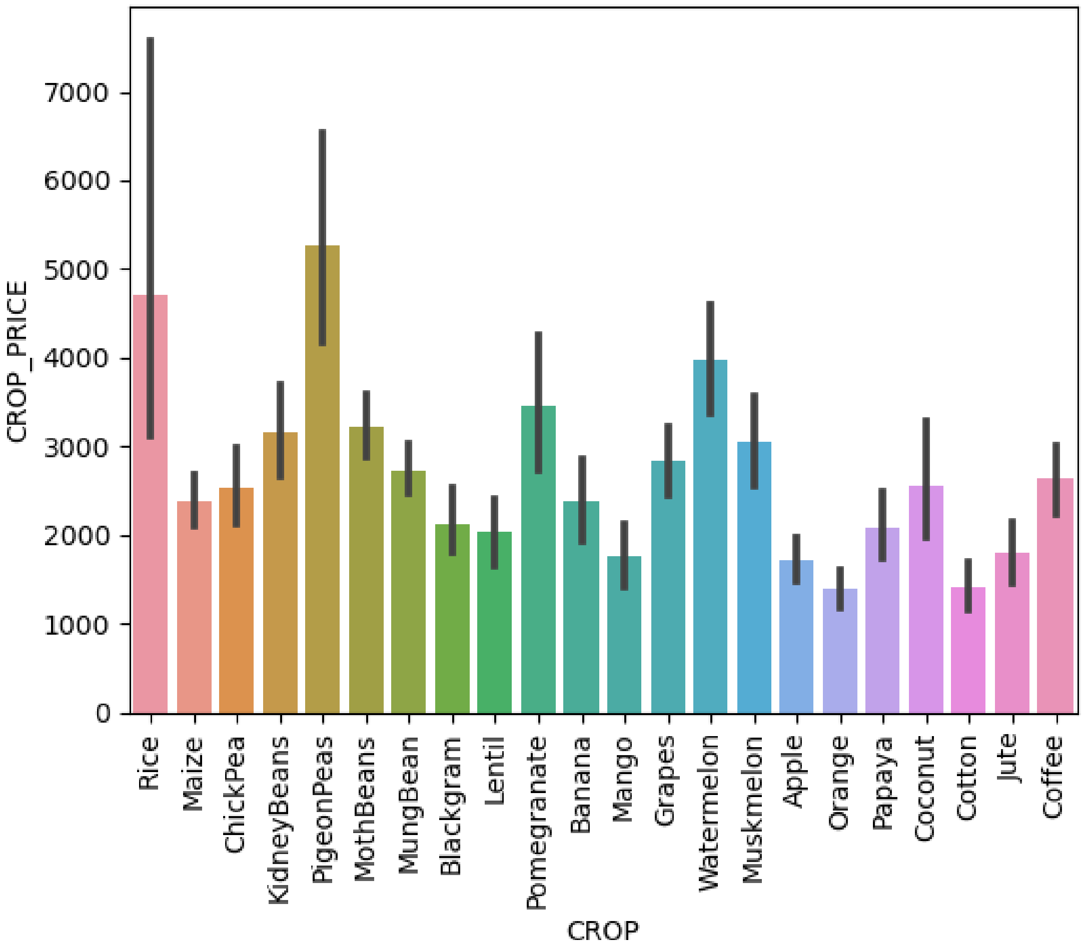
Out[4]:
```
<Axes: xlabel='CROP_PRICE', ylabel='Density'>
```

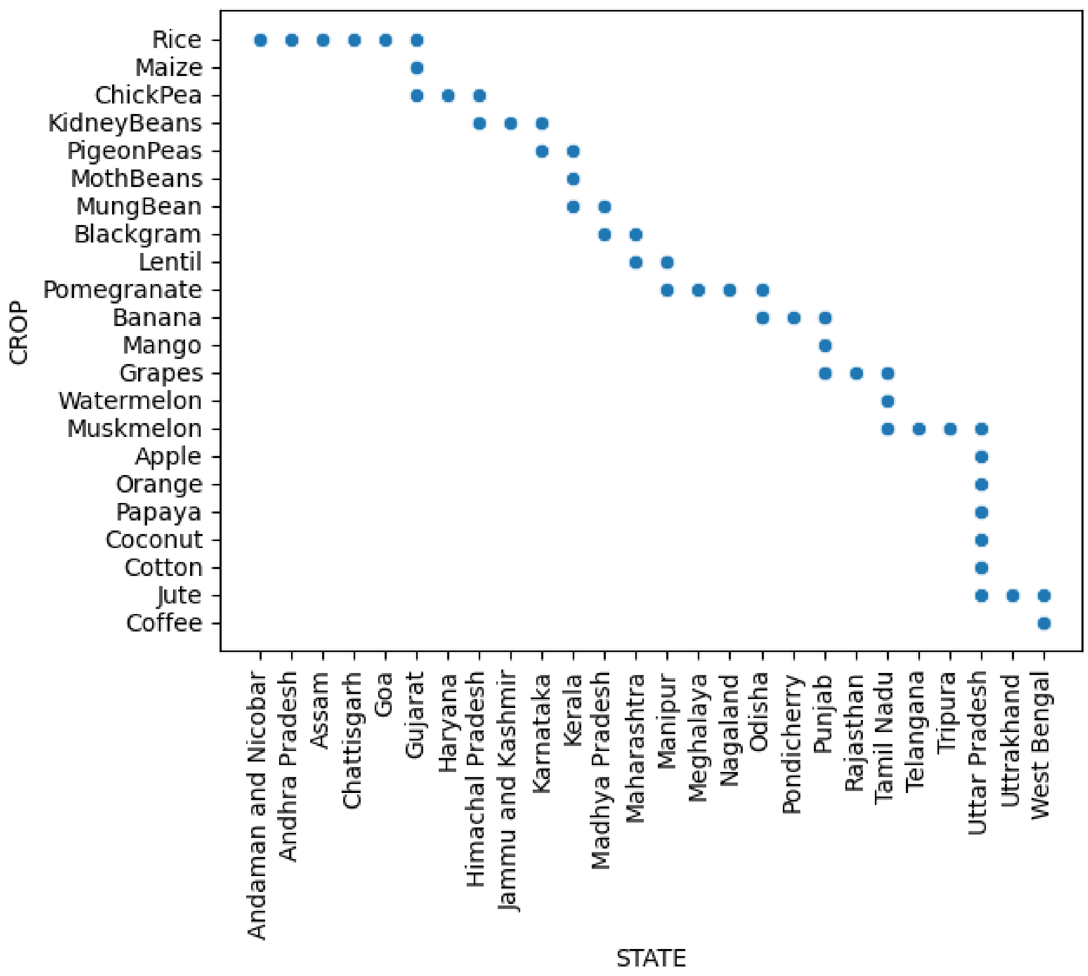
Out[4]: <Axes: xlabel='CROP_PRICE', ylabel='Density'>



In this plot we could that the distribution of crop price is left skewed.




7000
6000
5000
4000
3000
2000
1000
0
CROP_PRICE
Rice
Maize
ChickPea
KidneyBeans
PigeonPeas
MothBeans
MungBean
Blackgram
Lentil
Pomegranate
Banana
Mango
Grapes
Watermelon
Muskmelon
Apple
Orange
Papaya
Coconut
Cotton
Jute
Coffee
CROP


CROP
Rice
Maize
ChickPea
KidneyBeans
PigeonPeas
MothBeans
MungBean
Blackgram
Lentil
Pomegranate
Banana
Mango
Grapes
Watermelon
Muskmelon
Apple
Orange
Papaya
Coconut
Cotton
Jute
Coffee
STATE
Andaman and Nicobar
Andhra Pradesh
Assam
Chattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Nagaland
Odisha
Pondicherry
Punjab
Rajasthan
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttrakhand
West Bengal

ENCODING CATEGORICAL INTO NUMERICAL

In [23]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['CROP']=le.fit_transform(df['CROP'])
df['STATE'] = le.fit_transform(df['STATE'])
```

SPLITING OF FEATURES

In [24]:
```
x=df.iloc[:,0:9]
y=df.iloc[:,9]
x.head()
```

Out[24]:

	N_SOIL	P_SOIL	K_SOIL	TEMPERATURE	HUMIDITY	ph	RAINFALL	STATE	CROP_PRICE
0	90	3.737670	3.761200	20.879744	4.406753	6.502985	202.935536	0	3500.0
1	85	4.060443	3.713572	21.770462	4.386014	7.038096	226.655537	0	3500.0
2	60	4.007333	3.784190	23.004459	4.410623	7.840207	263.964248	0	3500.0
3	74	3.555348	3.688879	26.491096	4.384004	6.980401	242.864034	0	3500.0
4	78	3.737670	3.737670	20.130175	4.401889	7.628473	262.717340	0	3500.0

In [25]:
```
y.unique()
```

Out[25]:
```
array([20, 11,  3,  9, 18, 13, 14,  2, 10, 19,  1, 12,  7, 21, 15,  0, 16,
       17,  4,  6,  8,  5])
```

###Training a model

TRAIN-TEST-SPLIT

In [26]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,random_state=1)
```

In [27]:
```
print("Training data",x_train.shape)
```

```
Training data (1540, 9)
```

In [28]:
```
print("Training data",x_test.shape)
```

```
Training data (660, 9)
```

IMPORTING ALGORITHM

NAIVE BAYES and **XGBClassifier**
XGB perfoms slightly better

In [29]:
```
# Naive Bayes
# from sklearn.naive_bayes import GaussianNB
# model = GaussianNB()

#XGBoost
from xgboost import XGBClassifier
model = XGBClassifier(objective = 'multi:softmax', num_class = len(y.unique()))
```

In [30]:
```
model.fit(x_train,y_train)
```

Out[30]:

XGBClassifier

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_bin=256, max_cat_threshold=64, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_class=22, num_parallel_tree=1,
```

PREDICTION OF CROP

In [31]:
```
y_prediction=model.predict(x_test)
```

MODEL METRICS

In [32]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_prediction)
```

Out[32]: 1.0

In [40]:
```
output_df = pd.DataFrame({"Actual_output":y_test})
```

In [41]:
```
output_df
```

Out[41]:

	Actual_output
1276	7
1446	15
335	9
1458	15
2038	8
...	...
1418	15
478	18
1181	12

In [42]:
```
output_df['XGBClassifier Prediction'] = y_prediction
```

In [43]:
```
output_df
```

Out[43]:

	Actual_output	XGBClassifier Prediction
1276	7	7
1446	15	15
335	9	9
1458	15	15
2038	8	8
...	...	...
1418	15	15
478	18	18
1181	12	12
1000	1	1
1132	12	12

660 rows × 2 columns

In [44]:
```
import matplotlib.pyplot as plt
```

In [45]:
```
fig, ax = plt.subplots(figsize=(8,3))

sns.histplot(output_df['Actual_output'], color='blue', alpha=0.5, label="actual")
sns.histplot(output_df['XGBClassifier Prediction'], color='red', alpha=0.5, label="prediction")
```

Out[45]: <matplotlib.legend.Legend at 0x212036a10a0>

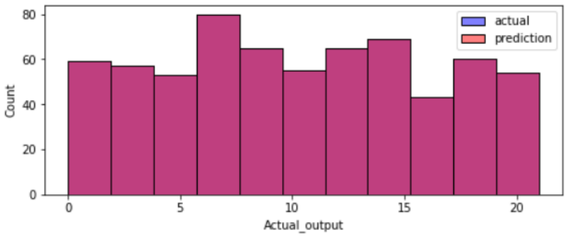


Fig - Prediction

1. BUSINESS MODELLING

The business model for Crop Prediction typically revolves around providing a valuable service to farmers, agricultural companies, or other stakeholders in the agriculture industry.

1. **Subscription or service fees**

Basic Plan: Provides predictions for a limited number of crops with monthly updates.

Pro Plan: Covers a wider range of crops with more frequent updates (e.g., weekly or bi-weekly).

Premium Plan: Includes additional features such as customized alerts, advanced analytics, and personalized recommendations.

Free Trial Period: Offer a free trial period (e.g., 7 or 14 days) during which users can access the full range of features. This allows potential customers to experience the value of the service before committing to a subscription.

By implementing a flexible pricing model tailored to the needs of different customer segments, the Crop Prediction platform can maximize its revenue potential while ensuring affordability and value for its users. Additionally, regular updates and improvements to the platform can help retain subscribers and attract new customers over time.

2. Customization and Consulting Services:

By offering customization and consulting services, the Crop Prediction platform can provide added value to its clients by delivering personalized solutions that address their unique challenges and objectives. This not only enhances the effectiveness of the platform but also strengthens the relationship between the platform provider and its customers.

3. Fee-for-Service:

By implementing a fee-for-service model, the Crop Prediction platform can provide users with flexibility in accessing prediction services while also ensuring a steady stream of revenue for the platform provider. This model allows users to pay for only the services they need, making it attractive to a wide range of customers in the agriculture industry.

4. Pay-Per-Use

By implementing a pay-per-use model, the Crop Prediction platform can provide users with flexibility in accessing prediction services while also ensuring that they only pay for the services they use. This can be particularly attractive to users with sporadic or occasional needs for crop predictions who prefer to pay on a per-usage basis.

1. Marketplace

By creating a marketplace for Crop Prediction services, you can connect users with qualified service providers, streamline the process of accessing prediction services, and create a vibrant ecosystem of collaboration and innovation within the agriculture industry.

Additionally, these training initiatives can serve as revenue streams and contribute to the platforms overall success and impact in the agriculture industry.

1. Determining the Overall Cost:

The total cost of an Crop Prediction platform can vary widely depending on factors such as the scope and complexity of the platform, the size of the target market, the level of customization required, and the chosen business model. Stakeholders need to conduct a thorough cost analysis and budgeting process to ensure adequate funding and financial sustainability for the platform.

Certainly, adjusting the subscription cost to make it more attractive and accessible to customers is a viable strategy.

Lower Subscription Fee: Decrease the average subscription fee to make it more affordable for customers. This could potentially attract more users to the platform.

Increase User Base: With a lower subscription fee, the platform may attract more users, increasing the total number of subscribers.

Revenue from Increased User Base: Although the individual subscription fee is lower, the increase in the number of users can offset the price reduction.

14. Financial Equation

In [148…

```
# Define the polynomial coefficients
a = -1.148e-08
b = - 1.014e-06
c = 0.0002617
d = 0.004356
e = 0.04133
# Profit calculation:
C= 1000     # for eg. taking cost price per production

# Evaluate the polynomial at certain value of x
x_val = 82.07022
polynomial_value = a * pow(x_val,4) + b * pow(x_val,3) + c * pow(x_val,2) + d * x_val + e
print("Polynomial value at x =", x_val, "is", polynomial_value)
print(C*polynomial_value) # profit eg
```

```
Polynomial value at x = 82.07022 is 1.0801730830662053
1080.1730830662052
```


In [165…

```
# Fit curve
x = np.arange(0, len(df))
y = df['Production'].values
p = np.polyfit(x, y, 2)
f = np.poly1d(p)
```

In [166…

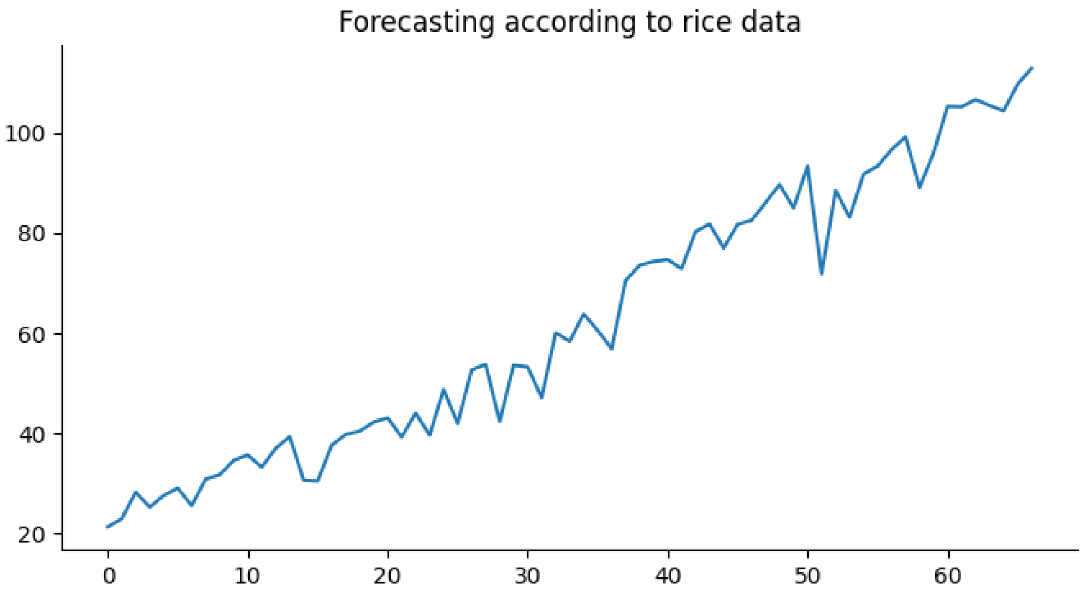
```
print(f) # Print equation
```

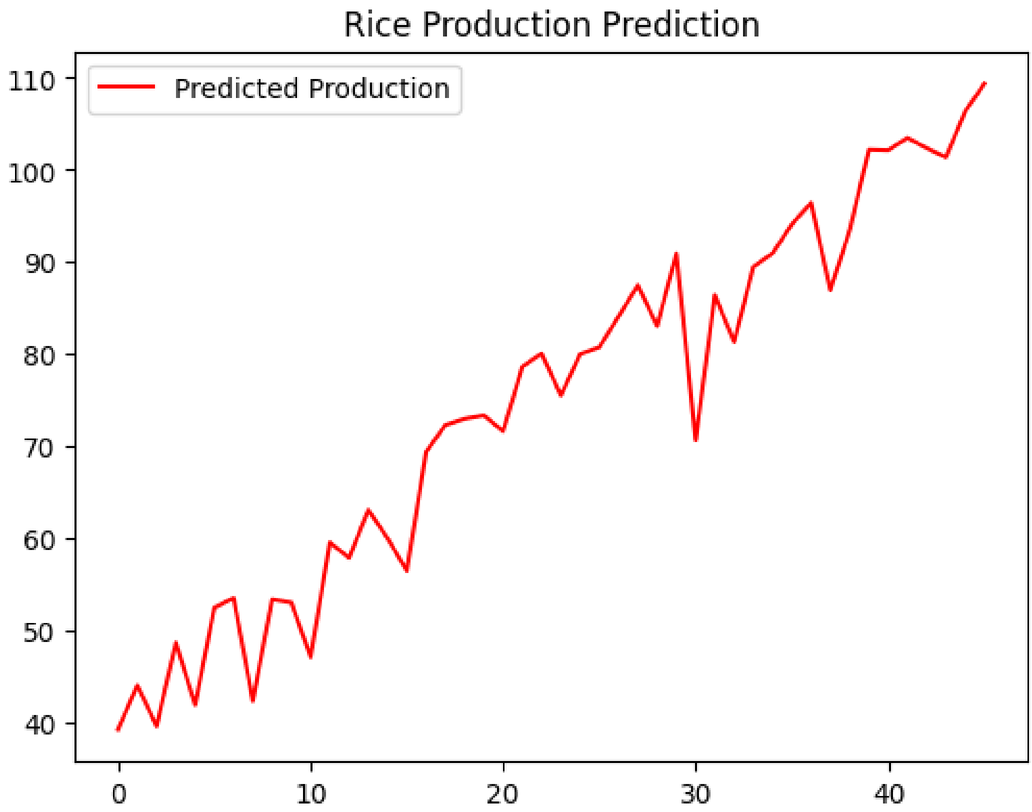
```
          2
0.007956 x + 0.9395 x + 1.586
```

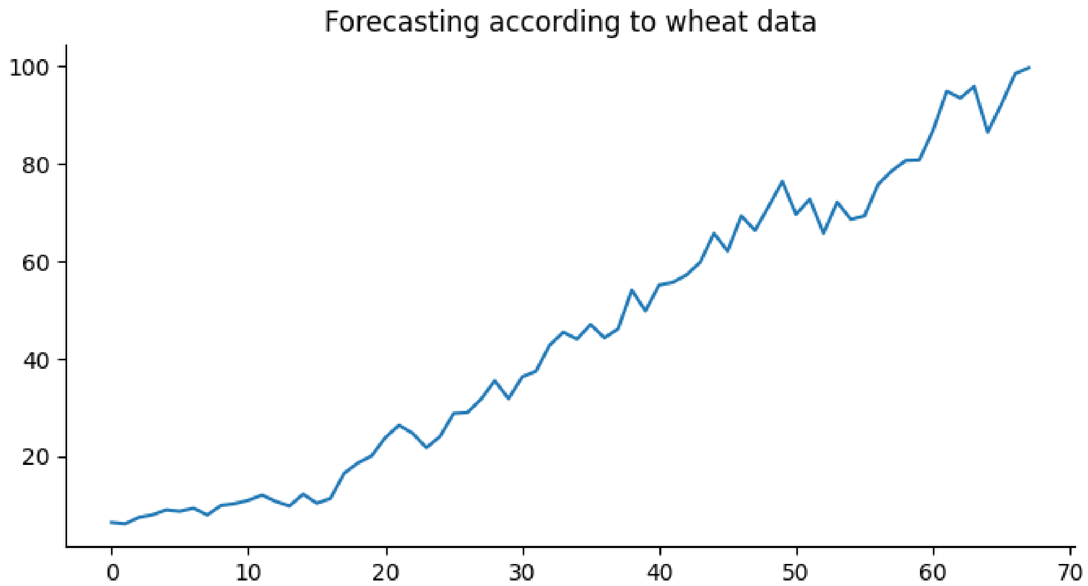
In [168…

```
# Define the polynomial coefficients
a = 0.007956
b = 0.9395
c = 1.586
# Profit calculation:
C= 1000     # for eg. taking cost price per production

# Evaluate the polynomial at certain value of x
x_val = 1.9178593e+13
polynomial_value = a * pow(x_val,2) + b * x_val + c
print("Polynomial value at x =", x_val, "is", polynomial_value)
print(C*polynomial_value) # profit eg
```

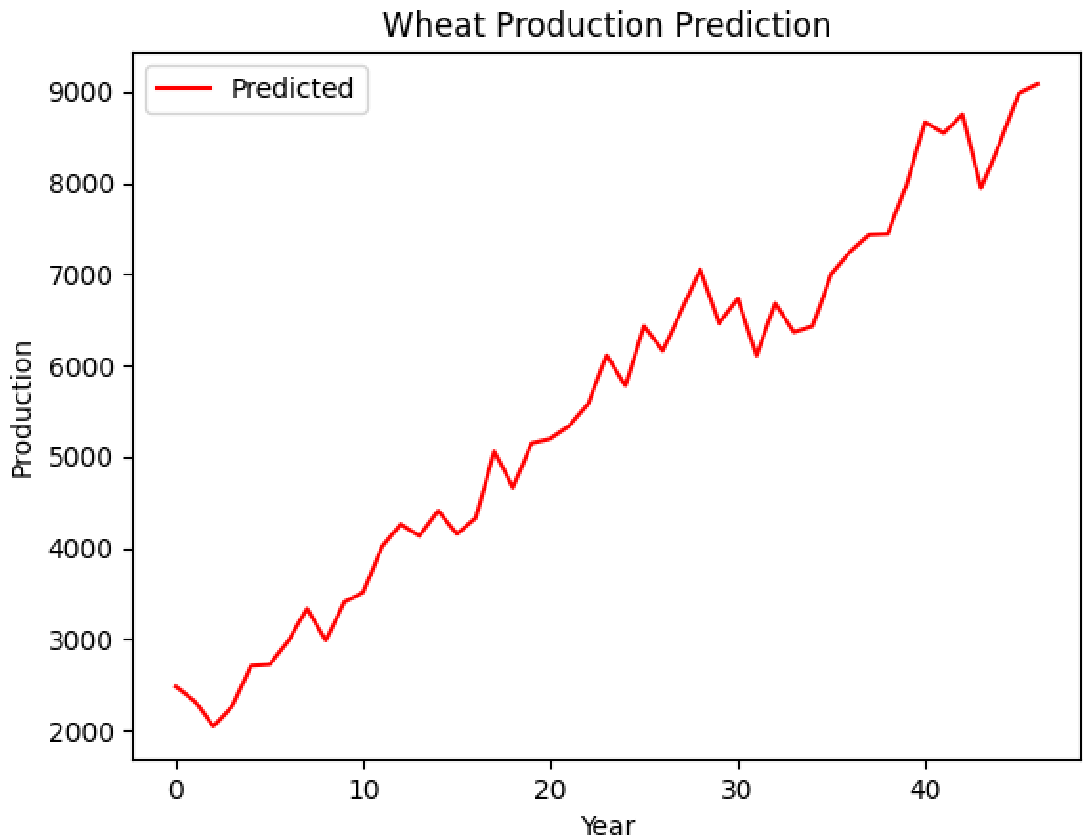
```
Polynomial value at x = 19178593000000.0 is 2.9263634247989856e+24
2.926363424798986e+27
```


Forecasting according to rice data
100
80
60
40
20
0
10
20
30
40
50
60


Rice Production Prediction
Predicted Production
110
100
90
80
70
60
50
40
0
10
20
30
40



Forecasting according to wheat data
100
80
60
40
20
0
10
20
30
40
50
60
70



Wheat Production Prediction
Predicted
Production
9000
8000
7000
6000
5000
4000
3000
2000
0
10
20
30
40
Year

15. Conclusion

The Crop Prediction System project, named AgriVision , represents a comprehensive and innovative solution for farmers and the agricultural community. The project aims to revolutionize farming practices by leveraging advanced technologies to provide accurate crop yield predictions, promote sustainable agriculture, and deliver valuable educational content. Through a user friendly interface, robust back end processes, and a range of features, AgriVision stands as a holistic tool to empower farmers and enhance their overall farming experience.

16. References

GitHub Link of Project:

https://github.com/UMAMAHESHWARRAO302001/feynn_Crop_Prediction_task

Patents: https://patents.google.com

Government Laws and Regulations:

https://www.indiacode.nic.in/
https://www.eurekaselect.com/chapter/17458