VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI – 590018

Project Report
on

CROP AND CROP YIELD PREDICTION

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering
In
Artificial Intelligence and Machine Learning

Submitted by
Pavithra M
1BG21AI079

Vidyayāmruthamashnuthe

B.N.M. Institute of Technology
An Autonomous Institution under VTU

Department of Artificial Intelligence and Machine Learning

2022-23

B.N.M. Institute of Technology

An Autonomous Institution under VTU

Department of Artificial Intelligence and Machine Learning

Vidyayāmruthamashnuthe

CERTIFICATE

Certified that the Mini Project entitled **Crop and Crop Yield Prediction** carried out by Ms. **Pavithra M** USN **1BG21AI079** a bonafide student of IV Semester B.E., **B.N.M Institute of Technology** in partial fulfillment for the Bachelor of Engineering in ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING of the **Visvesvaraya Technological University**, Belagavi during the year 2022-23. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of Python Programming and Applications prescribed for the said Degree.

Dr. Tejaswini R Murgod
Associate Professor
Department of AIML
BNMIT, Bengaluru

Dr. Sheba Selvam
Professor and HOD
Department of AIML
BNMIT, Bengaluru

ACKNOWLEDGEMENT

I would like to thank **Shri Narayan Rao R. Maanay,** Secretary, BNMEI, Bengaluru for providing an excellent academic environment in the College.

I would like to sincerely thank **Prof. T. J. Rama Murthy**, Director, BNMIT, Bengaluru, for having extended his support and encouraging me during the course of the work.

I would like to sincerely thank **Dr. S.Y. Kulkarni**, Additional Director, BNMIT, Bengaluru forhaving extended his support and encouraging me during the course of the work.

I would like to express my gratitude to **Prof. Eishwar N. Maanay**, Dean, BNMIT, Bengalurufor his relentless support, guidance and assistance.

I would like to thank **Dr. Krishnamurthy G.N**, Principal, BNMIT, Bengaluru for his constant encouragement.

I would like to thank **Dr. Sheba Selvam,** Professor and Head of the Department of Artificial Intelligence and Machine Learning, BNMIT, Bengaluru who has shared her opinions and thoughts which helped me in the completion of my project successfully.

I would also like to thank my Course teacher **Dr. Tejaswini R Murgod,** Associate Professor, Department of Artificial Intelligence and Machine Learning, BNMIT, Bengaluru for guiding ina systematic manner.

Finally, I would like to thank all technical and non-technical faculty members of Department of Artificial Intelligence and Machine Learning, BNMIT, Bengaluru, for their support. I would like to thank my Family and Friends for their unfailing moral support and encouragement.

PAVITHRA M

(1BG21AI079)

ABSTRACT

Agriculture plays a pivotal role in global sustenance, demanding effective technological solutions to enhance productivity and sustainability. This project focuses on leveraging machine learning techniques to predict crops and estimate crop yields, addressing crucial challenges faced by farmers. Through data preprocessing, model development, and performance evaluation, our project aims to provide accurate recommendations for crop selection and reliable estimates of crop yields. The successful deployment of predictive models as a web-based API further enhances the practical applicability of our solutions. This report encapsulates the journey from data analysis to model deployment, reflecting the project's contributions to the agricultural sector and underscoring the significance of data-driven innovation in shaping modern farming practices.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Agriculture is a crucial sector that plays a significant role in the economy of many countries worldwide. It is responsible for providing food and raw materials to industries and households. However, predicting crops and their yields accurately remains a challenging task due to various factors such as weather conditions, pest and disease infestations, soil fertility, and other external factors.

1.1. Overview

The realm of agriculture is undergoing a profound transformation through the integration of technology and data science. Among the pivotal challenges in this sector is the accurate prediction of both crop types and their corresponding yields. These predictions hold the potential to revolutionize farming practices, maximize resource utilization, and ensure global food security. By harnessing historical data, advanced machine learning techniques, and domain expertise, this project endeavors to offer solutions to these critical challenges.

1.2. Aim

The core objective of this project is twofold: first, to create a model capable of predicting the most suitable crop for a given set of environmental conditions, and second, to forecast the yield of the chosen crop. These aims address the needs of farmers, agricultural planners, and policymakers who seek accurate insights for optimal decision-making. The project's central focus on both crop recommendation and yield prediction underscores its holistic approach to modern agriculture.

1.3. Objective

The primary objectives of this project encompass:

- Compilation and preprocessing of comprehensive agricultural data, encompassing factors like soil nutrient levels, climate parameters, and historical crop yields.

- Development and training of machine learning models to predict the optimal crop for a specific context, considering factors such as soil quality, temperature, humidity, and more.
- Creation of predictive models that anticipate crop yields based on the amalgamation of relevant parameters.
- Evaluation of model efficacy through rigorous analysis and benchmarking against established metrics.
- Provision of actionable insights and recommendations to aid farmers, agricultural practitioners, and stakeholders in making informed decisions.

1.4. Scope

This project undertakes a dual-pronged scope: the accurate prediction of suitable crops for specific conditions and the estimation of potential crop yields. Encompassing data preprocessing, exploratory analysis, model training, and evaluation, this endeavor delves into the development of sophisticated machine learning algorithms that encapsulate the intricate interplay between environmental variables and crop outcomes. Nutrient levels, climatic conditions, and geographical attributes all factor into the project's purview.

1.5. Applications

The implications of precise crop and yield prediction are multifaceted and far-reaching. Farmers can harness these predictions to optimize planting schedules, irrigation practices, and fertilization strategies. Agricultural policymakers stand to benefit from informed decisions for resource allocation, food security planning, and policy formulation. Agribusinesses and supply chain stakeholders can leverage these predictions to streamline logistics, bolster market strategies, and enhance overall efficiency.

The ensuing chapters will delve into the project's technical intricacies, encompassing data preprocessing, model architecture, performance evaluation, and model deployment. The outputs of this project hold the potential to reshape modern agriculture by amalgamating data-driven insights into the intricate fabric of farming practices.

Chatpter 2

LITERATURE SURVEY

The literature survey is a comprehensive examination of existing research and studies relevant to crop prediction, yield estimation, and agricultural data analysis. The survey aims to provide an overview of the state of the art in the field, highlight key findings, and identify gaps or areas where your project can make a valuable contribution.

Crop Prediction and Recommendation

Research in the domain of crop prediction and recommendation has evolved significantly with the advent of machine learning and data analytics. Various studies have explored methods for predicting suitable crops based on environmental factors, soil characteristics, and historical crop performance. One prominent approach is the utilization of decision tree algorithms to determine the optimal crop for a given set of conditions. Additionally, ensemble techniques such as random forests have shown promise in improving prediction accuracy and robustness.

Crop Yield Estimation

Accurate estimation of crop yields is a critical component of modern agriculture. Researchers have explored a multitude of factors that influence crop yield, including nutrient levels, weather patterns, and irrigation practices. Machine learning models, such as support vector machines and neural networks, have been employed to model these complex interactions and predict yield outcomes. Furthermore, the integration of remote sensing data and satellite imagery has enabled remote monitoring and yield estimation at a larger scale.

Data Preprocessing and Feature Engineering

The quality of predictive models relies heavily on the preprocessing and selection of relevant features from the data. Studies have emphasized the importance of handling missing data, normalizing variables, and selecting informative features for accurate predictions. Feature engineering techniques, such as polynomial features and interaction terms, have been used to enhance model performance by capturing non-linear relationships and interactions between variables.

Challenges and Opportunities

While advancements have been made in crop prediction and yield estimation, challenges persist. Data scarcity in certain regions, the variability of climate patterns, and the complexity of crop-environment relationships pose challenges to achieving high prediction accuracy. However, emerging technologies such as Internet of Things (IoT) devices for real-time data collection and advanced sensor networks hold potential to mitigate some of these challenges.

Research Gap and Contributions

Despite the progress in the field, there remains a need for comprehensive models that simultaneously predict both the most suitable crop for given conditions and the corresponding yield. Existing research often focuses on either prediction or yield estimation, with limited integration between the two. This project seeks to address this gap by developing a unified framework that combines crop prediction and yield estimation, thereby offering a more holistic solution for modern agricultural practices.

Chapter 3

SYSTEM REQUIREMENTS

This chapter outlines the hardware and software components essential for the successful implementation and execution of the crop and crop yield prediction project. Ensuring compatibility and availability of the required resources is crucial to the project's efficiency and accuracy.

3.1. Hardware Requirements

- A computer with at least 4GB of RAM
- A CPU with at least 2 cores
- Sufficient disk space to store the dataset and the project files.

3.2. Software Requirements

- Python 3.x (preferably the latest version)
- Python libraries such as NumPy, Pandas, Matplotlib, Scikit-learn, TensorFlow or Keras, PyTorch (depending on the approach we take for modelling)
- A dataset that contains historical crop yield data, weather data, and soil data. The dataset should cover a sufficient period and geographical area to ensure that the model can make accurate predictions
- You can use any integrated development environment (IDE) that supports Python, such as PyCharm, Spyder, Jupyter Notebook, or Google Colaboratory.

Chapter 4

DESIGN AND IMPLEMENTATION

4.1. System Design

The system design of the crop and crop yield prediction project can be divided into the following steps:

Data Collection

The first step in the system design is to collect the data required for training the machine learning models. We can collect the data from various sources, such as weather stations, satellites, and ground surveys. We can use the following data for the prediction of crop yields:

- Soil type: The soil type plays a vital role in determining the crop yield. We can collect the soil type data from various sources, such as soil surveys, soil databases, and remote sensing data.
- Weather data: The weather data includes rainfall, temperature, humidity, and other environmental factors that affect the crop yield. We can collect the weather data from weather stations or use remote sensing data to collect the weather data.
- Crop data: The crop data includes the crop type, planting date, harvesting date, and other information related to the crops. We can collect the crop data from the farmers or use remote sensing data to collect the crop data.

Data Pre-processing

After collecting the data, the next step is to pre-process the data before training the machine learning models. The pre-processing steps include data cleaning, data normalization, feature selection, and feature engineering. The data cleaning step involves removing the missing values and outliers from the data. The data normalization step involves scaling the data to a common range. The feature selection step involves selecting the most relevant features for the prediction of crop yields. The feature engineering step involves creating new features from the existing features.

Model Training

The next step is to train the machine learning models on the pre-processed data. We can use various machine learning algorithms such as regression, decision trees, random forests, and neural networks for the prediction of crop yields. We can use the following approaches for the training of machine learning models:

Supervised Learning: In this approach, we train the machine learning models on the labelled data, where the crop yield is the target variable, and the other factors are the input variables.

Unsupervised Learning: In this approach, we train the machine learning models on the unlabelled data, where the machine learning algorithms can find patterns in the data without any labels.

Model Evaluation

After training the machine learning models, the next step is to evaluate the performance of the models. We can use various performance metrics such as mean squared error (MSE), mean absolute error (MAE), and coefficient of determination (R-squared) to evaluate the performance of the models. We can also use cross-validation techniques to evaluate the models' performance on the unseen data.

Deployment

The final step is to deploy the machine learning models in a user-friendly interface. We can develop a web-based application or a mobile application that farmers can use to predict the crop yields.

4.2. Implementation

Crop Prediction model

To predict appropriate crops based on input environmental factors, a decision tree classification model was implemented using Scikit-Learn. This choice was driven by the model's interpretability and ability to handle diverse feature types. Cross-validation was employed to evaluate the model's performance using accuracy, precision, recall, and F1-score metrics.

In [6]:

```
import numpy as np
import pandas as pd
from sklearn import tree

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams
from sklearn.metrics import accuracy_score
```

In [7]:

```
df=pd.read_csv("crop_production.csv",encoding = "ISO-8859-1")
df.dtypes
```

Out[7]:

```
State_Name        object
District_Name     object
Crop_Year          int64
Season            object
Crop              object
Area             float64
Production       float64
dtype: object
```

In [8]:

```
df.head()
```

Out[8]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0

In [9]:

```
df['Production']=pd.to_numeric(df['Production'],errors='coerce')
df
```

Out[9]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Arecanut	1254.0	2000.0
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0
...	...	...	...	...	...	...	...
246086	West Bengal	PURULIA	2014	Summer	Rice	306.0	801.0
246087	West Bengal	PURULIA	2014	Summer	Sesamum	627.0	463.0
246088	West Bengal	PURULIA	2014	Whole Year	Sugarcane	324.0	16250.0
246089	West Bengal	PURULIA	2014	Winter	Rice	279151.0	597899.0
246090	West Bengal	PURULIA	2014	Winter	Sesamum	175.0	88.0

246091 rows × 7 columns

In [12]:

```
data = df.groupby('Crop_Year')[['Area', 'Production']].mean()
data = data.reset_index()
data
```

Out[12]:

	Crop_Year	Area	Production	CPI
0	1997	26038.324081	9.565489e+04	3.673619
1	1998	14479.153906	5.172545e+05	35.724086
2	1999	12678.074790	5.172145e+05	40.795984
3	2000	12102.612169	5.496723e+05	45.417661
4	2001	12371.499489	5.616144e+05	45.395827
5	2002	9463.680476	4.654666e+05	49.184519
6	2003	9954.769395	4.619857e+05	46.408482
7	2004	11891.933465	5.909555e+05	49.693814
8	2005	11822.333236	5.949965e+05	50.328177
9	2006	11913.672644	6.212016e+05	52.141903
10	2007	10513.848637	4.821251e+05	45.856191
11	2008	11768.527148	5.423063e+05	46.081067
12	2009	11738.077997	5.564389e+05	47.404599
13	2010	12557.355280	4.573050e+05	36.417306
14	2011	10918.140920	1.037554e+06	95.030260
15	2012	11369.858240	6.197705e+05	54.509962
16	2013	10368.125223	9.575947e+05	92.359485
17	2014	10549.306622	8.011596e+05	75.944286
18	2015	8187.362989	1.236197e+04	1.509884

In [14]:

```
data.describe()
```

Out[14]:

	Crop_Year	Area	Production	CPI
count	19.000000	19.000000	1.900000e+01	19.000000
mean	2006.000000	12141.402985	5.496122e+05	48.098795
std	5.627314	3633.397954	2.364693e+05	22.994038
min	1997.000000	8187.362989	1.236197e+04	1.509884
25%	2001.500000	10531.577629	4.737958e+05	43.095905
50%	2006.000000	11768.527148	5.496723e+05	46.408482
75%	2010.500000	12237.055829	6.073835e+05	51.235040
max	2015.000000	26038.324081	1.037554e+06	95.030260

In [15]:

```
print(data)
```

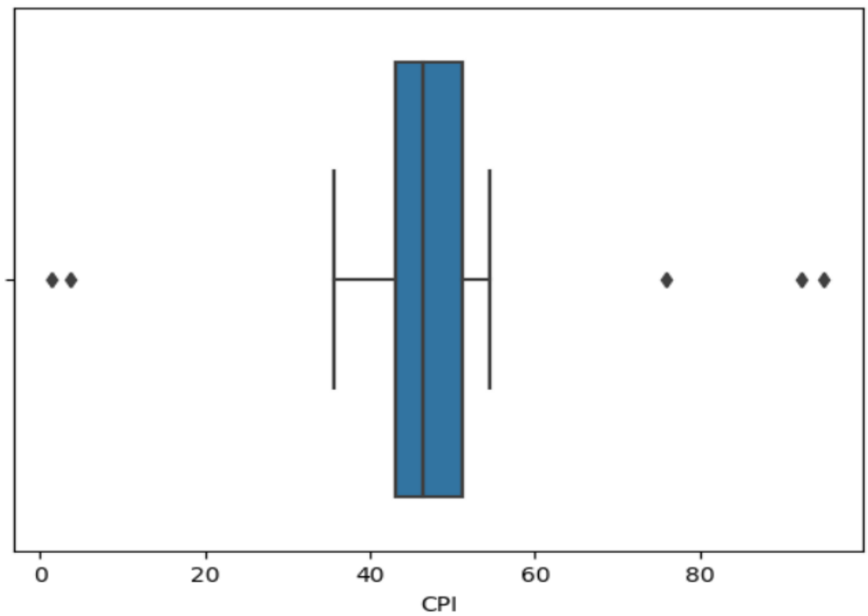
```
    Crop_Year          Area    Production        CPI
0        1997  26038.324081  9.565489e+04   3.673619
1        1998  14479.153906  5.172545e+05  35.724086
2        1999  12678.074790  5.172145e+05  40.795984
3        2000  12102.612169  5.496723e+05  45.417661
4        2001  12371.499489  5.616144e+05  45.395827
5        2002   9463.680476  4.654666e+05  49.184519
6        2003   9954.769395  4.619857e+05  46.408482
7        2004  11891.933465  5.909555e+05  49.693814
8        2005  11822.333236  5.949965e+05  50.328177
9        2006  11913.672644  6.212016e+05  52.141903
10       2007  10513.848637  4.821251e+05  45.856191
11       2008  11768.527148  5.423063e+05  46.081067
12       2009  11738.077997  5.564389e+05  47.404599
13       2010  12557.355280  4.573050e+05  36.417306
14       2011  10918.140920  1.037554e+06  95.030260
15       2012  11369.858240  6.197705e+05  54.509962
16       2013  10368.125223  9.575947e+05  92.359485
17       2014  10549.306622  8.011596e+05  75.944286
18       2015   8187.362989  1.236197e+04   1.509884
```

In [16]:

```
import seaborn as sns
sns.boxplot(x=data['CPI'])
```

Out[16]:

```
<Axes: xlabel='CPI'>
```



In [17]:

```
data = data[np.isfinite(data['CPI'])]
data=data[data.CPI >43]
data=data[data.CPI <51]
data.set_index('Crop_Year')
data
```

Out[17]:

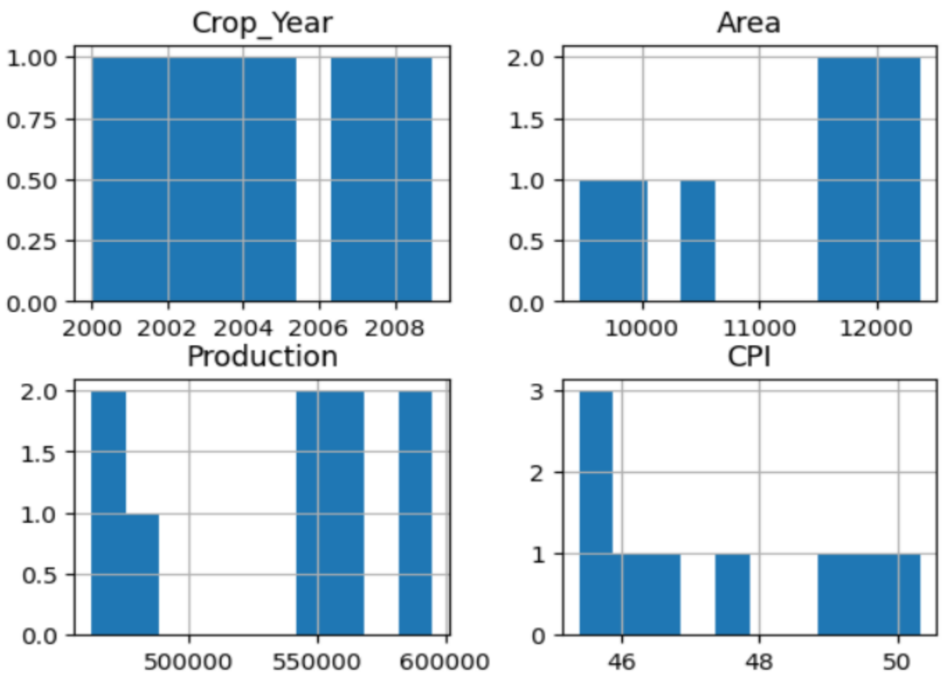
	Crop_Year	Area	Production	CPI
3	2000	12102.612169	549672.332849	45.417661
4	2001	12371.499489	561614.446722	45.395827
5	2002	9463.680476	465466.567649	49.184519
6	2003	9954.769395	461985.734566	46.408482
7	2004	11891.933465	590955.527122	49.693814
8	2005	11822.333236	594996.473832	50.328177
10	2007	10513.848637	482125.050009	45.856191
11	2008	11768.527148	542306.282654	46.081067
12	2009	11738.077997	556438.877374	47.404599

In [18]:

```
data.hist()
```

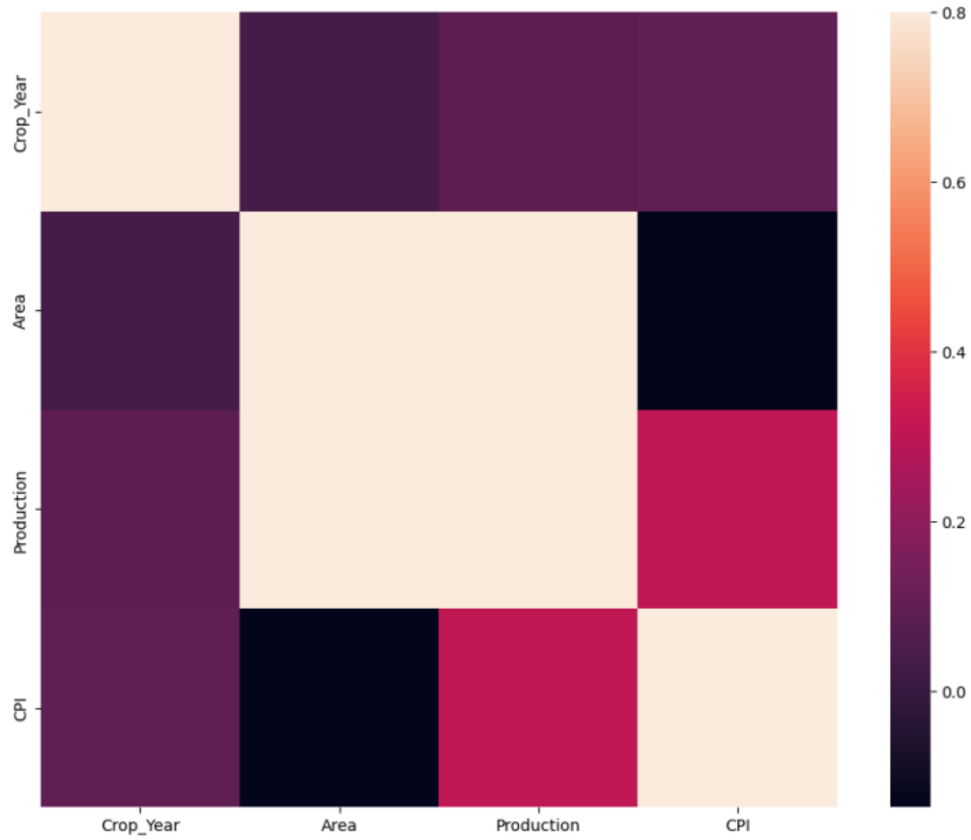
Out[18]:

```
array([[<Axes: title={'center': 'Crop_Year'}>,
        <Axes: title={'center': 'Area'}>],
       [<Axes: title={'center': 'Production'}>,
        <Axes: title={'center': 'CPI'}>]], dtype=object)
```



In [19]:

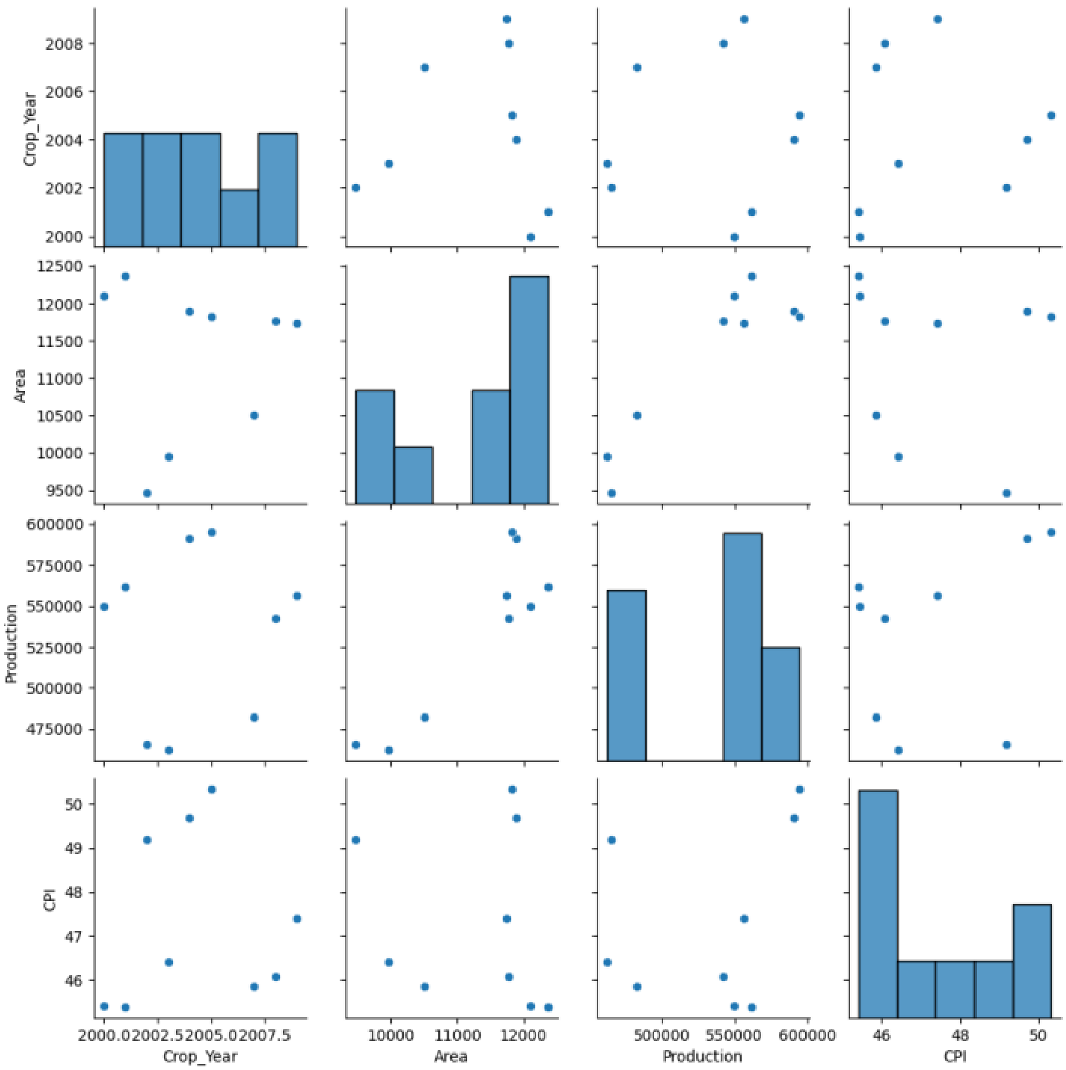
```
corrmat = data.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8,square=True);
```



In [20]:

```
cols = ['Crop_Year', 'Area', 'Production', 'CPI']
sns.pairplot(data[cols], size = 2.5)
plt.show()
```

```
c:\Users\pavit\AppData\Local\Programs\Python\Python311\Lib\site-packages\s
eaborn\axisgrid.py:2095: UserWarning: The `size` parameter has been rename
d to `height`; please update your code.
  warnings.warn(msg, UserWarning)
c:\Users\pavit\AppData\Local\Programs\Python\Python311\Lib\site-packages\s
eaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tigh
t
  self._figure.tight_layout(*args, **kwargs)
```



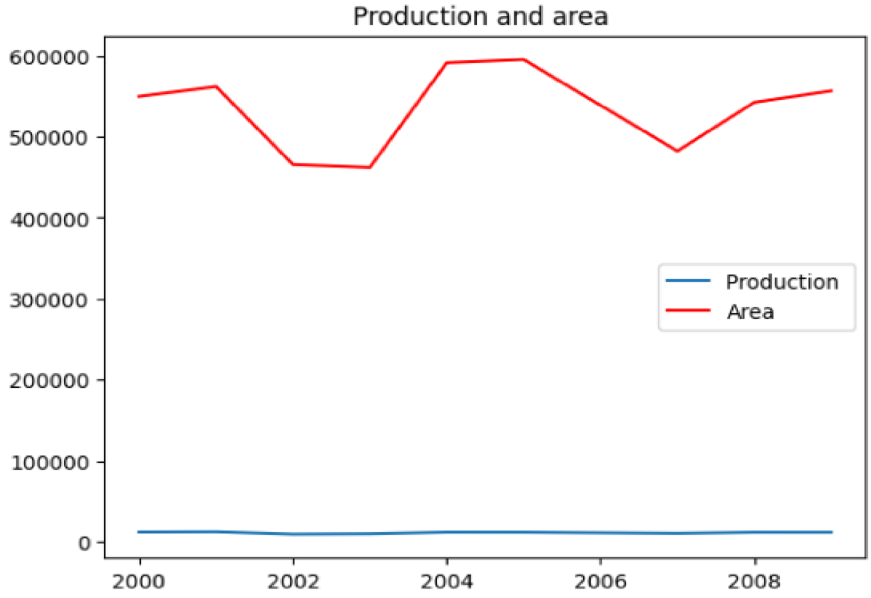
In [21]:

```
x_axis=data.Crop_Year
y_axis=data.Area

y1_axis=data.Production

plt.plot(x_axis,y_axis)
plt.plot(x_axis,y1_axis,color='r')

plt.title("Production and area ")
plt.legend(["Production ","Area"])
plt.show()
```



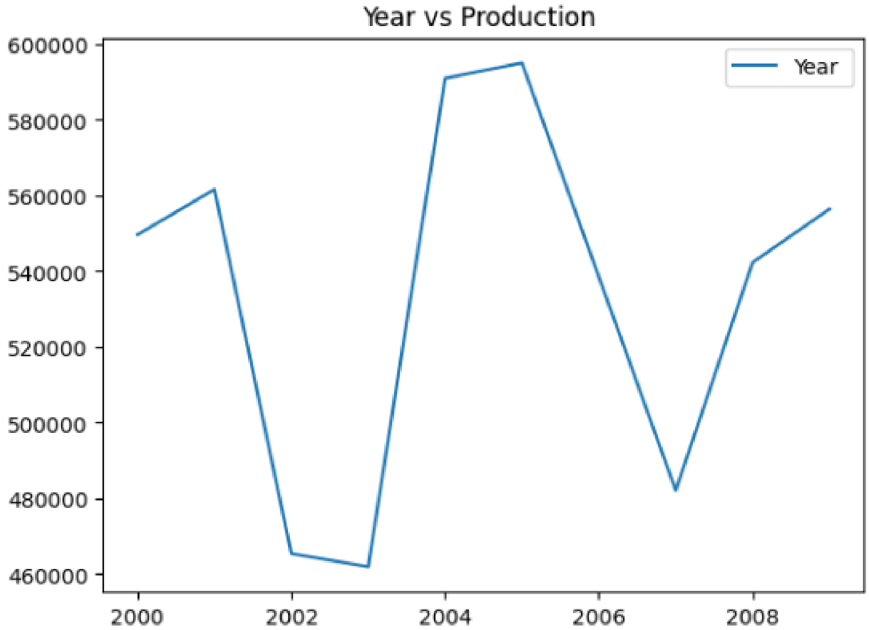
In [22]:

```
x_axis=data.Crop_Year
y1_axis=data.Production



plt.plot(x_axis,y1_axis)

plt.title("Year vs Production ")
plt.legend(["Year ","Production"])
plt.show()
```



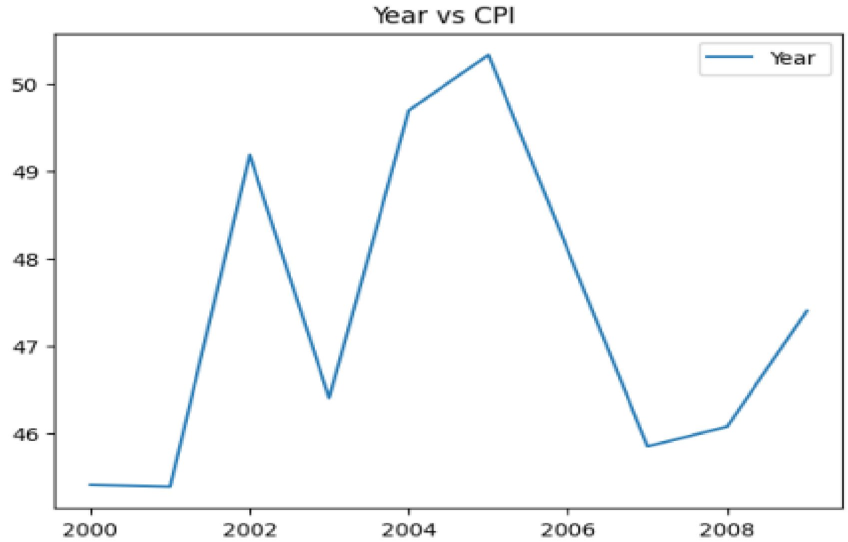
In [23]:

```
x_axis=data.Crop_Year
y1_axis=data.CPI



plt.plot(x_axis,y1_axis)

plt.title("Year vs CPI ")
plt.legend(["Year ","CPI"])
plt.show()
```



In [24]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
```

In [25]:

```
x=data.iloc[:,0:1].values
y=data.iloc[:,3].values
x_test,x_train,y_test,y_train=train_test_split(x,y,test_size=0.2,random_state=123)
print(x_test)
print(y_test)
print(x_train)
print(y_train)
regressor=RandomForestRegressor(n_estimators=12,random_state=0,n_jobs=1,verbose=13)

regressor.fit(x_test,y_test)
```

```
[[2005]
 [2007]
 [2003]
 [2001]
 [2009]
 [2004]
 [2002]]
[50.32817651 45.85619088 46.40848183 45.39582669 47.40459874 49.69381378
 49.18451852]
[[2008]
 [2000]]
```

```
[46.08106655 45.41766068]
building tree 1 of 12
building tree 2 of 12
building tree 3 of 12
building tree 4 of 12
building tree 5 of 12
building tree 6 of 12
building tree 7 of 12
building tree 8 of 12
building tree 9 of 12
building tree 10 of 12
building tree 11 of 12
building tree 12 of 12

[Parallel(n_jobs=1)]: Done   1 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   2 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   3 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   4 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   5 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   6 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   7 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   8 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   9 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done  10 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done  11 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done  12 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done  12 tasks      | elapsed:    0.0s
```

Out[25]:

```
RandomForestRegressor(n_estimators=12, n_jobs=1, random_state=0, verbose=1
3)
```

In [26]:

```
y_pred=regressor.predict(x_test)
y_pred
```

```
[Parallel(n_jobs=1)]: Done   1 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   2 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   3 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   4 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   5 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   6 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   7 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   8 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done   9 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done  10 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done  11 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done  12 tasks      | elapsed:    0.0s
[Parallel(n_jobs=1)]: Done  12 tasks      | elapsed:    0.0s
```

Out[26]:

```
array([49.69477621, 47.55205717, 47.46115121, 45.79593894, 47.20826469,
       48.55626186, 47.69028485])
```

In [27]:

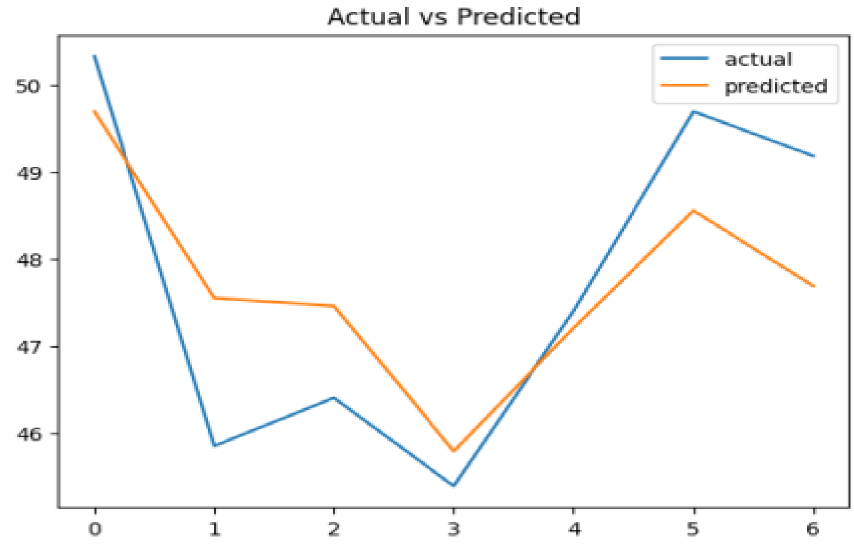
```
import numpy as np
corr_matrix = np.corrcoef(y_test, y_pred)
corr = corr_matrix[0,1]
R_sq = corr**2

print(R_sq)
```

```
0.7122528478179315
```

In [28]:

```
dm = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred}).reset_index()
x_axis=dm.index
y_axis=dm.Actual
y1_axis=dm.Predicted
plt.plot(x_axis,y_axis)
plt.plot(x_axis,y1_axis)
plt.title("Actual vs Predicted")
plt.legend(["actual ","predicted"])
b=plt.show()
b
```



In [30]:

```
df1=df.groupby('Crop')[['Area','Production']].mean()
df2=df1.reset_index(level=0, inplace=False)
df2.head(9)
```

Out[30]:

	Crop	Area	Production
0	Apple	2.250000	0.000000
1	Arcanut (Processed)	7205.800000	9641.550000
2	Arecanut	3812.309880	13229.253355
3	Arhar/Tur	7626.225417	5261.020643
4	Ash Gourd	37.363636	0.000000
5	Atcanut (Raw)	7205.800000	46362.500000
6	Bajra	26007.150175	24108.755531
7	Banana	1635.907893	46643.051274
8	Barley	2478.712643	5368.869514

In [31]:

```
df2['CPI']=df2['Production']/df2['Area']
df2.head(9)
```

Out[31]:

	Crop	Area	Production	CPI
0	Apple	2.250000	0.000000	0.000000
1	Arcanut (Processed)	7205.800000	9641.550000	1.338026
2	Arecanut	3812.309880	13229.253355	3.470141
3	Arhar/Tur	7626.225417	5261.020643	0.689859
4	Ash Gourd	37.363636	0.000000	0.000000
5	Atcanut (Raw)	7205.800000	46362.500000	6.434053
6	Bajra	26007.150175	24108.755531	0.927005
7	Banana	1635.907893	46643.051274	28.512028
8	Barley	2478.712643	5368.869514	2.165991

In [32]:

```
import pandas as pd
data1=pd.DataFrame({"predictedvalue":y_pred})
data1
```

Out[32]:

	predictedvalue
0	49.694776
1	47.552057
2	47.461151
3	45.795939
4	47.208265
5	48.556262
6	47.690285

In [33]:

```
df2['Predicted value'] = data1
df2.head(9)
```

Out[33]:

	Crop	Area	Production	CPI	Predicted value
0	Apple	2.250000	0.000000	0.000000	49.694776
1	Arcanut (Processed)	7205.800000	9641.550000	1.338026	47.552057
2	Arecanut	3812.309880	13229.253355	3.470141	47.461151
3	Arhar/Tur	7626.225417	5261.020643	0.689859	45.795939
4	Ash Gourd	37.363636	0.000000	0.000000	47.208265
5	Atcanut (Raw)	7205.800000	46362.500000	6.434053	48.556262
6	Bajra	26007.150175	24108.755531	0.927005	47.690285
7	Banana	1635.907893	46643.051274	28.512028	NaN
8	Barley	2478.712643	5368.869514	2.165991	NaN

In [34]:

```
df2[df2['CPI']==df2['Predicted value']]
df2.Crop.head()
```

Out[34]:

```
0                  Apple
1    Arcanut (Processed)
2               Arecanut
3              Arhar/Tur
4              Ash Gourd
Name: Crop, dtype: object
```

Corp Yield Estimation

For estimating crop yields, a random forest regression model was developed using Scikit-Learn. The random forest's capacity to capture complex interactions within the data was a decisive factor. Model evaluation was conducted using the Mean Absolute Error (MAE) metric.

In [5]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [6]:

```
df=pd.read_csv('Crop_recommendation.csv')
df
```

Out[6]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

2200 rows × 8 columns

In [7]:

```
len(df)
```

Out[7]:

```
2200
```

In [8]:

```
df.isnull().count()
```

Out[8]:

```
N              2200
P              2200
K              2200
temperature    2200
humidity       2200
ph             2200
rainfall       2200
label          2200
dtype: int64
```

In [9]:

```
df['N'].nunique()
```

Out[9]:

137

In [10]:

```
unique_values = df['N'].unique()
unique_values
```

Out[10]:

```
array([ 90,  85,  60,  74,  78,  69,  94,  89,  68,  91,  93,  77,  88,
        76,  67,  83,  98,  66,  97,  84,  73,  92,  95,  99,  63,  62,
        64,  82,  79,  65,  75,  71,  72,  70,  86,  61,  81,  80, 100,
        87,  96,  40,  23,  39,  22,  36,  32,  58,  59,  42,  28,  43,
        27,  50,  25,  31,  26,  54,  57,  49,  46,  38,  35,  52,  44,
        24,  29,  20,  56,  37,  51,  41,  34,  30,  33,  47,  53,  45,
        48,  13,   2,  17,  12,   6,  10,  19,  11,  18,  21,  16,   9,
         1,   7,   8,   0,   3,   4,   5,  14,  15,  55, 105, 108, 118,
       101, 106, 109, 117, 114, 110, 112, 111, 102, 116, 119, 107, 104,
       103, 120, 113, 115, 133, 136, 126, 121, 129, 122, 140, 131, 135,
       123, 125, 139, 132, 127, 130, 134], dtype=int64)
```

In [11]:

```
df.columns
```

Out[11]:

```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'labe
l'], dtype='object')
```

In [12]:

```
average_per_column = df['N'].mean()
print(average_per_column)
column_std = df['N'].std()
print(column_std)
```

```
50.551818181818184
36.9173338337566
```

In [13]:

```
threshold=2
outliers = df[(df['N'] -average_per_column).abs() > threshold * column_std]
print(outliers)
```

	N	P	K	temperature	humidity	ph	rainfall	label
1900	133	47	24	24.402289	79.197320	7.231325	90.802236	cotton
1901	136	36	20	23.095956	84.862757	6.925412	71.295811	cotton
1903	133	47	23	24.887381	75.621372	6.827355	89.760504	cotton
1904	126	38	23	25.362438	83.632761	6.176716	88.436189	cotton
1905	126	50	19	24.694571	81.735888	6.628723	78.584944	cotton
1909	129	60	22	24.584531	79.124042	5.947449	71.946081	cotton
1912	140	38	15	24.147295	75.882986	6.021440	69.915635	cotton
1915	131	35	18	24.491126	82.244158	7.057693	64.029494	cotton
1916	135	43	16	23.479869	81.730491	6.720450	86.762879	cotton
1924	125	39	21	25.031496	82.212766	7.954629	95.019132	cotton
1927	131	49	22	25.498482	79.975158	7.306919	67.059619	cotton
1928	139	35	15	25.248679	83.463015	5.898293	86.555178	cotton
1932	125	60	17	24.143862	84.515913	6.785724	80.361470	cotton
1934	131	52	16	23.657241	84.476015	6.486068	88.544791	cotton
1941	132	41	22	24.291449	81.024534	7.810866	90.416946	cotton
1943	133	50	25	25.721800	81.196662	7.569455	99.931008	cotton
1944	127	37	18	24.876637	76.300504	7.041066	91.922347	cotton
1946	131	38	19	23.868140	75.683397	6.814342	90.454718	cotton
1950	140	40	17	22.727672	77.075981	6.006086	77.551763	cotton
1956	133	57	19	23.542347	75.982033	7.947011	84.125367	cotton
1957	129	47	20	24.412123	80.803438	6.281914	98.604574	cotton
1960	131	60	17	25.320237	81.794759	7.425041	83.465325	cotton
1965	130	59	19	25.072787	82.502579	6.520404	93.510427	cotton

```
1966   127   53   24     22.215070   76.178519   6.127940   70.405576   cotton
1967   134   52   18     23.964313   76.591759   7.994680   76.130906   cotton
1970   132   52   19     24.164023   76.743390   6.436692   61.946261   cotton
1974   136   36   24     22.744470   80.411985   7.597820   90.073266   cotton
1975   134   56   18     23.808346   83.919026   6.691268   70.973583   cotton
1978   140   45   15     25.530827   80.046628   5.801048   99.395572   cotton
1979   126   46   25     24.438474   81.698017   6.757458   60.796459   cotton
1985   129   43   16     25.550370   77.850556   6.732109   78.584885   cotton
1988   126   37   21     25.849973   84.168552   6.614486   77.034212   cotton
1991   131   56   20     22.008171   81.838961   7.762648   92.236452   cotton
```

In [14]:

```
thershold=3
lower_outliers = df[df['N'] < average_per_column - threshold * column_std]
lower_outliers
```

Out[14]:

N	P	K	temperature	humidity	ph	rainfall	label

In [15]:

```
df= df[df['N'] >= 10]
```

In [16]:

```
df
```

Out[16]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

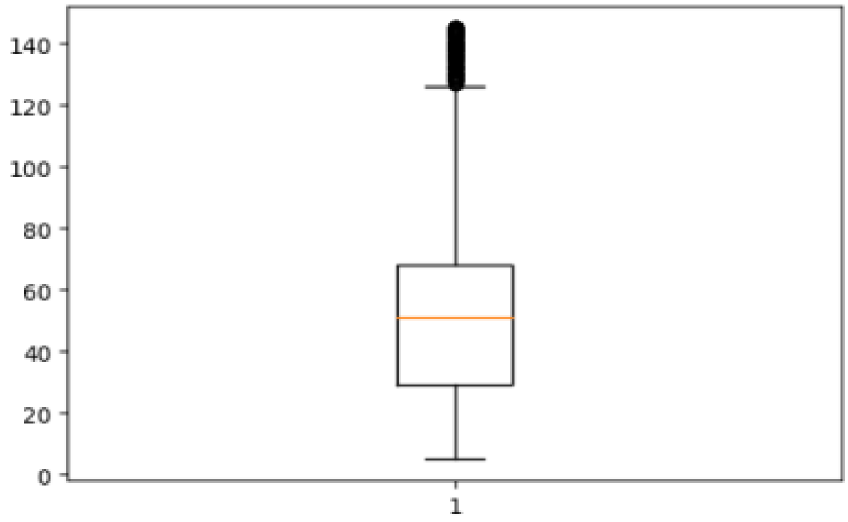
1936 rows × 8 columns

In [17]:

```
plt.figure(figsize=(6, 4))  # Optional: Adjust the figure size
plt.boxplot(df['P'])
```

Out[17]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1f981c0add0>,
  <matplotlib.lines.Line2D at 0x1f981c18310>],
 'caps': [<matplotlib.lines.Line2D at 0x1f981bd4290>,
  <matplotlib.lines.Line2D at 0x1f981c19d10>],
 'boxes': [<matplotlib.lines.Line2D at 0x1f981c0aa10>],
 'medians': [<matplotlib.lines.Line2D at 0x1f981c1a8d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1f981c1b410>],
 'means': []}
```



In [18]:

```
df= df[(df['P'] > 20) & (df['P'] <= 100)]
df
```

Out[18]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...	...	...	...
2193	116	38	34	23.292503	50.045570	6.020947	183.468585	coffee
2194	97	35	26	24.914610	53.741447	6.334610	166.254931	coffee
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee

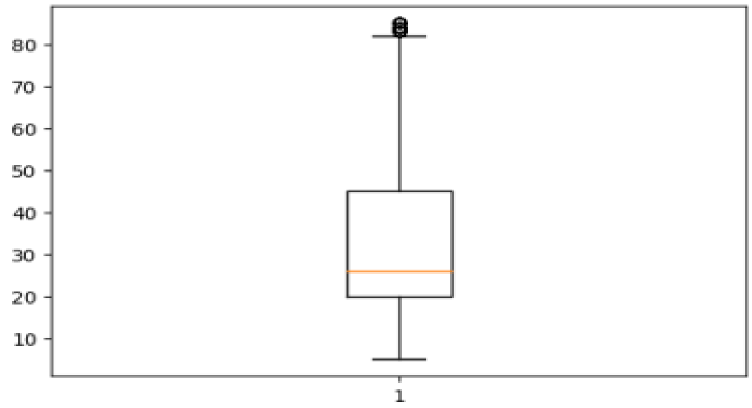
1493 rows × 8 columns

In [19]:

```
plt.figure(figsize=(6, 4))  # Optional: Adjust the figure size
plt.boxplot(df['K'])
```

Out[19]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1f983ccca10>,
  <matplotlib.lines.Line2D at 0x1f983ccd650>],
 'caps': [<matplotlib.lines.Line2D at 0x1f983cce2d0>,
  <matplotlib.lines.Line2D at 0x1f983ccedd0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1f983cc7d50>],
 'medians': [<matplotlib.lines.Line2D at 0x1f983ccf9d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1f983cd8510>],
 'means': []}
```



In [20]:

```
df= df[(df['P'] >= 20) & (df['P'] <= 50)]
df
```

Out[20]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
5	69	37	42	23.058049	83.370118	7.073454	251.055000	rice
11	90	46	42	23.978982	81.450616	7.502834	250.083234	rice
...	...	...	...	...	...	...	...	...
2193	116	38	34	23.292503	50.045570	6.020947	183.468585	coffee
2194	97	35	26	24.914610	53.741447	6.334610	166.254931	coffee
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee

663 rows × 8 columns

In [21]:

```
df.tail(5)
```

Out[21]:

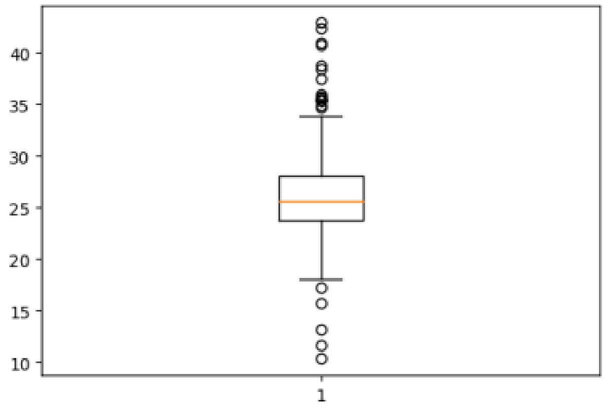
	N	P	K	temperature	humidity	ph	rainfall	label
2193	116	38	34	23.292503	50.045570	6.020947	183.468585	coffee
2194	97	35	26	24.914610	53.741447	6.334610	166.254931	coffee
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee

In [22]:

```
plt.figure(figsize=(6, 4))  # Optional: Adjust the figure size
plt.boxplot(df['temperature'])
```

Out[22]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1f983d36490>,
  <matplotlib.lines.Line2D at 0x1f983d37110>],
 'caps': [<matplotlib.lines.Line2D at 0x1f983d37c90>,
  <matplotlib.lines.Line2D at 0x1f983d40790>],
 'boxes': [<matplotlib.lines.Line2D at 0x1f983d35890>],
 'medians': [<matplotlib.lines.Line2D at 0x1f983d41310>],
 'fliers': [<matplotlib.lines.Line2D at 0x1f981c5d010>],
 'means': []}
```



In [23]:

```
df
```

Out[23]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
5	69	37	42	23.058049	83.370118	7.073454	251.055000	rice
11	90	46	42	23.978982	81.450616	7.502834	250.083234	rice
...	...	...	...	...	...	...	...	...
2193	116	38	34	23.292503	50.045570	6.020947	183.468585	coffee
2194	97	35	26	24.914610	53.741447	6.334610	166.254931	coffee
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee

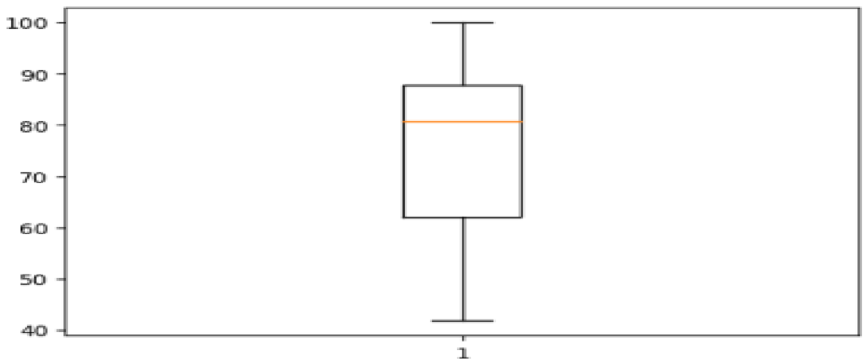
663 rows × 8 columns

In [24]:

```
plt.figure(figsize=(6, 4))  # Optional: Adjust the figure size
plt.boxplot(df['humidity'])
```

Out[24]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1f981ba54d0>,
  <matplotlib.lines.Line2D at 0x1f983da8d10>],
 'caps': [<matplotlib.lines.Line2D at 0x1f983da9850>,
  <matplotlib.lines.Line2D at 0x1f983daa2d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1f983d9f450>],
 'medians': [<matplotlib.lines.Line2D at 0x1f983daae10>],
 'fliers': [<matplotlib.lines.Line2D at 0x1f983dab810>],
 'means': []}
```

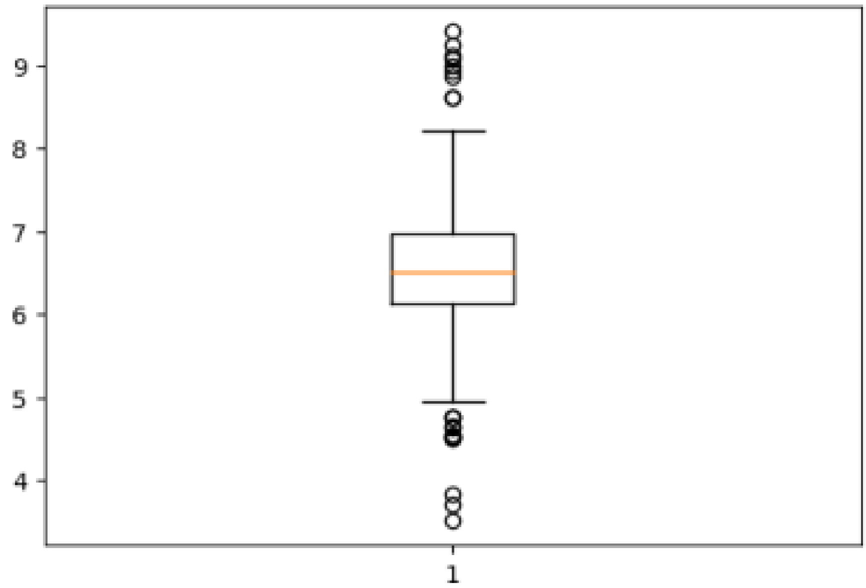


In [25]:

```
plt.figure(figsize=(6, 4))  # Optional: Adjust the figure size
plt.boxplot(df['ph'])
```

Out[25]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1f983e196d0>,
  <matplotlib.lines.Line2D at 0x1f983e045d0>],
 'caps': [<matplotlib.lines.Line2D at 0x1f983e1acd0>,
  <matplotlib.lines.Line2D at 0x1f983e1b850>],
 'boxes': [<matplotlib.lines.Line2D at 0x1f983e18990>],
 'medians': [<matplotlib.lines.Line2D at 0x1f983e1c410>],
 'fliers': [<matplotlib.lines.Line2D at 0x1f983e1a950>],
 'means': []}
```



In [28]:

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [29]:

```
X=df[['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']]
Y=df[['label']]
```

In [30]:

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.5,random_state=23)
```

In [31]:

```
X_test
```

Out[31]:

	N	P	K	temperature	humidity	ph	rainfall
2111	85	33	25	26.208114	52.509880	6.910824	189.094482
651	11	46	24	27.652802	89.806506	6.459252	56.525580
1404	95	26	45	29.916906	94.556956	6.117530	28.160572
609	21	39	20	28.144485	82.119305	7.064782	46.756901
39	63	44	41	24.172988	83.728757	5.583370	257.034355
...	...	...	...	...	...	...	...
167	73	45	21	24.605322	73.588685	6.636803	96.591953
1861	31	29	35	27.187228	92.199068	6.137103	141.322058
2136	84	27	29	23.322932	53.003663	7.167093	168.264429
1683	24	30	11	32.395240	94.517685	6.601396	113.253730
1942	103	42	17	24.294702	84.615276	6.527542	81.059023

332 rows × 7 columns

In [32]:

```
model=DecisionTreeClassifier()
```

In [33]:

```
model.fit(X_train,Y_train)
```

Out[33]:

```
DecisionTreeClassifier()
```

In [34]:

```
y_pred = model.predict(X_test)
y_pred
```

Web Connection

```
import streamlit as st
import pickle
st.cache_data
st.title('Crop Recommender')
model = pickle.load(open('model.pkl','rb'))
N=st.number_input("Enter N")
P=st.number_input("Enter P")
K=st.number_input("Enter K")
temperature=st.number_input("Enter temperature")
humidity=st.number_input("Enter humidity")
ph=st.number_input("Enter ph")
rainfall=st.number_input("Enter rainfall")
button =st.button("click for predection")
if button:
       y_pred = model.predict([[N,P, K, temperature, humidity, ph, rainfall]])
       st.write("the predicted crop can be grown is ")
       st.write(y_pred[0])
```

Chapter 5

RESULTS

In the outcomes of the crop and crop yield prediction project, we discuss the performance of the predictive models, their accuracy in recommending crops, and their ability to estimate yields accurately.

5.1. Results

Crop Recommendation

The crop prediction model's accuracy in recommending suitable crops based on environmental conditions can aid farmers in making informed decisions about crop selection. By suggesting crops that align with local conditions, we empower farmers to optimize their yield potential.

Yield Estimation

Accurate yield estimation is crucial for resource allocation and planning. The yield estimation model's low Mean Absolute Error (MAE) highlights its potential to provide reliable yield estimates, enabling farmers to manage resources efficiently and plan for storage and distribution.

5.2. Screenshots

```
In [45]: y_pred = model.predict(X_test)
         y_pred_reshaped = y_pred.reshape(-1, 1)

         accuracy = accuracy_score(Y_test, y_pred_reshaped)
         print("Accuracy:", accuracy)

         Accuracy: 0.9608433734939759
```

Fig.5.2.1. Crop Recommendation Accuracy

```
In [28]: from sklearn.metrics import mean_absolute_error

         mae = mean_absolute_error(y_test, y_pred)
         print("Mean Absolute Error:", mae)

         Mean Absolute Error: 0.9443096929588292

In [29]: from sklearn.metrics import r2_score

         r2 = r2_score(y_test, y_pred)
         print("R-squared (R2) score:", r2)

         R-squared (R2) score: 0.6553170117082417
```

Fig.5.2.2. Crop Yield Mean squared error and r2 score

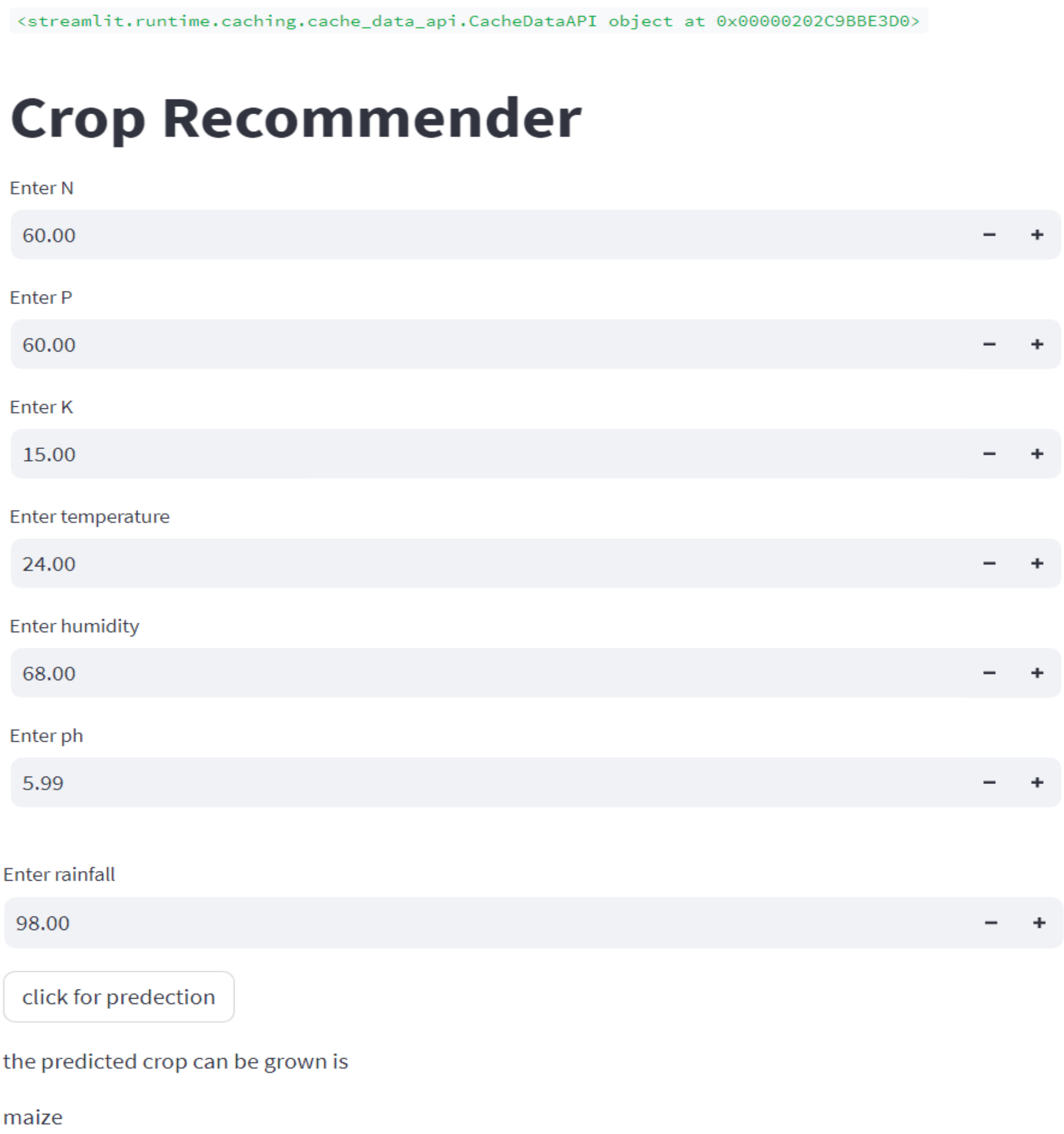


Fig.5.2.3. Crop Recommendation

Chapter 6

CONCLUSION AND LEARNING OUTCOMES

6.1. Conclusion

In conclusion, this project has successfully harnessed the power of machine learning to address critical challenges in agriculture. The implementation of predictive models for crop recommendation and yield estimation has yielded accurate results, providing farmers with valuable insights for optimized decision-making. The project's contribution extends to the deployment of these models as a practical web-based API, offering immediate accessibility to users. Through data-driven solutions and model deployment, this project has demonstrated the potential of technology to revolutionize agricultural practices.

6.2. Learning Outcomes

This project has been a transformative learning experience, equipping us with a profound understanding of data preprocessing, model development, and evaluation. Through hands-on exploration, we've honed technical skills, delved into real-world agricultural issues, and grasped the ethical considerations of data handling. Collaboration within the team has sharpened our teamwork and communication abilities, and we've learned to structure and convey complex insights through comprehensive project reporting. This journey has underscored the vital role of data-driven innovation in addressing industry challenges, fostering continuous growth in our knowledge and capabilities.

References

[1] Crop Yield Prediction using Machine Learning Algorithms

Link: https://www.ijert.org/crop-yield-prediction-using-machine-learning-algorithms

[2] Predicting Crops Yield: Machine Learning Nanodegree Capstone Project

Link: https://towardsdatascience.com/predicting-crops-yield-machine-learning□nanodegree-capstone-project-e6ec9349f69

[3] Crop yield prediction using machine learning techniques

Link: https://www.sciencedirect.com/science/article/abs/pii/S0965997822002277

[4] Agriculture Crop Yield Prediction Using Machine Learning

Link: https://www.ijraset.com/research-paper/agriculture-crop-yield-prediction-using□ml

[5] Design And Implementation Of Crop Yield Prediction Model In Agriculture

Link: https://www.researchgate.net/publication/344560876_Design_And_Implementation_Of_Crop_Yield_Prediction_Model_In_Agriculture