**AGRO HANDS - AN INTERACTIVE WEB APP FOR FARMERS**

*Report submitted to the SASTRA Deemed to be University*

*as the requirement for the course*

**CSE300 - MINI PROJECT**

*Submitted by*

**Name: Preethi Narayanan Reg No: 124156032**

**PROGRAMME: B.Tech Computer Science Engineering(AI & DS)**

**Name: Akash R Reg No: 124003021**

**PROGRAMME: B.Tech Computer Science Engineering**

**Name: Sairamraj S Reg No: 124003262**

**PROGRAMME: B.Tech Computer Science Engineering**

**May 2023**



**SCHOOL OF COMPUTING**

**THANJAVUR, TAMIL NADU, INDIA - 613401**



**SCHOOL OF COMPUTING**

**THANJAVUR-613401**

**Bonafide Certificate**

This is to certify that the report titled **“Agro Hands-An interactive web application for Farmers”** submitted as a requirement for the course, CSE300: MINI PROJECTfor B.Tech. is a bonafide record of the work done by **Ms. Preethi Narayanan(Reg. No: 124156032, CSE(Ai&Ds), Mr. Akash R(Reg. No: 124003262, CSE) andMr. Sairamraj S(Reg. No: 124003262, CSE)** during the academic year 2022-23, in the School of Computing

**Signature of Project Supervisor:**

**Name with Affiliation:**

**Date:**

Mini Project *Viva voce* held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Examiner 1 Examiner 2**

**Acknowledgment**

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing

us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam**

and **Dr. S. Swaminathan**, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to

be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. A. Umamakeswari,** Dean, School of Computing,

**Dr. S. Gopalakrishnan**, **Associate Dean**, Department of Computer Application, **Dr. B.Santhi**, Associate Dean, Research, **Dr. V. S. Shankar Sriram**, Associate Dean, Department of Computer Science and Engineering, **Dr. R. Muthaiah**, Associate Dean, Department of Information Technology and Information & Communication Technology.

Our guide **Dr K R Manjula,Associate Professor**, School of Computing was the driving force behind this whole idea from the start. Her profound insight into the field and invaluable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better. We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and

friends resulting in the successful completion of this project. We thank you all for allowing me to showcase my skills through this project.

**Abbreviations**

| **RF** | Random forest |
| --- | --- |
| **SVM** | Support Vector Machine |
| **RESNET** | Residual Network |
| **CNN** | Convolution Neural Network |
| **API** | Application programming interface |
| **PH** | Potential of Hydrogen |
| **ARIMA** | Auto-regressive integrated moving average |
| **MAP** | Mean absolute percentage error |

**List of Tables**

| **Table no.** | **Table Name** | **Page no.** |
| --- | --- | --- |
| 2.1 | Literature Survey | 12 |

**List of Figures**

| **Figure no** | **Figure name** | **Page no.** |
| --- | --- | --- |
| 1 | Workflow of all the modules | 10 |
| 2 | Home Page | 39 |
| 3 | Crop recommendation module(input) | 40 |
| 4 | Crop recommendation module(output) | 41 |
| 5 | Weed identification module | 41 |
| 6 | Pest identification module | 42 |
| 7 | Crop cost estimation module | 42 |
| 8 | Crop cost estimation using exponential smoothing model | 43 |
| 9 | Crop cost estimation using multivariate ARIMA model | 43 |
| 10 | Accuracy scores | 44 |
| 11 | plot for actual and forecasted values for the exponential smoothing model | 44 |
| 12 | Plot for actual and forecasted crop cost values | 45 |
| 13 | Mean Absolute Percentage Error for ARIMA Model | 45 |

**Abstract**

A nation's economic development is significantly influenced by its agricultural sector. Even though it has been practiced widely over the globe for years, traditional farming is inefficient and unsustainable. To increase productivity, reduce labor hours, and ensure effective fertiliser and irrigation management, an innovative concept known as precision agriculture has received widespread support. Precision agriculture is aided by contemporary technology like data mining, artificial intelligence, and data science.

AGRO HANDS is a web application. There are four modules: weed identification, pest identification, crop recommendations, and crop cost estimation. It has been created to access these four components via a web application. Along with hyperparameter optimization, machine learning, and deep learning methods are employed.

**Keywords**: Data mining, Artificial intelligence, Precision Agriculture

**TABLE OF CONTENTS**

**Title Page No.**

Bonafide Certificate ii

Acknowledgments iii

List of Figures iv

List of Tables v

Abbreviations vi

Notations vii

Abstract viii

1. Summary of the base paper 1

2. Merits and Demerits of the base paper 4

3. Source Code 7

4. Snapshots 30

5. Conclusion and Future Plans 40

6. References 41

7. Appendix -Base Paper 42

**CHAPTER 1: Summary of Base Paper**

**1.1 Base Paper Details:**

**Title:** “Smart Farming using Machine Learning and Deep Learning Techniques”.

**Authors:** “Senthil Kumar Swami Durai, Mary Divya Shamili”

**Publication year:** 2022

**Journal:** Decision Analytics

**1.2 Introduction:**

The process of preparing the soil for the growing of crops and the keeping of livestock is called farming. A nation's economic development is significantly influenced by its agricultural sector. Around 58 percent of a country's primary source of income comes from agriculture. For so long, farmers have relied on conventional farming techniques. The length and lack of precision of these procedures reduced production. Precise farming increases productivity by predicting the actions that must be conducted at the proper time of year. Numerous methods are used in precision farming, including soil analysis, crop suggestions, weather forecasts, and figuring out the right amounts of herbicides and fertilizers.

In this work, four modules—crop recommendation, weed identification, pesticide recommendation, and crop cost estimation—are proposed.

**1.2.1 Crop recommendation**

The Crop recommendation was used for the training model since it contains attributes such as temperature, humidity, average rainfall, soil Ph, nitrogen requirement ratio, potassium requirement ratio, and phosphorus requirement ratio essential for predicting a crop.

The optimal crop prediction model is chosen using three machine learning algorithms.

1. SVM
2. Random forest
3. Decision tree

Python object architectures are serialized using pickle files, which is a technique for turning an object in memory into a byte stream that can be saved to drive

**1.2.2 Weed Identification:**

In this module, a dataset containing different weed class images is passed to a pre-trained RESNET model resnet152v2.

**RESNET152v2**:It is an expansion of the initial RESNET-152 model with a deep CNN architecture. It is a component of the residual network series, which was created to address the vanishing gradient issue. There are 152 layers, including convolutional, batch normalization, and other layers.

Some key features include:

* Reduces computational complexity and improves accuracy through bottleneck architecture
* The pre-activation approach executes batch normalization and activation before implementing each convolutional layer.
* Global Average Pooling: The model averages the spatial dimensions of the feature map to produce a fixed-length feature vector, which is then supplied to the softmax classifier for classification, as opposed to fully connected layers at the end of the network.

The pre-trained layers are frozen so the weights don’t change throughout the execution.

After that, a file in the Hierarchical Data Format 5 (.h5) is created with the trained model.H5 is one of the HDFs (Hierarchical Data Formats) used to store a lot of data. Here huge amounts of data are stored in multidimensional arrays. For simple retrieval and analysis, well-organized scientific data is typically kept in this format. The module gets the image as input, and using that data, the model predicts the class of the image.

**Herbicides** are recommended based on the class of weed identified by the model.

**1.2.3 Pest Identification:**

In this module, a dataset containing different pest class images is passed to a pre-trained RESNET model resnet152v2. The model is trained and saved as a Hierarchical Data Format 5 file(.h5). The image is inputted into the module and the model predicts the corresponding class of the image.

**Pesticides** are recommended based on the class of pest identified by the model.

**1.2.4 Crop Cost Estimation:**

In this module, the cost for each crop is taken from 2016-2022. Using this time series data, the production cost of the particular crop is forecasted for user inputted period. Multivariate ARIMA model and Exponential smoothing are used for forecasting the data.

**1.3 Proposed Method (Architecture):**

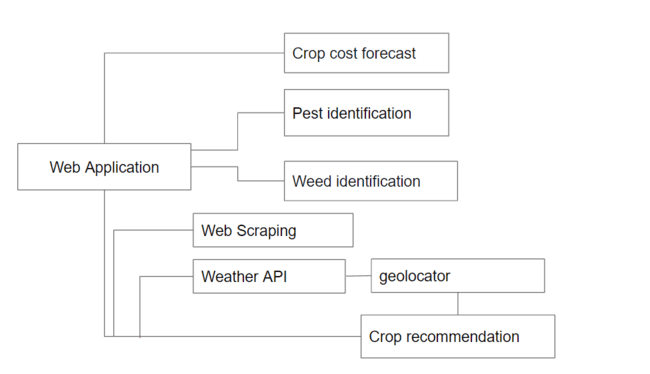


Fig 1 Workflow of all the modules

**1.4 Dataset:**

**1.4.1 Crop Recommendation Module:**

Three datasets are used - containing the temperature, humidity, average rainfall, NPK content ratio, and crop names saved as CSV files. [8 attributes and 2200 records ] <https://www.kaggle.com/atharvaingle/crop-recommendation-dataset>

**1.4.2 Weed identification module:**

An image dataset that contains images of various types of weeds[5539 image data]

<https://www.kaggle.com/vbookshelf/v2-plant-seedlings-dataset>

**1.4.3 Pest identification module:**

An image dataset that contains images of different pests[3150 image data]

<https://www.kaggle.com/simranvolunesia/pest-dataset>

**1.4.4 Crop cost estimation module:**

CSV files of the specific crop with annual cultivation crop cost

**1.5 Algorithms Used:**

In Crop Recommendation Module,3 algorithms are used to recommend the best crop to cultivate in the soil

**1.5.1 Random Forest Algorithm:**

It is a classifier that uses many decision trees on different subsets of the given dataset and averages the results to increase the dataset's predicted accuracy.

A well-known machine learning technique that uses the supervised learning approach.

**1.5.2 Decision Tree Algorithm:**

It is a tree-structured classifier, where each leaf node represents its outcome while the inner nodes represent the features of the dataset. It is a supervised learning technique that uses the graphical representation to find every possible solution based on the given condition.

**1.5.3 Support Vector Machine:**

The SVM's objective is to establish the best line or decision boundary that can divide n-dimensional space into classes so that subsequent data points can be quickly assigned to the appropriate category.

The main use of this supervised learning technique is classification.

**1.5.4 Residual Networks(ResNet):**

This architecture addresses the vanishing gradient issue by introducing residual blocks.

Regularisation will skip any layer that negatively affects architecture performance.

**1.5.5 Multivariate ARIMA:**

To accommodate many time series variables simultaneously, the ARIMA model is extended.

The relationship between the factors increases the accuracy of predictions.

Cross-dependencies between the series are included.

It is denoted as ARIMA(p,d,q)(P, D, Q), where (p,d,q) stands for the orders for auto-regression, differentiating, and moving average, and (P, D, Q) is the cross-dependencies between the series components.

**1.5.6 Double Exponential Smoothing:**

It is called Holt's approach as well. It is used in time series analysis to make predictions about a series' future values based on historical data.

It makes use of a trend component (rate of change in the series) and a level component (smoothed value). The estimated values are predicated on the idea that they will be more relevant to the recent observation than to earlier observations.

**Chapter-2 MERITS AND DEMERITS OF THE BASE PAPER**

**2.1 Literature Survey:**

Table 2.1 Literature Survey

| **Sno** | **Paper title** | **Published year** | **Authors** | **Paper lags** |
| --- | --- | --- | --- | --- |
| 1 | A Comprehensive Review of Crop Yield Prediction Using Machine Learning Approaches With Special Emphasis on Palm Oil Yield Prediction | April 2021 | Mamunur Rashid, Yusri Yusup, Nuzat Khan | The model accuracy will be less as the parameters of the algorithms used are not hyper-tuned. |
| 2 | Multilayer Convolutional Neural Network for the Classification of Mango Leaves Infected by Anthracnose Disease | March 2019 | Uday Pratap Singh, Siddharth Singh Chauhan | Inspired by AlexNet CNN architecture, the performance of the model will be less than compared to the model using ResNet architecture |
| 3 | Weed Identification Using Deep Learning and Image Processing in Vegetable Plantation | January 2021 | Xiaojun Jin, Jun Che | Proposes a method that combines deep learning and image processing technology. |
| 4 | Improvement of Crop Production Using Recommender System by Weather Forecasts | March 2019 | Uday Pratap Singh, Siddharth Singh Chauhan | Predictive analysis to analyze the best crop suggests a hybrid recommender system that adopts Case-Based Reasoning for enhancing the success ratio of the system. |
| 5 | Smart Agriculture Applications Using Deep Learning Technologies: A Survey | June 2022 | Maha Altalak,n Mohammad Ammad uddin | Proposed an intelligent agriculture system based on a hybrid model of CNN and SVM, capable of detecting and classifying plant leaves disease early. |

**2.2 Merits:**

* Deep learning techniques are used for weed and pest identification.
* Removal of agricultural weeds and pests enhances crop growth.

**2.3 Demerits**

* The cost module is not implemented properly
* In the crop recommendation module, for the location it has been asked to web scrape from a website called <https://ipinfo.io/> but it's not showing the correct location all the time
* In the crop recommendation module, they just get live location temperature and humidity values but it changes over time and will not give correct crop recommendation.
* In the crop cost estimation module, forecasting was not done instead the values were duplicated and displayed for each year.

**2.4 Novelty:**

* In the crop recommendation module,
* In the weed identification module, a suitable **herbicide** is recommended to inhibit the growth of agricultural weeds.
* In the pest identification module, a suitable **pesticide** is recommended to deter the growth of pests.
* **Multivariate ARIMA** model and **Exponential Smoothing** model are used to forecast the crop cost for the user inputted period.

**CHAPTER- 4 SOURCE CODE**

**Source Code:**

import numpy as np

from flask import Flask,request,render\_template

import pickle

import requests, json

import csv

import numpy as np

from flask import Flask,request,render\_template

import pickle

import requests, json

import csv

import requests

from bs4 import BeautifulSoup

import pandas as pd

from selenium import webdriver

import urllib.request

from pprint import pprint

from html\_table\_parser.parser import HTMLTableParser

import pandas as pd

from geopy.geocoders import Nominatim

from flask import Flask,render\_template,request,redirect,flash

from werkzeug.utils import secure\_filename

from main1 import getPrediction

from main import getPrediction1

import os

from geopy.geocoders import Nominatim

import numpy as np

from flask import Flask,request,render\_template

from flask import Flask,render\_template,request,redirect,flash

import pickle

import requests, json

import csv

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.holtwinters import ExponentialSmoothing

UPLOAD\_FOLDER = 'static/images/'

app = Flask(\_\_name\_\_,static\_folder="static")

app.secret\_key = "secret key"

app.config['UPLOAD\_FOLDER']= UPLOAD\_FOLDER

des = pickle.load(open('models\DS.pkl','rb'))

svm = pickle.load(open('models\SVM.pkl','rb'))

rf = pickle.load(open('models\RF.pkl','rb'))

desfull = pickle.load(open('models\dsfull.pkl','rb'))

svmfull = pickle.load(open('models\svmfull.pkl','rb'))

rff = pickle.load(open("models\s1.pkl",'rb'))

@app.route('/')

def home():

return render\_template('home1.html')

@app.route('/crop',methods=['get'])

def crop():

return render\_template("crop.html")

@app.route('/wnpk',methods=['get'])

def wnpk():

return render\_template('index.html')

@app.route('/npk',methods=['get'])

def npk():

return render\_template('index2.html')

@app.route('/pest',methods=['get'])

def pest():

return render\_template('pest.html')

@app.route('/cost',methods=['get'])

def cost():

return render\_template('cost.html')

@app.route('/weed',methods=['get'])

def weed():

return render\_template('weed.html')

@app.route('/predict',methods=['post'])

def predict():

#int\_features=[float(x) for x in request.form.values()]

#features= [np.array(int\_features)]

# temp=float(request.form['temp'])

# hum=float(request.form['hum'])

city\_name =request.form['city']

# initialize Nominatim API

geolocator = Nominatim(user\_agent="project")

from selenium.webdriver.chrome.options import Options

# Configure Chrome options

chrome\_options = Options()

chrome\_options.headless = True

PATH\_TO\_DRIVER = "C:\Program Files (x86)\chromedriver.exe"

driver = webdriver.Chrome(PATH\_TO\_DRIVER)

driver.get("https://en.climate-data.org/search/?q="+city\_name)

search\_results = driver.find\_element("class name", "search\_results")

link = search\_results.find\_element("tag name", "a").get\_attribute("href")

print(link)

driver.quit()

# Make a request to the website

url = link

# Define headers to make the request look like it is coming from a web browser

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}

# Make a request to the website with the headers included

req = urllib.request.Request(url, headers=headers)

f = urllib.request.urlopen(req)

# Read the contents of the website

xhtml = f.read().decode('utf-8')

# Define the HTMLTableParser object and feed the HTML contents into it

p = HTMLTableParser()

p.feed(xhtml)

# Print the table data

pprint(p.tables[1])

# Convert the parsed data to a Pandas DataFrame

df = pd.DataFrame(p.tables[1])

print("\n\nPANDAS DATAFRAME\n")

print(df)

df.to\_excel('D:\Mini project\scrap\output.xlsx', index=False)

df = pd.read\_excel('D:\Mini project\scrap\output.xlsx')

i=int(request.form['mon'])

hum=df.iloc[5][i]

temp=df.iloc[1][i]

temp = float(temp.split(' ')[0])

base\_url = "http://api.openweathermap.org/data/2.5/weather?"

complete\_url = base\_url + "appid=" + "98a25fbf985ed7da9eb47274c9d2d495" + "&q=" + city\_name

response = requests.get(complete\_url)

x = response.json()

if x["cod"] != "404":

y = x["main"]

current\_temperature = y["temp"]

c = current\_temperature - 273.15

print(c)

ch = y["humidity"]

print(ch)

z=x["coord"]

print(type(z['lat']))

print(type(z['lon']))

# Latitude & Longitude input

Latitude = str(z['lat'])

Longitude =str(z['lon'])

location = geolocator.reverse(Latitude +","+ Longitude)

address = location.raw['address']

# traverse the data

city = address.get('city', '')

state = address.get('state', '')

country = address.get('country', '')

code = address.get('country\_code')

zipcode = address.get('postcode')

print('City : ', city)

print('State : ', state)

print('Country : ', country)

print('Zip Code : ', zipcode)

with open('pin.csv','r') as csvfile:

reader=csv.reader(csvfile)

for row in reader:

if zipcode in row:

dist=row[7]

print("District : " + dist)

with open('rain.csv','r') as csvfile:

reader=csv.reader(csvfile)

for row in reader:

if dist.upper() in row:

avgrf=float(row[1])/12

print("Average rainfall is = " + str(avgrf))

else:

print(" City Not Found ")

ph=request.form['ph']

start, end = map(float, ph.split("-"))

# features=np.array([[temp,hum,ph,avgrf]])

#prediction=des.predict(features)

#prediction1=svm.predict(features)

#prediction3=rf.predict(features)

for i in range(int(start\*10), int(end\*10)+1):

features=np.array([[temp,hum,(i/10),avgrf]])

prediction=des.predict(features)

prediction1=svm.predict(features)

prediction3=rf.predict(features)

all\_predictions = np.concatenate((prediction, prediction1, prediction3))

# Convert the array into a set of strings

prediction\_set = set(map(str, all\_predictions))

# Convert the set into a string

prediction\_string = ', '.join(prediction\_set)

# Print the resulting string

print(prediction\_string)

return render\_template('index.html',prediction\_text1=' The set of crops recommened are :{} '.format(prediction\_string))

#return render\_template('index.html',prediction\_text1=' Support Vector Machine {}'.format(prediction1),prediction\_text='Decision Tree {} '.format(prediction),prediction\_text3=' Random Forest(Highest Accuracy){} '.format(prediction3))

@app.route('/predict1',methods=['post'])

def predict1():

city\_name =request.form['city']

# initialize Nominatim API

geolocator = Nominatim(user\_agent="project1")

PATH\_TO\_DRIVER = "C:\Program Files (x86)\chromedriver.exe"

driver = webdriver.Chrome(PATH\_TO\_DRIVER)

driver.get("https://en.climate-data.org/search/?q="+city\_name)

search\_results = driver.find\_element("class name", "search\_results")

link = search\_results.find\_element("tag name", "a").get\_attribute("href")

print(link)

driver.quit()

# Make a request to the website

url = link

# Define headers to make the request look like it is coming from a web browser

headers = {

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}

# Make a request to the website with the headers included

req = urllib.request.Request(url, headers=headers)

f = urllib.request.urlopen(req)

# Read the contents of the website

xhtml = f.read().decode('utf-8')

# Define the HTMLTableParser object and feed the HTML contents into it

p = HTMLTableParser()

p.feed(xhtml)

# Print the table data

pprint(p.tables[1])

# Convert the parsed data to a Pandas DataFrame

df = pd.DataFrame(p.tables[1])

print("\n\nPANDAS DATAFRAME\n")

print(df)

df.to\_excel('D:\Mini project\scrap\output.xlsx', index=False)

df = pd.read\_excel('D:\Mini project\scrap\output.xlsx')

i=int(request.form['mon'])

hum=df.iloc[5][i]

temp=df.iloc[1][i]

temp = float(temp.split(' ')[0])

base\_url = "<http://api.openweathermap.org/data/2.5/weather>?"

complete\_url = base\_url + "appid=" + "98a25fbf985ed7da9eb47274c9d2d495" + "&q=" + city\_name

response = requests.get(complete\_url)

x = response.json()

if x["cod"] != "404":

y = x["main"]

current\_temperature = y["temp"]

c = current\_temperature - 273.15

print(c)

ch = y["humidity"]

print(ch)

z=x["coord"]

print(type(z['lat']))

print(type(z['lon']))

# Latitude & Longitude input

Latitude = str(z['lat'])

Longitude =str(z['lon'])

location = geolocator.reverse(Latitude +","+ Longitude)

address = location.raw['address']

# traverse the data

city = address.get('city', '')

state = address.get('state', '')

country = address.get('country', '')

code = address.get('country\_code')

zipcode = address.get('postcode')

print('City : ', city)

print('State : ', state)

print('Country : ', country)

print('Zip Code : ', zipcode)

with open('pin.csv','r') as csvfile:

reader=csv.reader(csvfile)

for row in reader:

if zipcode in row:

dist=row[7]

print("District : " + dist)

with open('rain.csv','r') as csvfile:

reader=csv.reader(csvfile)

for row in reader:

if dist.upper() in row:

avgrf=float(row[1])/12

print("Average rainfall is = " + str(avgrf))

else:

print(" City Not Found ")

n=float(request.form['n'])

p=float(request.form['p'])

k=float(request.form['k'])

print(n,p,k)

ph=request.form['ph']

start, end = map(float, ph.split("-"))

# features=np.array([[temp,hum,ph,avgrf]])

#prediction=des.predict(features)

#prediction1=svm.predict(features)

#prediction3=rf.predict(features)

for i in range(int(start\*10), int(end\*10)+1):

feat=np.array([[n,p,k,temp,hum,(i/10),avgrf]])

predi=desfull.predict(feat)

predi1=svmfull.predict(feat)

predi3=rff.predict(feat)

all\_predictions = np.concatenate((predi, predi1, predi3))

# Convert the array into a set of strings

prediction\_set = set(map(str, all\_predictions))

# Convert the set into a string

prediction\_string = ', '.join(prediction\_set)

# Print the resulting string

print(prediction\_string)

return render\_template('index2.html',prediction\_text1=' The set of crops recommened are :{} '.format(prediction\_string))

@app.route('/wee' ,methods=['POST'])

def wee():

if request.method=='POST':

if 'file' not in request.files:

flash('No file part')

return redirect(request.url)

file=request.files['file']

if file.filename=='':

flash('No file selected for uploading')

return redirect(request.url)

if file:

filename = secure\_filename(file.filename) #Use this werkzeug method to secure filename.

file.save(os.path.join(app.config['UPLOAD\_FOLDER'],filename))

#getPrediction(filename)

label = getPrediction1(filename)

print(label)

name=label

rec=""

if name == "Black-grass":

rec="use Flufenacet and pendimethalin"

elif name == "Small-flowered Cranesbill":

rec=" use clomazone+napropamide"

elif name == "Sugar beet":

rec=" use paraquat (Gramoxone SL 2.0)"

elif name == "Shepherd’s Purse":

rec= " use 2, 4-D, MCPP (mecoprop), Dicamba\*, or Triclopyr"

elif name == "Scentless Mayweed":

rec=" use bromoxynil (Buctril®) and dicambda (Clarity®) "

elif name == "Maize":

rec=" use Sempra"

elif name == "Loose Silky-bent":

rec=" use Apera spica-venti (L.) P.B"

elif name == "Fat Hen":

rec=" use Roundup Fast Action or Weedol Fast Acting Weedkiller."

elif name == "Common wheat":

rec=" use Isoproturon 800 g/ha "

elif name == "Common Chickweed":

rec=" use 41% Glyphosate "

elif name == "Cleavers":

rec=" use amidosulfuron, florasulam, fluroxypyr, mecoprop"

else:

rec="use Sulfonylurea herbicides (e.g. metsulfuron), triazolopyrimidines "

flash("The Identified Weed is " + label )

flash("The possible weedicide : " + rec)

full\_filename = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

flash(full\_filename)

return redirect('/weed')

@app.route('/pes',methods=['POST'])

def submit\_file():

if request.method=='POST':

if 'file' not in request.files:

flash('No file part')

return redirect(request.url)

file=request.files['file']

if file.filename=='':

flash('No file selected for uploading')

return redirect(request.url)

if file:

filename = secure\_filename(file.filename) #Use this werkzeug method to secure filename.

file.save(os.path.join(app.config['UPLOAD\_FOLDER'],filename))

#getPrediction(filename)

label = getPrediction(filename)

print(label)

name=label

rec=""

if name == "aphids":

rec="insecticidal soaps and oils"

elif name == "armyworm":

rec="spinosad, bifenthrin, cyfluthrin, and cypermethrin"

elif name == "beetle":

rec="Pyrethrin "

elif name == "bollworm":

rec=" chlorpyrifos, methomyl, and lambda-cyhalothrin. "

elif name == "grasshopper":

rec=" carbaryl"

elif name == "mites":

rec="Azobenzene, dicofol, ovex, and tetradifon "

elif name == "mosquito":

rec=" Bifen IT "

elif name == "sawfly":

rec=" permethrin, bifenthrin, lambda cyhalothrin, and carbaryl "

else:

rec="Neem seed kernel extract 5% or Azadirachtin 0.03% 400 ml/ac"

flash("The Identified Pest is " + label )

flash("The possible pesticide : " + rec)

full\_filename = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

flash(full\_filename)

return redirect('/pest')

@app.route('/calc',methods=['post'])

def calc():

crop =request.form['cropname']

n=float(request.form['years'])

if crop=='wheat':

df=pd.read\_csv('wheat.csv', parse\_dates=['Time'], index\_col='Time')

elif crop=='maize':

df=pd.read\_csv('maize.csv', parse\_dates=['Time'], index\_col='Time')

elif crop=='rice':

df=pd.read\_csv('rice.csv', parse\_dates=['Time'], index\_col='Time')

# define alpha and beta values

alpha = 0.8

beta = 0.2

# apply exponential smoothing

fit = ExponentialSmoothing(df['Value'], trend='add', seasonal=None, initialization\_method="estimated").fit(smoothing\_level=alpha, smoothing\_slope=beta)

# predict future values

future\_years = pd.date\_range(start=df.index[-1], periods=n, freq='Y')

forecast = fit.forecast(len(future\_years))

# print forecast values

print(forecast)

f=str(forecast)

print(f)

print("after")

p=0

p1=24

op=""

n=int(n)

for i in range(n):

c=f[p:p1]

op=c

p=p+28

p1=p1+28

flash(op)

print(op)

df=pd.read\_csv('data.csv',parse\_dates=True)

df=df.dropna()

df

df\_m=df.set\_index('date')

df\_m

df=df\_m.drop(['yield'], axis=1)

df

df.corr()

print("\nMissing values : ", df.isnull().any())

import pmdarima as pm

model = pm.auto\_arima(df['production'],seasonal=False,

start\_p=0, start\_q=0, max\_p=3,max\_d=2,max\_q=3, test='adf',error\_action='ignore',suppress\_warnings=True,stepwise=True, trace=True)

train=df[(df.index.get\_level\_values(0) >= '2016-01-01') & (df.index.get\_level\_values(0)

<= '2022-01-01')]

test=df[(df.index.get\_level\_values(0) > '2022-01-01')]

model.fit(train['production'])

forecast=model.predict(n\_periods=n, return\_conf\_int=True)

forecast

flash("VALUES PREDICTED USING MULTIVARIATE ARIMA MODEL")

flash(" ")

f=str(forecast)

p=1

p1=24

n=int(n)

op=""

for i in range(n):

c=f[p:p1]

op=c

p=p+28

p1=p1+28

flash(op)

return render\_template('cost.html')

if \_\_name\_\_=="\_\_main\_\_":

app.run()

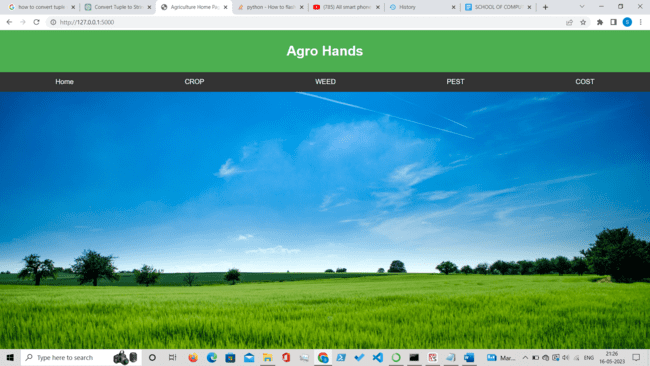
**OUTPUT SCREENSHOTS**

Fig.2 Home Page

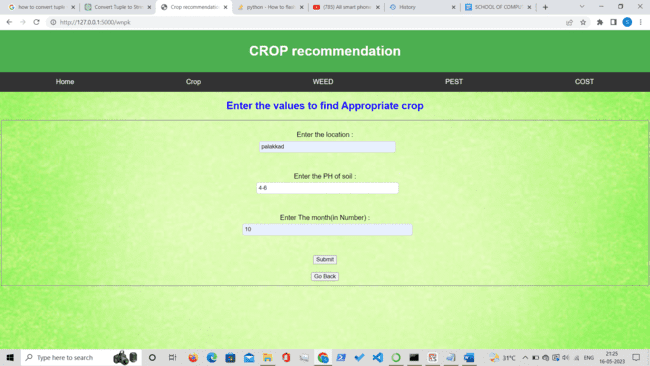
****

Fig.3 Crop recommendation Module(Input)

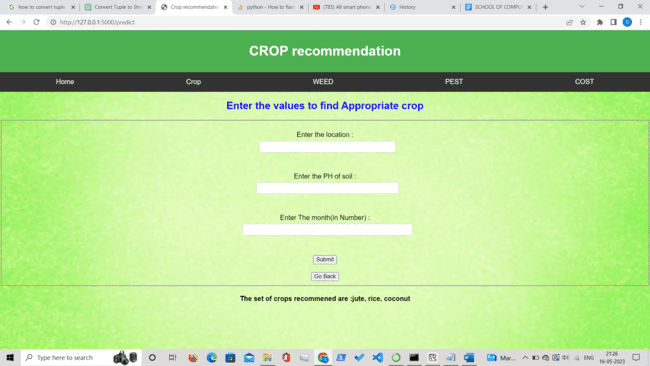
****

Fig.4 Crop recommendation Module(Output)

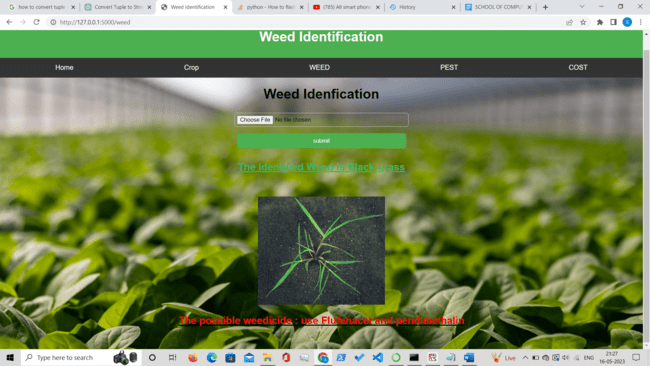
****

Fig.5 Weed Identification Module

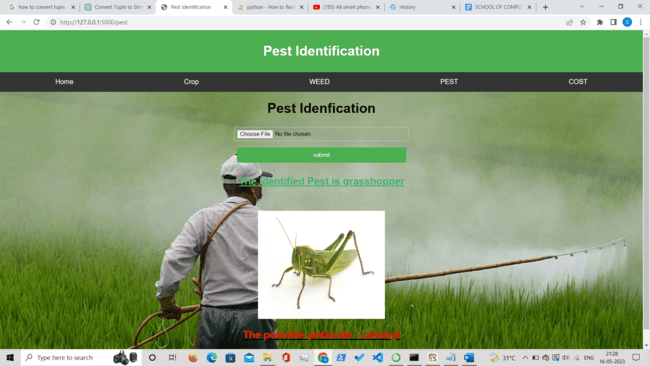


Fig.6 Pest Identification Module

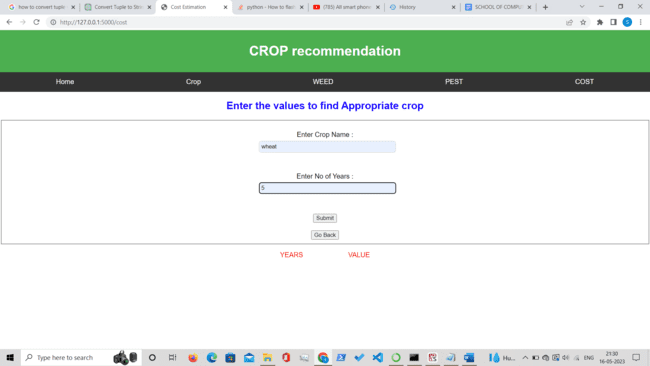


Fig 7. Crop Cost Estimation Module

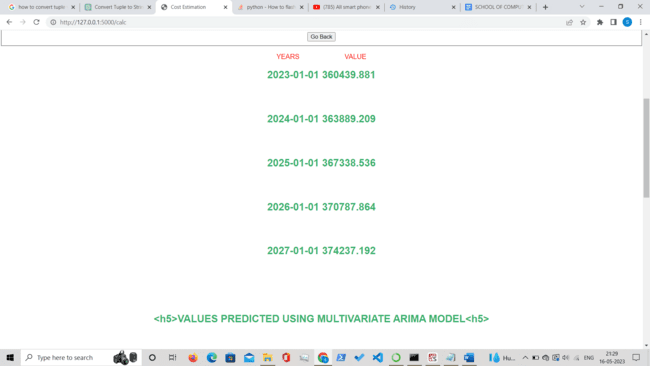


Fig.8 Crop Cost Module output using Exponential Smoothing

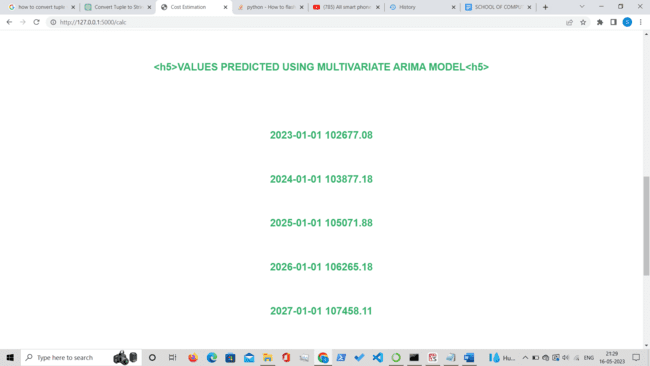


Fig.9 Crop Cost Module output using Multivariate ARIMA Model

**RESULT ANALYSIS:**

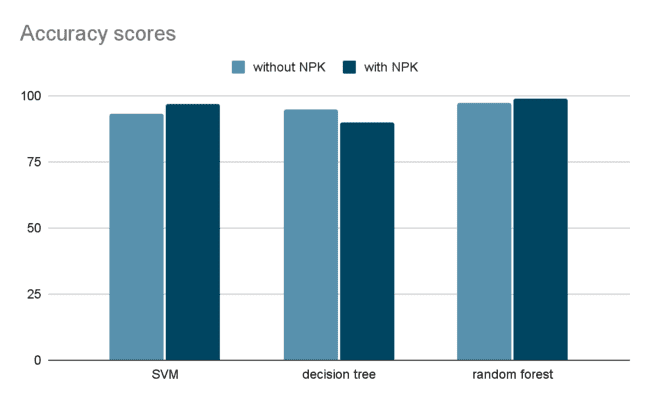
**Crop Recommendation module:** 

Fig.10 Accuracy scores

**Weed and Pest identification module:**

Accuracy of RESNET model for weed identification: 98.23

Accuracy of RESNET model for Pest identification: 98.23

**Crop Cost Estimation:**

**Plot for actual and forecasted values**

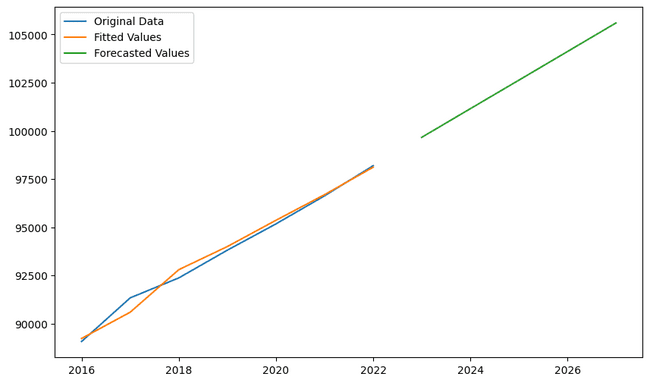
****

Fig. 11 Plot for actual and forecasted values for the exponential smoothing model

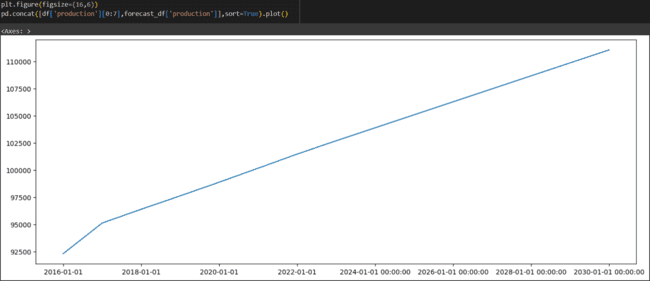
****

Fig.12 Plot for actual and forecasted crop cost values for multivariate ARIMA model

**MAPE(Mean absolute percentage error)**: For the best accurate model the value should be less than 10%

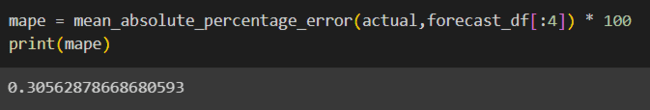
****

Fig.13 Mean Absolute Percentage Error for ARIMA Model

**CHAPTER - 6 CONCLUSIONS**

**Conclusions:**

* The modules in the work will provide a helping hand to farmers to identify the crops that can be grown based on their location.
* Forecasting crop costs will aid in improved decision-making, which minimizes loss.
* Agriculture and development would be sustainable as a result of the system.
* This work will influence many youngsters to practice agriculture.

**CHAPTER - 7 REFERENCES**

**References:**

* Paustian, M., Theuvsen, L. Adoption of precision agriculture technologies by German crop farmers. *Precision Agric* 18, 701–716 (2017). <https://doi.org/10.1007/s11119-016-9482-5>
* A. Sharma, A. Jain, P. Gupta, and V. Chowdary, "Machine Learning Applications for Precision Agriculture: A Comprehensive Review," in *IEEE Access*, vol. 9, pp. 4843-4873, 2021, doi: 10.1109/ACCESS.2020.3048415.
* Andreas Kamilaris, Francesc X. Prenafeta-Boldú, Deep learning in agriculture: survey, Computers and Electronics in Agriculture, Volume 147,2018, Pages 70-90, ISSN 0168-1699,

<https://doi.org/10.1016/j.compag.2018.02.016>.

* Meena, Susheela, I. P. Singh, and Ramji Lal Meena. "Cost of cultivation and returns on different cost concepts basis of onion in Rajasthan." *Economic Affairs* 61.1 (2016): 11-16.
* Priyadharshini, A., et al. "Intelligent crop recommendation system using machine learning." *2021 5th international conference on computing methodologies and Communication (ICCMC)*. IEEE, 2021.

Agro Hand

**Submissiondate:**22-May-202309:58AM(UTC+0530)

**SubmissionID:**2098890748

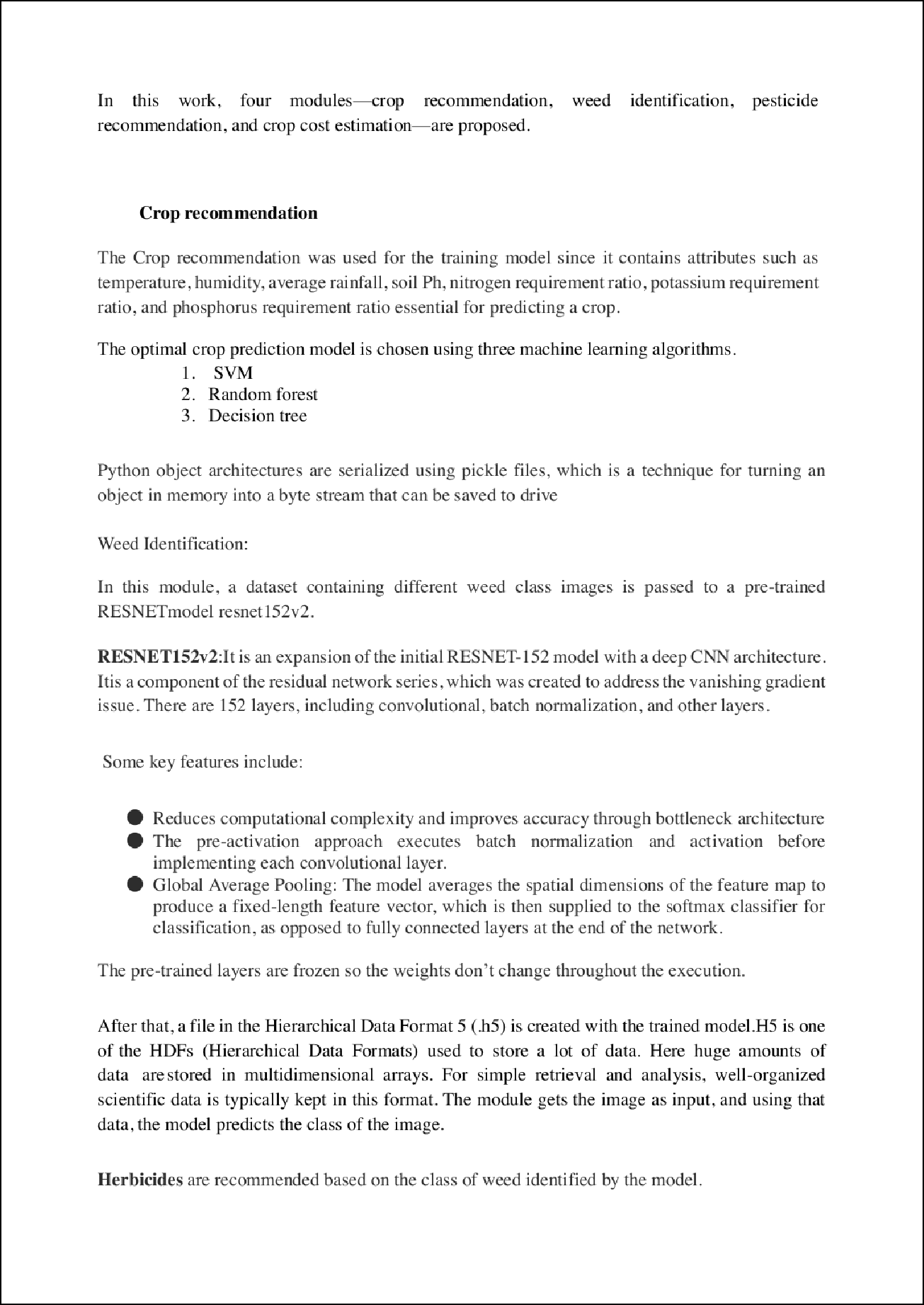
**Filename:**Crop\_identiﬁcation.odt(43.16K)

**Wordcount:**1431

**Charactercount:**8153



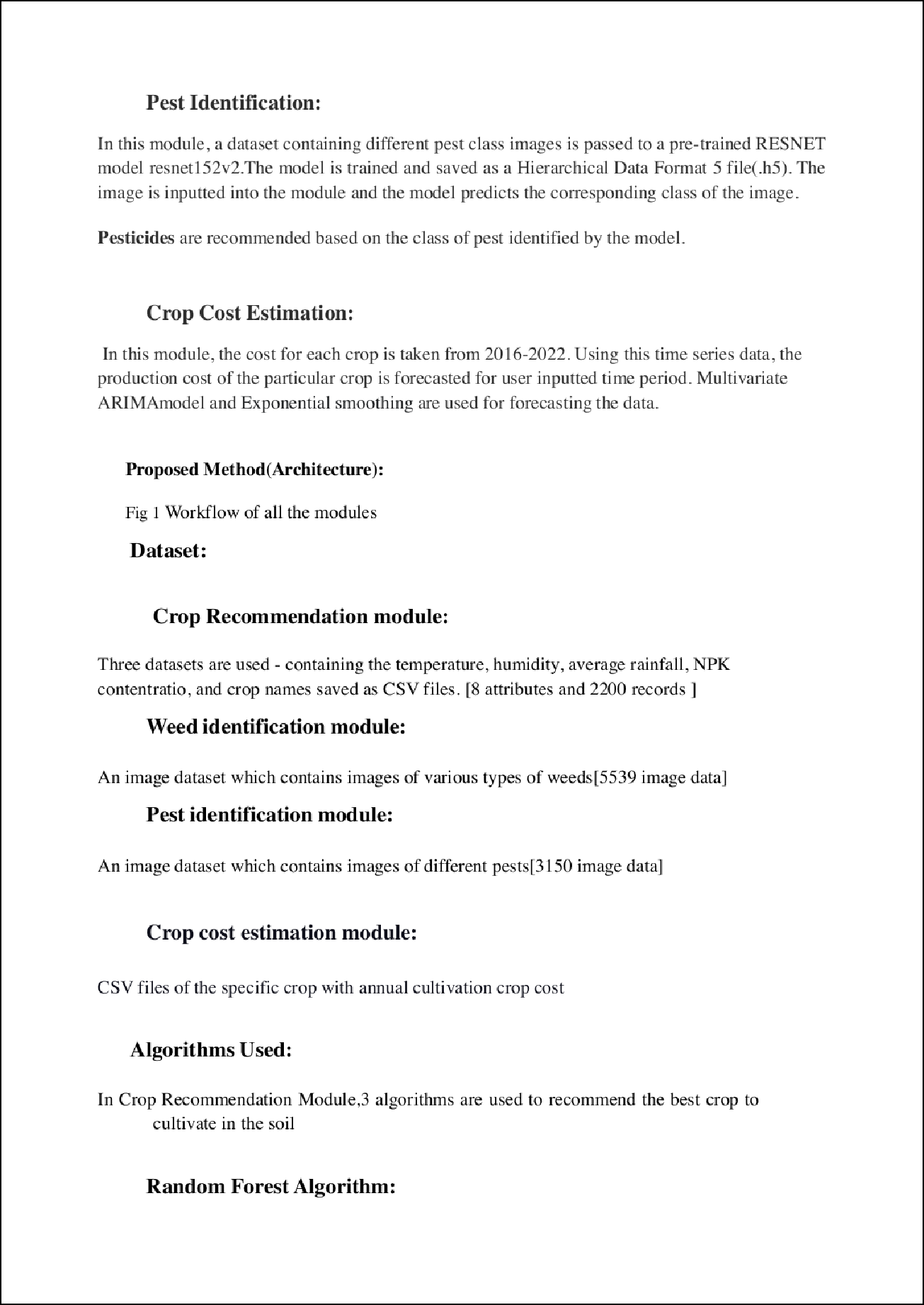
.

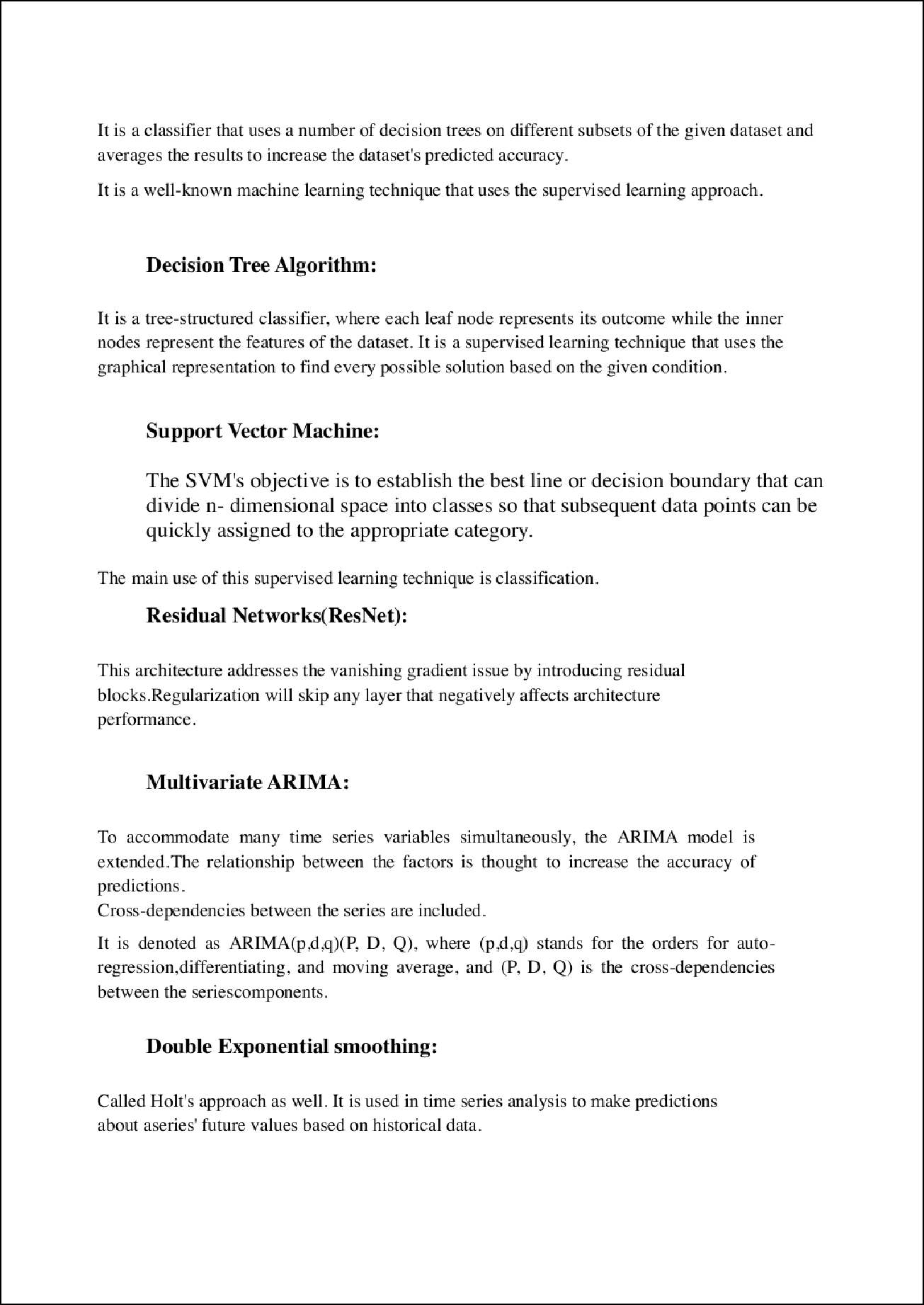


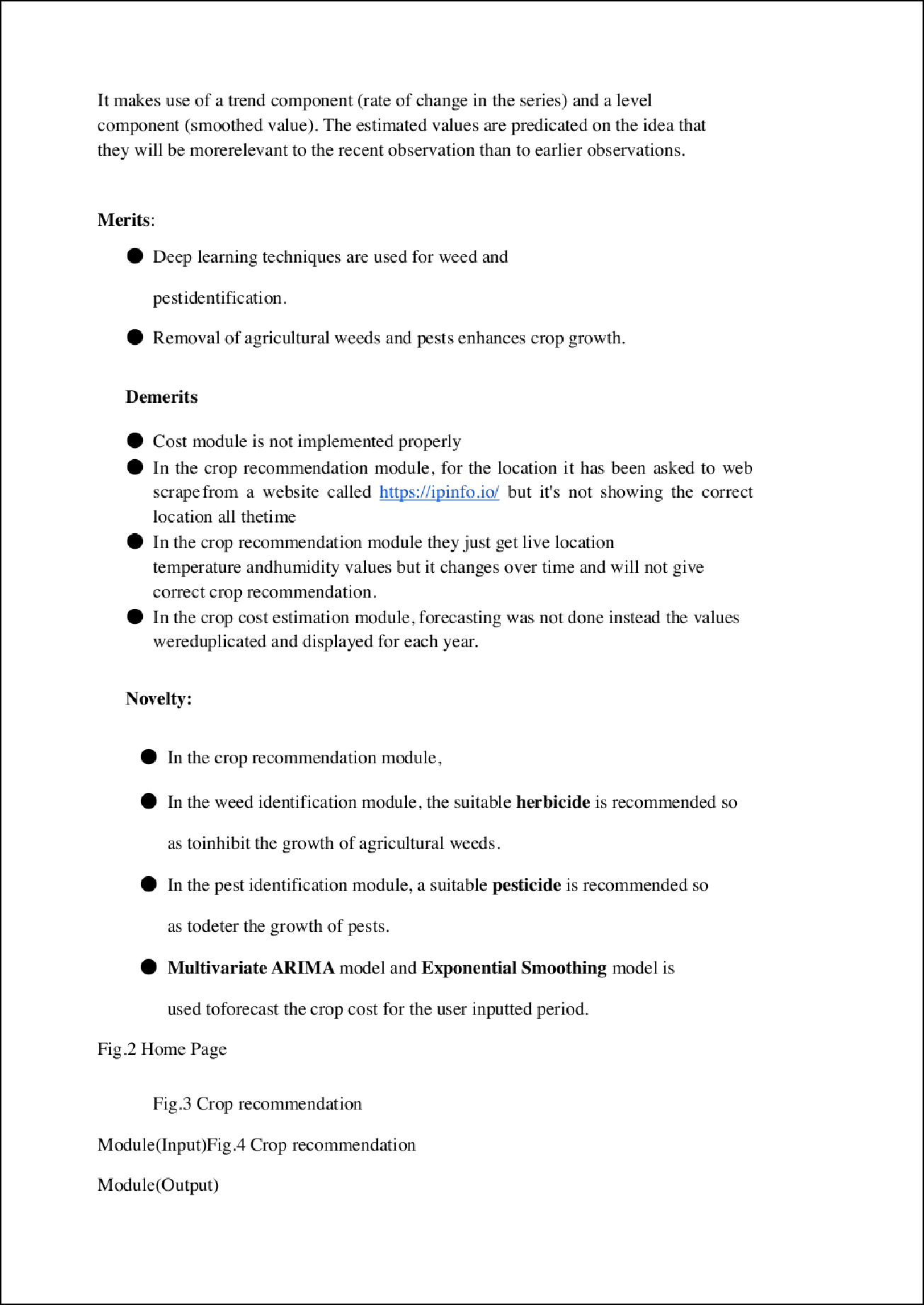
.

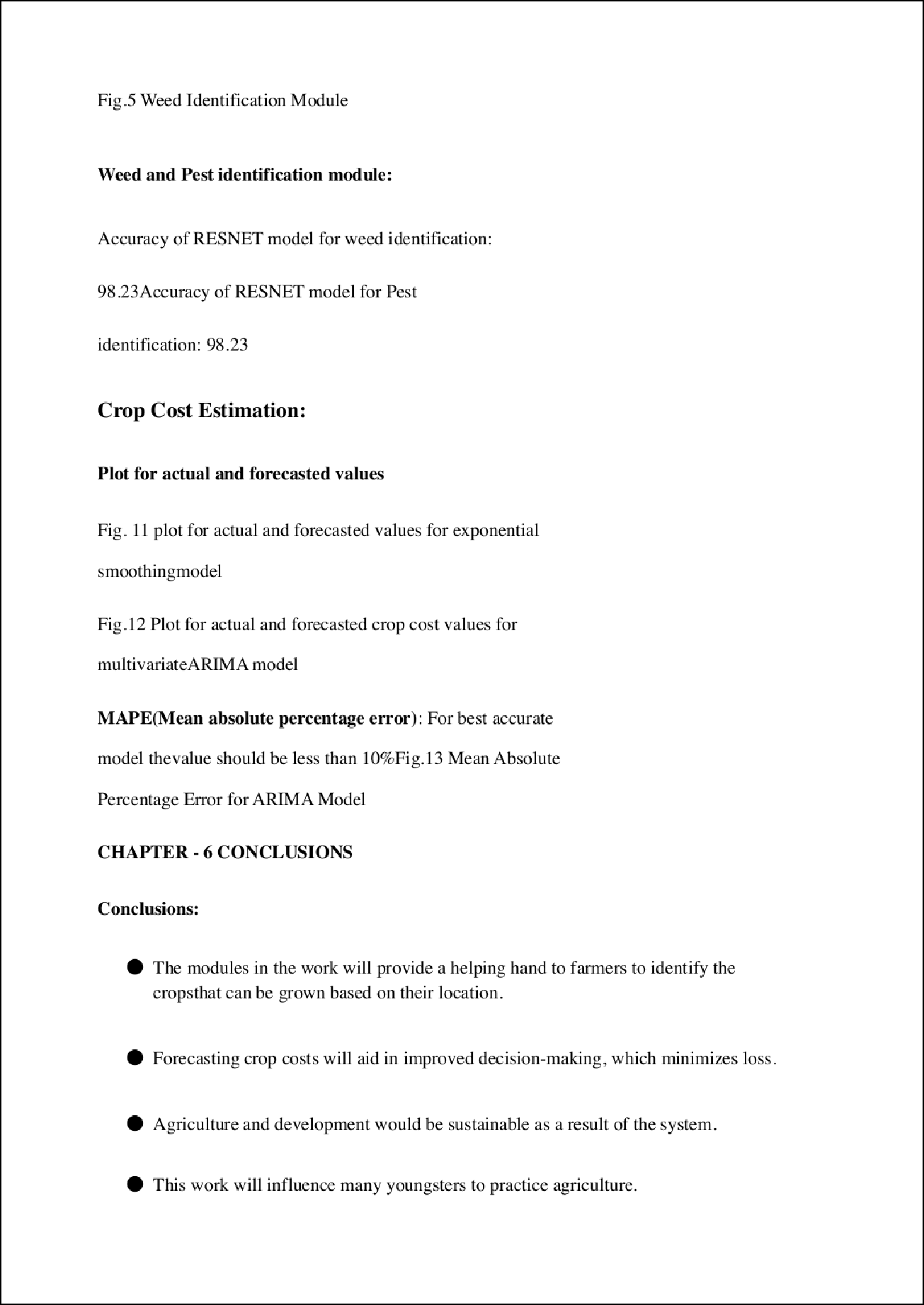
.

.









.

Agro hand

ORIGINALITYREPORT

13

%

SIMILARITYINDEX

6%

INTERNETSOURCES

10%

PUBLICATIONS

5%

STUDENTPAPERS

MATCHALLSOURCES(ONLYSELECTEDSOURCEPRINTED)

6%

SenthilKumarSwamiDurai,MaryDivyaShamili."SmartfarmingusingMachineLearningandDeepLearningtechniques",DecisionAnalyticsJournal, 2022

Publication

Excludequotes OnExcludebibliography On

Excludematches <5words