HARVESTIFY-A CROP AND FERTILIZER RECOMMENDATION PROJECT

Yashwantej Dyavari Shetty

CONTENTS

1. ABSTRACT

Agriculture has always been the central pillar of the Indian economy. The sector has served the food consumption needs of the entire country, while also placing among the highest exporters of agricultural produce within the world. This sector has been facing its share of challenges in recent years, but few are as severe because the domestic and international travel restrictions during Covid-19. As people realize the importance of healthy food – and why words like "organic" really matter, agricultural and farming businesses have a chance to grow.

For the farming and agricultural sectors, online marketing can easily cause more sales, brand awareness and increased market share. And, unlike within the past, this growth is allowing the farmers and business owners to dictate how they need to work, which is one among the most important reasons why websites really are helpful. But with the reverse labour migration leading to scarcity of labour which affected harvesting, farmers need a way to increase productivity. In order to keep the output constant, farmers need to grow the right crops and use the right fertilizers. Many farmers are unaware and grow crops which are unsuitable on a particular land which lead to a huge loss of time and equipment. Therefore, to solve this problem, farmers can take help from online applications to know the most suitable crop to grow on their land.

Harvestify is a website that recommends the optimal crop to grow and fertilizers to use for your crops. In this website, farmers can get Crop recommendations and Fertilizer recommendations. In the crop recommendation module, the user can provide the soil data from their side and the application will predict which crop should the user grow. For the fertilizer suggestion module, the user can input the soil data and the type of crop they are growing, and the application will predict what the soil lacks or has an excess of and will recommend improvements.

2. INTRODUCTION

Agriculture is the primary source of livelihood for about 58% of India's population. India has second highest land of more than 1.6 million square-kilometres under cultivation. India possesses a power potential to be a superpower in the field of agriculture. Agriculture remains a central pillar of the Indian economy. The sector serves the food consumption needs of the whole country, while also placing among the top exporters of agricultural produce in the world. The sector has been facing its share of challenges in recent years, but few have been as severe as the domestic and international travel restrictions during Covid-19 pandemic. The pandemic disrupted demand and supply of food impacting the global supply chain.

Many farmers are unaware and grow crops which are unsuitable on a particular land which lead to a huge loss of time and equipment. There is no universal system to assist optimum crop to cultivated by farmers. Since India has been following agriculture from ages, we have a rich collection of agriculture data which can used for new technologies like precision agriculture. Precision agriculture is modern farming technique that uses data of soil characteristics, soil types, crop yield data, whether conditions and suggests farmers with the optimal crop to grow in their farmland for maximum yield and profit. This technique can reduce crop failures and will help farmers to take informed decision about their farming strategy. The main motive of this project is to recommend optimum crops to be cultivated by farmers based on several parameters and help them make an informed decision before cultivation. Farmers can also know what type of fertilizer to use on the crop based on their soil ph. value.

2.1 Motivation

As technology is advancing at lightning speed, with that people are getting more addicted to mobile phones, tablets & other smart devices. Agriculture remains a central pillar of the Indian economy. The sector serves the food consumption needs of the whole country, while also placing among the top exporters of agricultural produce in the world. The sector has been facing its share of challenges in recent years, but few have been as severe as the domestic and international travel restrictions during Covid-19. The pandemic disrupted demand and supply of food impacting the global supply chain.

2.2 Problem Statement

The main objective of this project is to create a mobile application that can provide a platform for various artists to display and sell their artwork by finding potential customers and for them to stay informed about exhibitions. The reverse labor migration led to scarcity of labor which affected harvesting. Many farmers are unaware and grow crops which are unsuitable on a particular land which lead to a huge loss of time and equipment.

2.3 Requirements

Specification

2.3.1 Hardware Requirements

Processor :Intel core (i5 or higher)
RAM : 4 GB
ROM : 8 GB

2.3.2 Software Requirements

Operating System : MacOS/Windows
Programming Language : Python Integrated Development Environment : Google Colaboratory

2.4 Technology

1.5.1 Google Colab Google is quite aggressive in AI research. Over many years, Google developed an AI framework called Tensor Flow and a development tool called Colaboratory. Today Tensor Flow is open-sourced and since 2017, Google made Colaboratory free for public use. Colaboratory is now known as Google Colab or simply Colab. Colaboratory or Colab for

short, is a Google Research product, which allows developers to write and execute Python code through their browser. Google Colab is an excellent tool for deep learning tasks. It is a hosted Jupyter notebook that requires no setup and has an excellent free version, which gives free access to Google computing resources such as GPUs and TPUs. Colaboratory combines code, output and descriptive text into one document (interactive notebook). By using Google Colab, you can: • Build your analytics products quickly in a standardized environment. • Facilitates popular Deep Learning libraries on the go such as PyTorch and TensorFlow. • Share code and results within your Google Drive. • Save copies and create playground modes for knowledge sharin 5 1.5.2 Python • Python is an interpreted high-level programming language for generalpurpose programming. Created by Guido van Rossum, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. • It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management • It supports multiple programming paradigms, including object- oriented, imperative, functional and procedural, and has a large and comprehensive standard library. • Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation

3. METHODOLOGY

MODULES USED

3. 1**NumPy:** NumPy is a library for the Python programming language, adding support for large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers

3.2 OpenCV: OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Starting with 2011, OpenCV features GPU acceleration for real-time operations. OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in

realtime operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human.

3.3 Flask: Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

4. MODELS

The dataset of this project is taken from Kaggle. The data is made by augmenting and combining various publicly available datasets of India like weather, soil, etc. This data is relatively simple with very few but useful features unlike the complicated features affecting the yield of the crop. The data have Nitrogen, Phosphorous, Potassium, and pH values of the soil. Also, it also contains the humidity, temperature, and rainfall required for a particular crop. The crops considered in our model include rice, maize, chickpea, kidney beans, pigeon peas, moth beans, mung beans, black gram, lentil, pomegranate, banana, mango, grapes, watermelon, musk melon, orange, papaya, coconut, cotton, jute, coffee. The dataset has a hundred records of Nitrogen, Phosphorous, Potassium, and pH values of the soil corresponding to each crop mentioned above.

The above-stated parameters of soil play a major role in the crop's ability to grow. Nitrogen is used by plants for lots of leaf growth and good green colour. Phosphorous is used by plants to help form new roots, make seeds, flowers, and fruits. It’s also used by plants to fight diseases. Potassium helps plants make strong stems and keep growing fast. It’s also used by plants to fight diseases. The soil pH can influence plant growth by its effect on the activity of beneficial microorganisms Bacteria that decompose soil organic matter are hindered in strong acid soils. Temperature influences most plant processes, including photosynthesis, transpiration, respiration, germination, and flowering. Hence for the following reasons, the above-stated parameters are considered for choosing a crop.

The dataset is trained using the following algorithms:

8. Decision Tree
9. Gaussian Naïve Bayes
10. Support Vector Machine (SVM)
11. Logistic Regression
12. Random Forest
13. XGBoost

Decision Tree

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a treestructured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome**.** In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node**.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

Gaussian Naïve Bayes

Gaussian Naïve Bayes is a variant of Naïve Bayes that follows Gaussian normal distribution and supports continuous data.**Naive Bayes** are a group of supervised machine learning classification algorithms based on the **Bayes theorem**. It is a simple classification technique, but has high functionality. They find use when the dimensionality of the inputs is high.
Complex classification problems can also be implemented by using Naive Bayes Classifier.

Support Vector Machine

Support Vector Machine (SVM) is a supervised Machine Learning Algorithm which can be used for both classification or regression challenges. It is mostly used in classification problems. In the SVM algorithm, we can plot each data item as a point in n-dimensional space (where n is number of features, we have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper plane that differentiates the two classes.

Logistic Regression

Logistic regression is a supervised learning classification algorithm used to predict theprobability of a target variable. The nature of target or dependent variable is dichotomous,which means there would be only two possible classes. In simple words, the dependentvariable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (standsfor failure/no). Mathematically, a logistic regression model predicts P(Y=1) as a function of X.

Random Forest

Random tree is similar to that of a decision tree. But it differs from random tree in a way that for each split only a random subset of attributes is available. Random trees can be built for both nominal and numerical data. The Random Tree is similar to C4.5 or CART but it varies in the fact that before it is applied for training it selects only a random subset of attributes. At each node it considers K randomly chosen attributes. The subset ratio parameter specifies the size of the subset.

XGBoost

XGBoost is an implementation of gradient boosted decision trees designed for speed performance. XGBoost is an ensemble learning method. Ensemble learning offers a systematic solution to combine the predictive power of multiple learners. The resultant is a single model which gives the aggregated output from several models. The models that form the ensemble, also known as base learners, could be either from the same learning algorithm or different learning algorithm. Bagging and boosting are two widely used ensemble learners. Though these two techniques can be used with several statistical models, the most predominant usage has been with decision trees.

Flow chart of the process consists of dataset split into training data and testing data. Training data gets feed into machine learning algorithms. A model is evaluated using algorithms and Testing data. The model is used to predict crop to be yield based on given parameters.

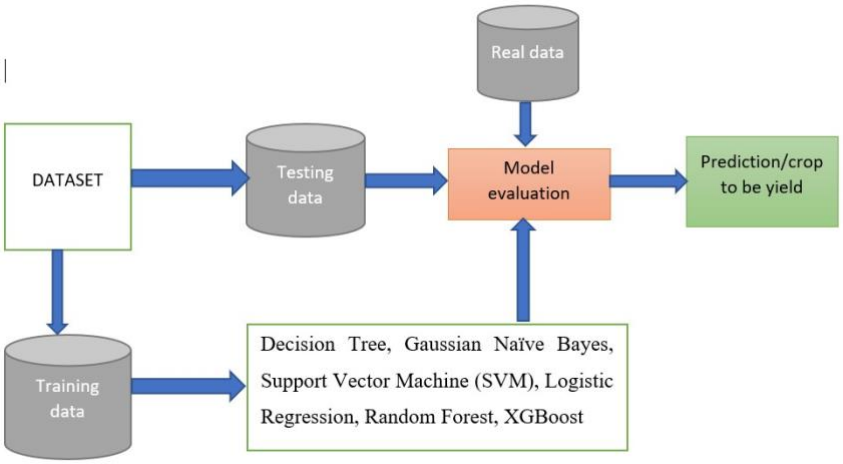

Fig 4.1: Flow chart of the process

The dataset gets trained using the algorithms mentioned above. Every trained model is then saved into files, respectively. Accuracy of the models is then compared.

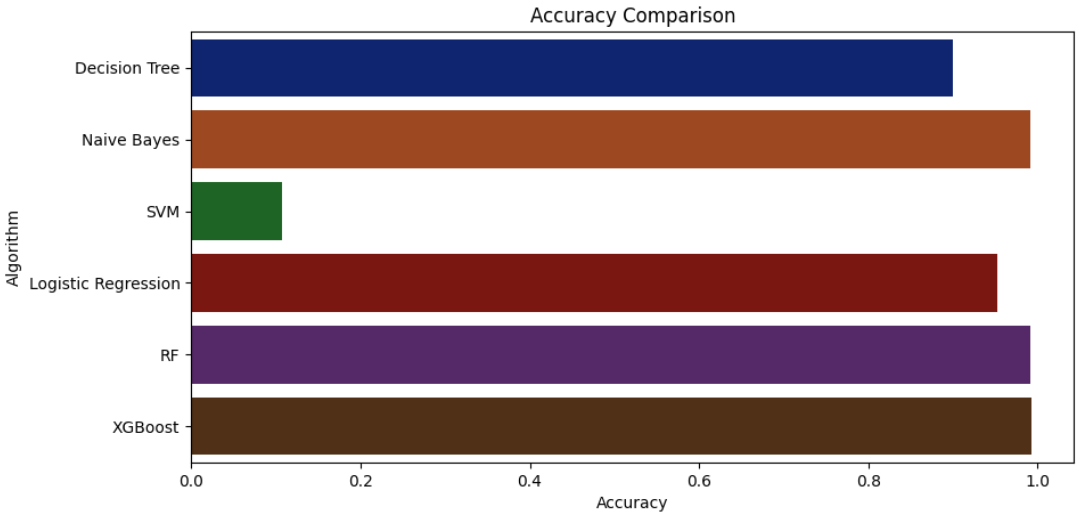

Fig 4.2: Accuracy comparison

The prediction accuracies of the models are Decision Tree 90%, Naive Bayes 99.090%, SVM 10.6%, Logistic Regression 95.227%, Random forest 99.7%, XGBoost 99.318%. Since prediction accuracy of Random Forest is high, the random forest model is used to predict which crop to be cultivated.

Flask is a web application framework written in Python. Flask supports the extensions to add such functionality to the application. This project is deployed using flask.

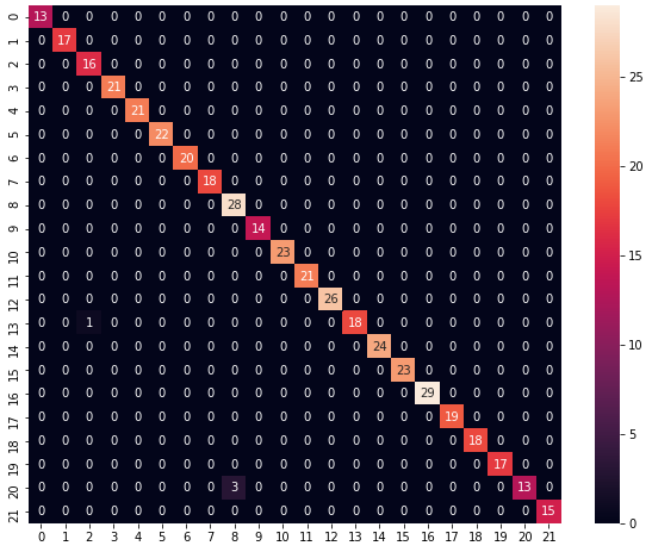

Fig 4.3: Confusion Matrix

5. CODE

```
# Importing essential libraries and modules


from flask import Flask, render_template, request, Markup

import numpy as np import pandas as pd from

utils.disease import disease_dic from utils.fertilizer import

fertilizer_dic import requests import config import pickle

import io import torch from torchvision import transforms

from PIL import Image from utils.model import ResNet9

#
==========================================================================
==============================


# -------------------------LOADING THE TRAINED MODELS -----------------------------------
----------


# Loading plant disease classification model


disease_classes = ['Apple___Apple_scab',

                   'Apple___Black_rot',

                   'Apple___Cedar_apple_rust',

                   'Apple___healthy',

                   'Blueberry___healthy',

                   'Cherry_(including_sour)___Powdery_mildew',

                   'Cherry_(including_sour)___healthy',

                   'Corn_(maize)___Cercospora_leaf_spotGray_leaf_spot',
```

'Corn_(maize)___Common_rust_',

'Corn_(maize)___Northern_Leaf_Blight',

'Corn_(maize)___healthy',

'Grape___Black_rot',

'Grape___Esca_(Black_Measles)',

'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',

'Grape___healthy',

'Orange___Haunglongbing_(Citrus_greening)',

'Peach___Bacterial_spot',

'Peach___healthy',

'Pepper,_bell___Bacterial_spot',

'Pepper,_bell___healthy',

'Potato___Early_blight',

'Potato___Late_blight',

'Potato___healthy',

'Raspberry___healthy',

'Soybean___healthy',

'Squash___Powdery_mildew',

'Strawberry___Leaf_scorch',

'Strawberry___healthy',

'Tomato___Bacterial_spot',

'Tomato___Early_blight',

'Tomato___Late_blight',

'Tomato___Leaf_Mold',

'Tomato___Septoria_leaf_spot',

```
                'Tomato___Spider_mites Two-spotted_spider_mite',
                'Tomato___Target_Spot',
                'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
                'Tomato___Tomato_mosaic_virus',
                'Tomato___healthy']

disease_model_path = 'models/plant_disease_model.pth'
disease_model = ResNet9(3, len(disease_classes))
disease_model.load_state_dict(torch.load( disease_model_path,
map_location=torch.device('cpu'))) disease_model.eval()



# Loading crop recommendation model

crop_recommendation_model_path = 'models/RandomForest.pkl'
crop_recommendation_model = pickle.load( open(crop_recommendation_model_path, 'rb'))



#
==========================================================================
=======================

# Custom functions for calculations
```

```
def weather_fetch(city_name):
    """
    Fetch and returns the temperature and humidity of a city
:params: city_name
:return: temperature, humidity
    """
api_key = config.weather_api_key base_url =
"http://api.openweathermap.org/data/2.5/weather?"

complete_url = base_url + "appid=" + api_key + "&q=" + city_name
response = requests.get(complete_url) x = response.json()

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

        temperature = round((y["temp"] - 273.15), 2)
humidity = y["humidity"] return
temperature, humidity
    else:
        return None


def predict_image(img, model=disease_model):
    """
    Transforms image to tensor and predicts disease label
```

```
:params: image
:return: prediction (string)
    """
    transform = transforms.Compose([
transforms.Resize(256), transforms.ToTensor(),
    ])
    image = Image.open(io.BytesIO(img))
img_t = transform(image) img_u =
torch.unsqueeze(img_t, 0)

    # Get predictions from model yb
= model(img_u)
  # Pick index with highest probability
    _, preds = torch.max(yb, dim=1)
    prediction = disease_classes[preds[0].item()]
    #   Retrieve   the   class   label
return prediction

#
================================================================================
==============================
# ----------------------------------- FLASK APP -------------------------------------------------


app = Flask(__name__)
```

```
# render home page


@ app.route('/') def
home():
    title = 'Harvestify - Home'              return
render_template('index.html', title=title)


# render crop recommendation form page


@ app.route('/crop-recommend') def
crop_recommend():
    title = 'Harvestify - Crop Recommendation'
    return render_template('crop.html', title=title)


# render fertilizer recommendation form page


@    app.route('/fertilizer')    def
fertilizer_recommendation():
    title = 'Harvestify - Fertilizer Suggestion'


    return render_template('fertilizer.html', title=title)
```

```
# render disease prediction input page




#
==========
==========


# RENDER PREDICTION PAGES


# render crop recommendation result page



@ app.route('/crop-predict', methods=['POST'])
def crop_prediction():
    title = 'Harvestify - Crop Recommendation'


    if request.method == 'POST':
        N = int(request.form['nitrogen'])
        P = int(request.form['phosphorous'])
K = int(request.form['pottasium']) ph =
float(request.form['ph'])         rainfall =
float(request.form['rainfall'])
```

```
        #   state   =   request.form.get("stt")
city = request.form.get("city")

        if weather_fetch(city) != None:
            temperature, humidity = weather_fetch(city)             data =
np.array([[N, P, K, temperature, humidity, ph, rainfall]])
my_prediction = crop_recommendation_model.predict(data)
final_prediction = my_prediction[0]

            return render_template('crop-result.html', prediction=final_prediction, title=title)

else:

            return render_template('try_again.html', title=title)

# render fertilizer recommendation result page



@ app.route('/fertilizer-predict', methods=['POST']) def
fert_recommend():
    title = 'Harvestify - Fertilizer Suggestion'

crop_name   =   str(request.form['cropname'])
N = int(request.form['nitrogen'])
    P = int(request.form['phosphorous'])
```

```
    K = int(request.form['pottasium'])
    # ph = float(request.form['ph'])

    df = pd.read_csv('Data/fertilizer.csv')

    nr = df[df['Crop'] == crop_name]['N'].iloc[0]
pr = df[df['Crop'] == crop_name]['P'].iloc[0] kr =
df[df['Crop'] == crop_name]['K'].iloc[0]

    n = nr - N     p = pr - P     k = kr - K     temp =
{abs(n): "N", abs(p): "P", abs(k): "K"}
max_value = temp[max(temp.keys())]     if
max_value == "N":
        if n < 0:
            key     =     'NHigh'
else:
            key = "Nlow" elifmax_value
== "P":
        if p < 0:
            key = 'PHigh'
else:             key =
"Plow"     else:
if k < 0:
            key     =     'KHigh'
else:
```

```
            key = "Klow"

    response = Markup(str(fertilizer_dic[key]))

    return render_template('fertilizer-result.html', recommendation=response, title=title)

# render disease prediction result page


@app.route('/disease-predict', methods=['GET', 'POST']) def
disease_prediction():
    title = 'Harvestify - Disease Detection'

    if request.method == 'POST':
        if 'file' not in request.files:
return redirect(request.url)
file = request.files.get('file')     if
not file:
            return render_template('disease.html', title=title)
        try:
img = file.read()

            prediction = predict_image(img)
```

```
        prediction = Markup(str(disease_dic[prediction]))          return
render_template('disease-result.html', prediction=prediction, title=title)        except:
pass     return render_template('disease.html', title=title)


#
================================================================================
================================

if __name__ == '__main__':
app.run(debug=False)
```

6. RESULTS

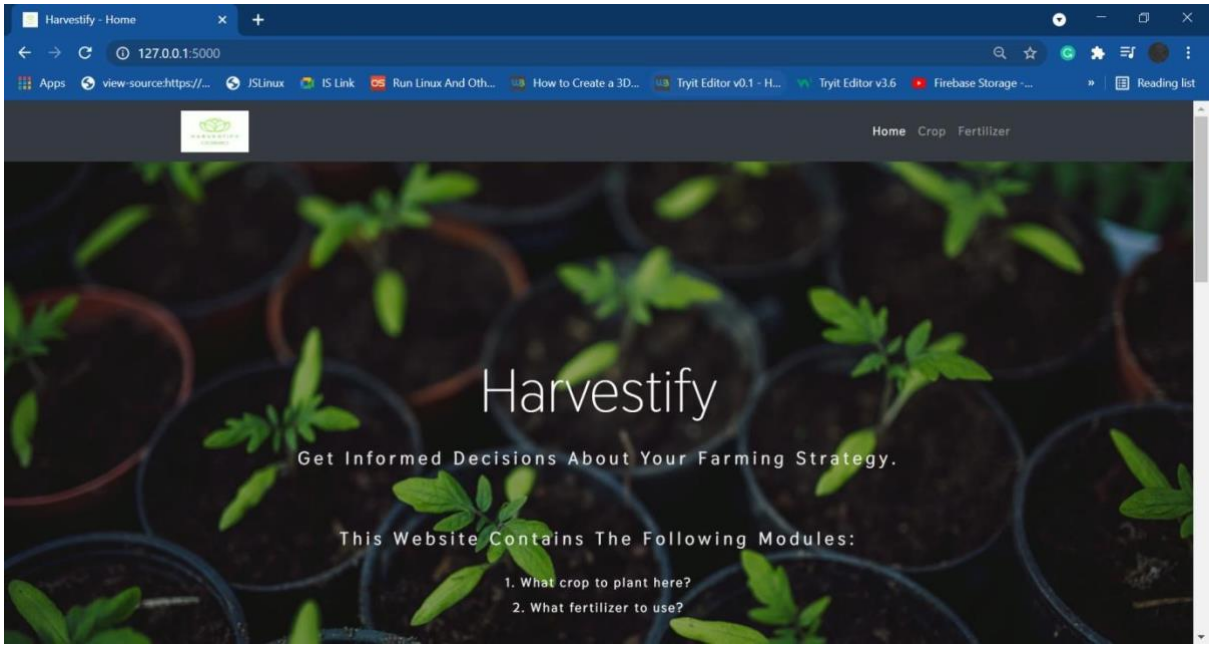

Fig 6.1: Home Page

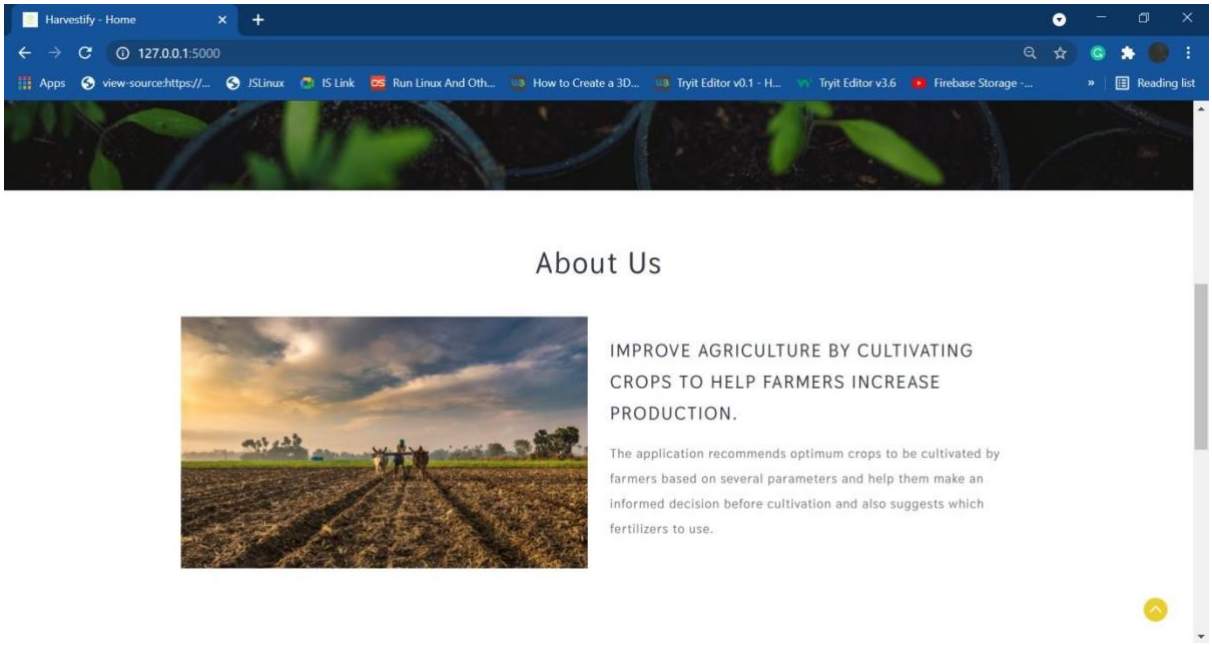

Fig 6.2: Home Page - Intro

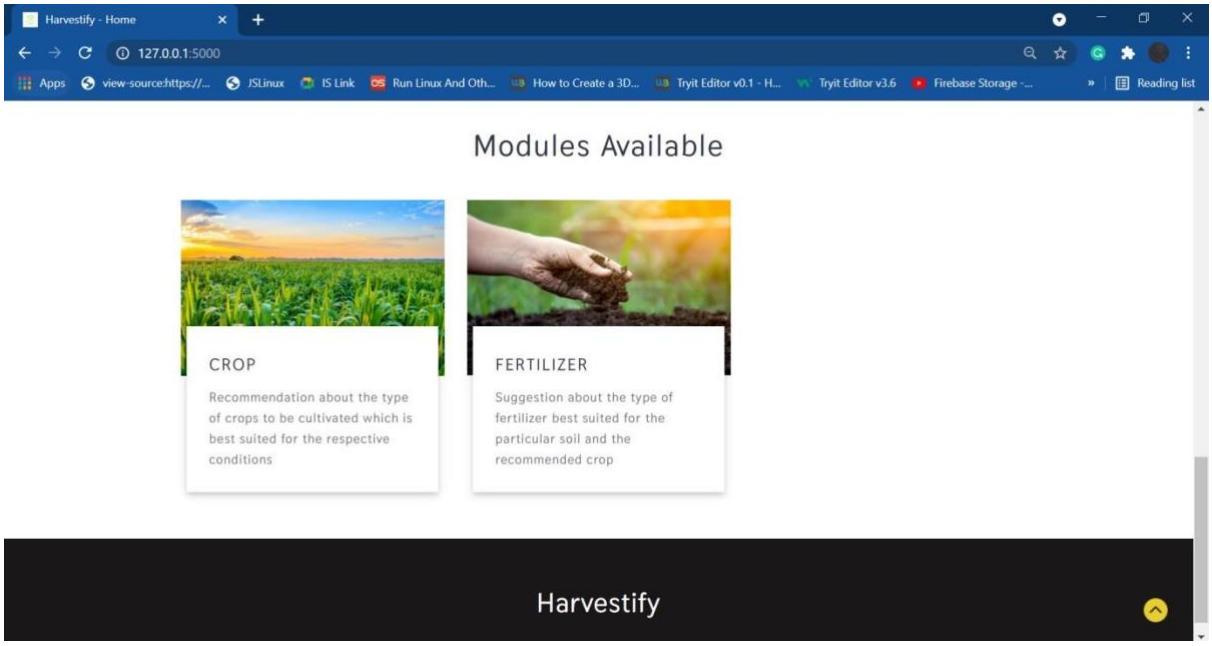

Fig 6.3: Home Page - Modules

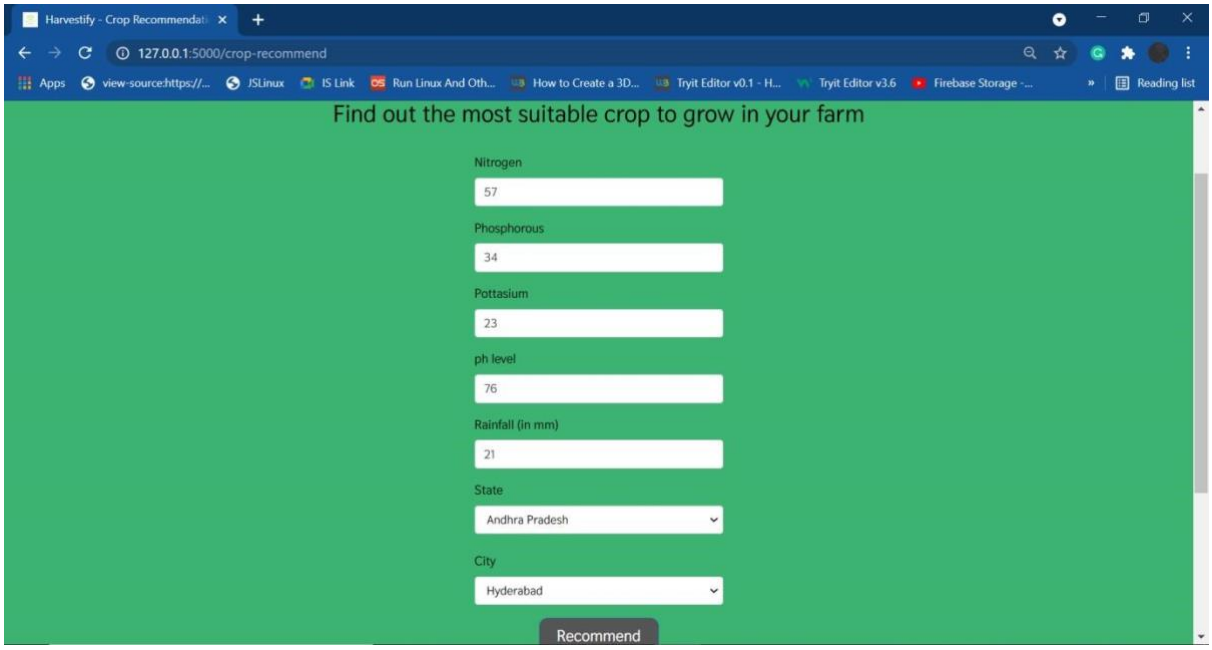

Fig 6.4: Crop Recommendation Page

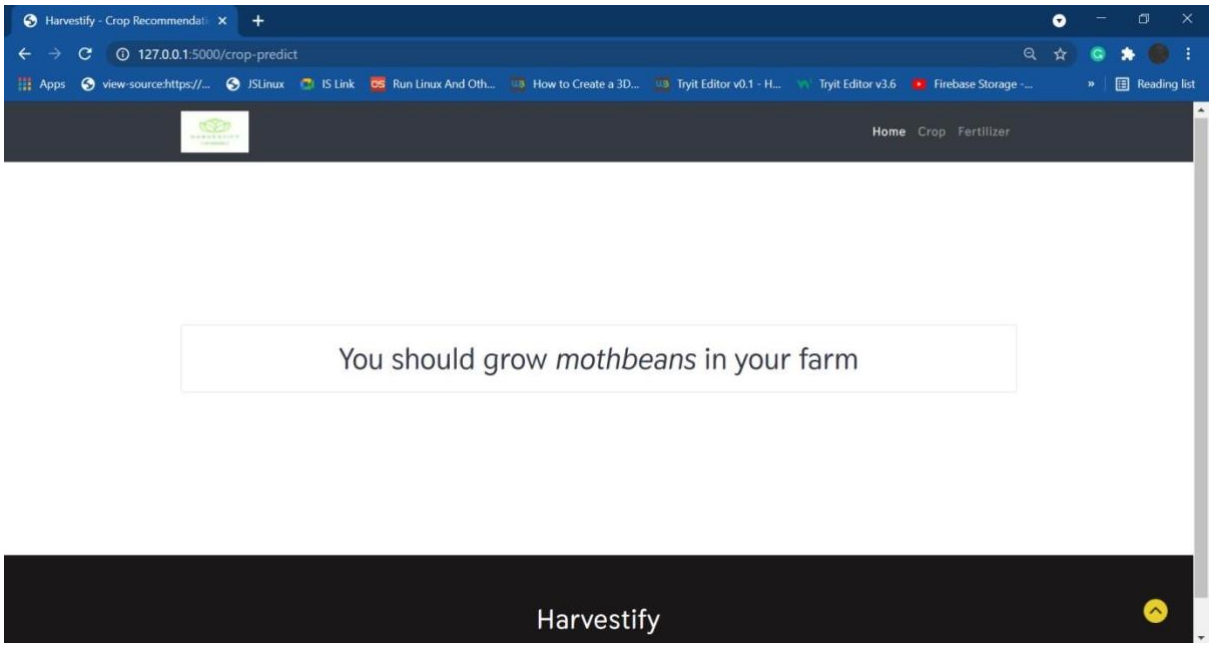

Fig 6.5: Crop Result Page

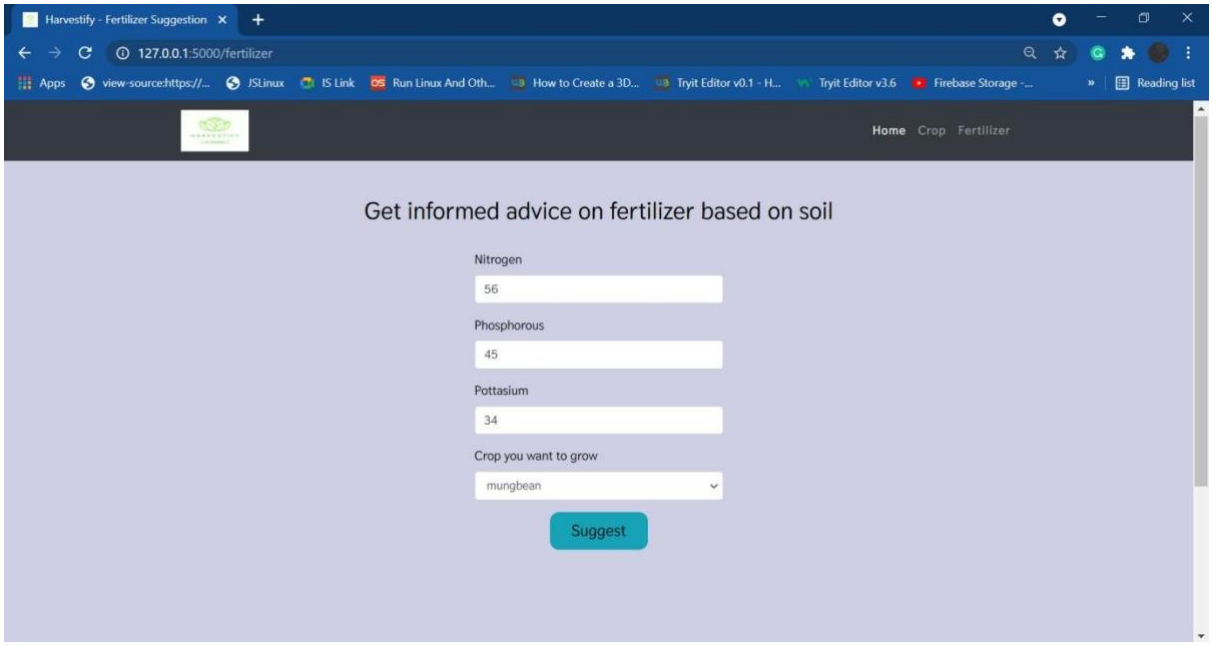

Fig 6.6: Fertilizer Suggestion Page

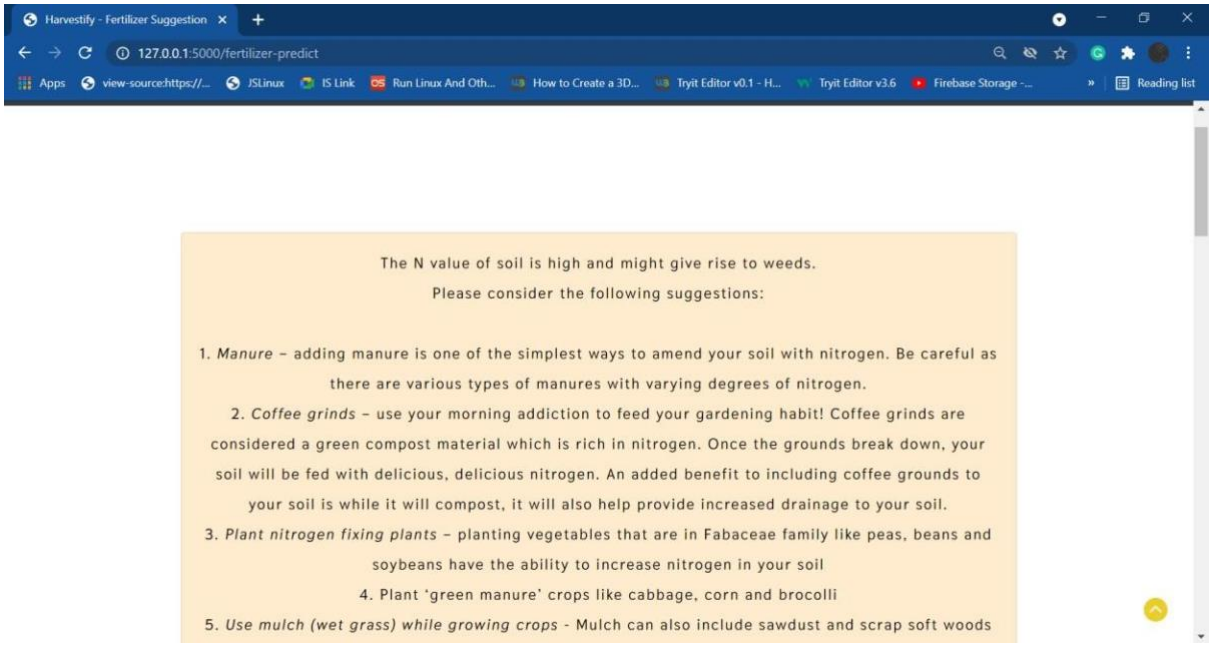

Fig 6.7: Fertilizer Result Page

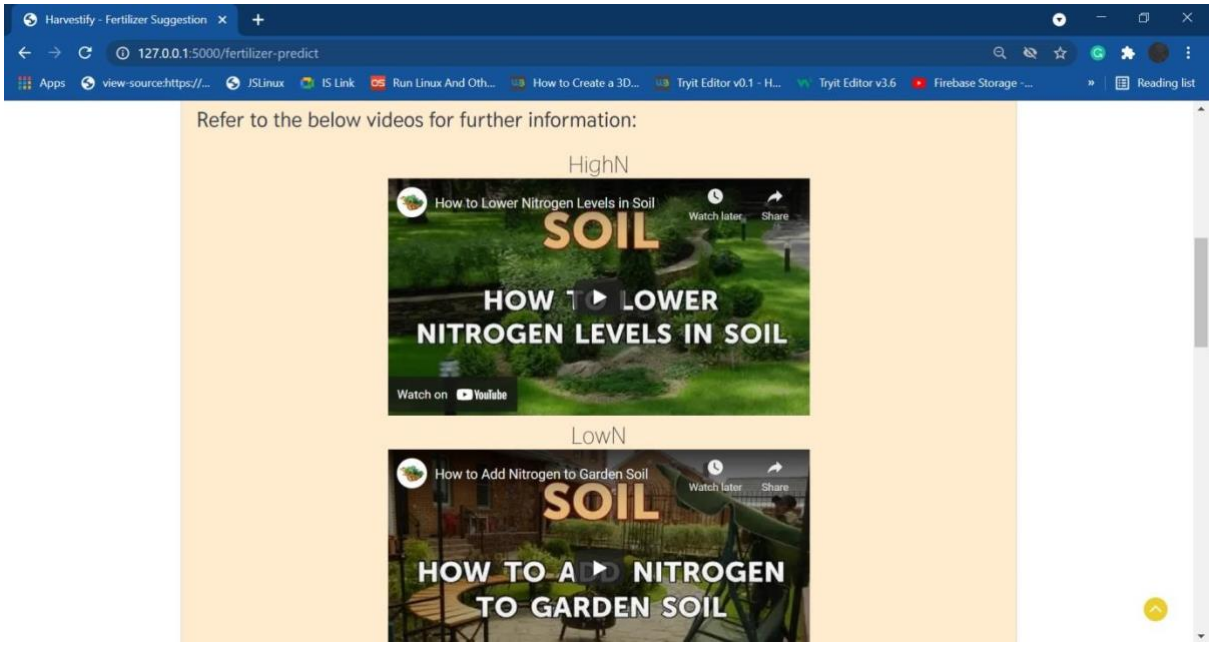

Fig 6.8: Fertilizer Result Page - Videos

7.
CONCLUSIONS

After extensive testing and the final validation, it was concluded that the project is successful in carrying out all the functionalities mentioned throughout this report. This application will therefore provide a platform for farmers to get informed strategies and is developed in such a way that they crop recommendation w.r.t real-time weather data.

8.

FUTURE WORK

A better user interface can be made. More features can be added. Multi-language support can also be added as not all the farmers know English. Only states and cities of India have been taken in the present version of the module and therefore the cities of other countries can be added in the upcoming versions. In the future, this website can also be made into a mobile app and be uploaded onto the Google Play Store where everyone can download and install this app for their use. These limitations are implemented in further versions, along with any further areas of growth. New features and modules can be added into the system in the future.

REFERENCES

9.

[1] https://docplayer.net/102704394-Survey-of-crop-recommendation-systems.html

[2] https://www.researchgate.net/publication/317696446_Crop_recommendation_system_for_precision_agriculture

[3] http://sersc.org/journals/index.php/IJAST/article/download/7794/4513/

[4] http://www.sersc.org/journals/index.php/IJFGCN/article/download/28584/15900/

[5] https://www.ijrte.org/wp-content/uploads/papers/v7i5c/E10300275C19.pdf

[6] http://www.ijcst.com/vol21/veenadhari.pdf

[7] https://www.tutorialspoint.com/flask/index.htm

[8] https://www.javatpoint.com/first-flask-application

[9] https://sci-hub.se/

[10] https://en.wikipedia.org/wiki/Ensemble_learning