**CROP DETECTION WEBSITE**

**A PROJECT REPORT**

**Submitted By**

**Abhinav Choudhary - University Roll No: 2100290140002**

**Akshay Chaudhary - University Roll No: 2100290140010**

**Akshit Sherawat - University Roll No: 2100290140019**

**MASTER OF COMPUTER APPLICATION**

**Session: 2022-23(4th Semester)**

**Under the supervision of**

**Dr. Vidushi**

**Assistant Professor**



**Submitted to**

**Department Of Computer Applications**

**KIET Group of Institutions, Ghaziabad**

**Uttar Pradesh-201206**

**( June 2023)**

**Declaration**

I hereby declare that the work presented in report entitled “Crop Detection Website” was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University of Institute. I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, that are not my original contribution. I have used quotation marks to identify verbatim sentences and give credit to the original authors/sources. I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name: Abhinav Choudhary Name: Akshay Choudhary

Roll No.: 2100290140002 Roll No.: 2100290140010

(Candidate Signature) (Candidate Signature)

Name: Akshit Sherawat

Roll No.: 2100290140019

(Candidate Signature)

**CERTIFICATE**

Certified that **Abhinav Choudhary (210290140002)**, **Akshay Chaudhary (2100290140010), Akshit Sherawat (2100290140019)** have carried out the project work **“Crop Detection Website”** for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the students themselves and the contents of the project report do not form the basis for the award of any other degree to the candidates or to anybody else from this or any other University/Institution.

**Date:**

**Abhinav Choudhary (2100290140002)**

**Akshay Chaudhary (2100290140010)**

**Akshit Sherawat (2100290140019)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

**Dr. Vidushi**

**Assistant Professor**

**Department of Computer Applications**

**KIET Group of Institutions, Ghaziabad**

**Signature of Internal Examiner Signature of External Examiner**

**Dr. Arun Kumar Tripathi**

**Head, Department of Computer Applications**

**KIET Group of Institutions, Ghaziabad**

**ABSTRACT**

Earlier, crop cultivation was undertaken on the basis of farmers’ hands-on expertise. However, climate change has begun to affect crop yields badly. Consequently, farmers are unable to choose the right crop/s based on soil and environmental factors, and the process of manually predicting the choice of the right crop/s of land has, more often than not, resulted in failure. Accurate crop prediction results in increased crop production. This is where machine learning playing a crucial role in the area of crop prediction.

Automating agricultural aspects is a mechanical process with or without human intervention in agriculture. Due to less space of domestic lands, it has become an important area of choosing the most suitable crops based on prevailing factors in the selected area.

Crop prediction depends on the soil, geographic and climatic attributes. Selecting appropriate attributes for the right crop/s is an intrinsic part of the prediction undertaken by feature selection techniques. In this work, a comparative study of various wrapper feature selection methods are carried out for crop prediction using classification techniques that suggest the suitable crop/s for land. The experimental results show the Recursive Feature Elimination technique with the Adaptive Bagging classifier outperforms the others.

**ACKNOWLEDGEMENTS**

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, Dr. Vidushi for her guidance, help and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions. Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Abhinav Choudhary**

**Akshay Choudhary**

**Akshit Sherawat**

**TABLE OF CONTENTS**

Page no.

Declaration ⅱ

Certificate iii

Abstract ⅳ

Acknowledgements ⅴ

Table of contents ⅵ

List of Figures ⅶ

Chapter 1: 1

1. Overview 1
2. Problem Statement 1
3. Objectives 1
4. Scope 2
5. Hardware / Software used in project 3

Chapter 2: Literature Review4

Chapter 3: Feasibility Study 5

1. Technical feasibility 5
2. Operational feasibility 5
3. Behavioural feasibility 5

Chapter 4: System requirements 6

1. Functional requirements 6
2. Non-functional requirements 6

Chapter 5: Methodology 7

1. Introduction To Machine Learning 7
2. Training The Data 7
   * 1. Supervised Learning 8
     2. Unsupervised Learning 8
3. Method in Supervised 9
   * 1. Classification 9
     2. Regression 9
4. System Architecture 10

Chapter 6: Algorithms 11

1. Random Forest Classifier 11
2. K Nearest Neighbours 12
3. Decision Tree Classifier 13
4. Data Sets 14
5. Packages 14
   * 1. Data Manipulation Packages 15
     2. Data Building Packages 16
     3. Data Visualization Packages 16

Chapter 7: System Demonstration 19

Chapter 8: Modules Description 90

Chapter 9: Conclusion 91

Chapter 10: References 92

**LIST OF FIGURES**

**Figure No. Name of Figure Page No.**

5.7.1 Classification vs Regression 9

5.7.2 System Architecture 11

6.1 Random Forest 12

6.2 KNN 13

6.3 Decision Tree 13

**CHAPTER 1**

**INTRODUCTION**

* 1. **Overview**

Agriculture is the one amongst the substantial area of interest to society since a large portion of food is produced by them. Currently, many countries still experience hunger because of the shortfall or absence of food with a growing population. Expanding food production is a compelling process to annihilate famine. Developing food security and declining hunger by 2030 are beneficial critical objectives for the United Nations. Hence crop protection; land assessment and crop yield prediction are of more considerable significance to global food production.

This project uses Python 3.6 for the programming in a scientific development environment called the PyCharm. Various data manipulation, machine learning and visualization packages are used to create and analyze the dataset using a traditional machine learning model. A Data Visualization tool called Tableau is used to interpret the results provided by the model after analysis to represent and act as a proof for the intended result.

* 1. **Problem Statement**

Failure of farmers to decide on the best suited crop for his land using traditional and non-scientific methods is a serious issue for a country where approximately 50 percent of the population is involved in farming. both availability and accessibility of correct and up to date information hinders potential researchers from working on developing country case studies. with resources within our reach we have proposed a system which can address this problem by providing predictive insights on crop sustainability and recommendations based on machine learning models trained considering essential environmental and economic parameters.

* 1. **Objective**

In Indian economy and employment agriculture plays major role. The most common problem faced by the Indian farmers is they do not opt crop based on the necessity of soil, as a result they face serious setback in productivity.

This problem can be addressed through precision agriculture. This method takes three parameters into consideration, viz: soil characteristics, soil types and crop yield data collection based on these parameters suggesting the farmer suitable crop to be cultivated.

Precision agriculture helps in reduction of non-suitable crop which indeed increases productivity, apart from the following advantages like efficacy in input as well as output and better decision making for farming.

This method gives solutions like proposing a recommendation system through an ensemble model with majority voting techniques using Random Forest and K - Nearest Neighbor as learner to recommend suitable crop based on soil parameters with high specific accuracy and efficiency.

* 1. **Scope**
* Our proposed system is an application which predicts name of the crop as well as calculate its corresponding yield.
* Name of the crop is determined by several features like temperature, humidity, wind-speed, rainfall etc. and yield is determined by the area and production.
* In this project KNN and Random Forest is used for prediction. It will attain the crop prediction with best accurate values.
  1. **Hardware & Software used**

Hardware Requirements

|  |  |
| --- | --- |
| **S. N.** | **Description** |
| 1 | PC with 10 GB or more Hard disk. |
| 2 | PC with 2 GB RAM. |
| 3 | PC with core i3 or above processor. |

Software Requirements

|  |  |  |
| --- | --- | --- |
| **S. N.** | **Description** | **Type** |
| 1 | Operating System | Windows 10 or 11 or Ubuntu 18.04 or above |
| 2 | Language | Python 3 |
| 3 | Front End | React 17 |
| 4 | IDE | Google Colab, VS Code,PyCharm |
| 5 | Browser | Chrome, Firefox, Edge |

**CHAPTER 2**

**LITERATURE REVIEW**

In a research carried out by Zaminur Rahman a comparative study of several machine learning techniques has been carried out. They have carried out the classification using the data of Bangladesh. Considered the six district soil data and used the geographical features for classification. They have used k Nearest Neighbour, Bagged tree and SVM finally compared the results of three algorithms and brought out a model for classifying the soil types and the suitable crop that can be cultivated in that particular soil type[3].Among the used three algorithms SVM has obtained the average accuracy.

In a research carried out by Leisa J.Armstrong a comparative study of data mining algorithms. They have used a large dataset extracted from the Australian Department of Agriculture and Food(AGRIC) to conduct the research[4].

In an approach carried out by Jay Gholap carried out a modal to classify the soil based on fertility. The dataset was collected from the soil testing laboratories of Pune District. They have used WEKA tool for developing an automated system[5].

Chiranjeevi M. N carried out a research for classifying the soil types so that it can be useful for the farmers for analyzing the type o soil and the crop that can be cultivated so that there will a good yield and profit. They have considered the data mining algorithms for classifying the soil.They have used algorithms such as J48 decision tree classifier and NaÃ¯ve bayes classifier among these two algorithms NaÃ¯ve bayes has obtained the maximum accuracy of 98%[6].

**CHAPTER 3**

**FEASIBILITY STUDY**

Feasibility Study in Software Engineering is a study to evaluate feasibility of proposed project or system. Feasibility study is one of stage among important four stages of Software Project Management Process. As name suggests feasibility study is the feasibility analysis or it is a measure of the software product in terms of how much beneficial product development will be for the organization in a practical point of view. Feasibility study is carried out based on many purposes to analyze whether software product will be right in terms of development, implantation, contribution of project to the organization etc.

**3.1 Technical Feasibility**

In Technical Feasibility current resources both hardware software along with required technology are analyzed/assessed to develop project. This technical feasibility study gives report whether there exists correct required resources and technologies which will be used for project development. Along with this, feasibility study also analyzes technical skills and capabilities of technical team, existing technology can be used or not, maintenance and up-gradation is easy or not for chosen technology etc.

**3.2 Operational Feasibility**

In Operational Feasibility degree of providing service to requirements is analyzed along with how much easy product will be to operate and maintenance after deployment. Along with this other operational scopes are determining usability of product, Determining suggested solution by software development team is acceptable or not etc.

**3.3 Economic Feasibility**

In Economic Feasibility study cost and benefit of the project is analyzed. Means under this feasibility study a detail analysis is carried out what will be cost of the project for development which includes all required cost for final development like hardware and software resource required, design and development cost and operational cost and so on. After that it is analyzed whether project will be beneficial in terms of finance for organization or not.

**CHAPTER 4**

**SYSTEM REQUIREMENTS**

**4.1 Functional requirements**

* The system must provide clear information about workflow of model.
* The system must provide clear and fully detailed information history results.
* The system must provide clear prediction results.
* The system must provide clear accuracy of the classifier.
* The system must provide clear name of the classifier used.
* The system should navigate with the URL enters by the user.
* The system should clarify if the wrong URL is entered by the user.
* The system should store the feedback of the user.
* The system should provide information if there is any error or loading.

**4.2 Non-Functional requirements**

* The system shall store multiple feedbacks inputed by the user,
* The system should show the predict crops frequently user should not have to wait for a long period.
* The system should have a data training.

**CHAPTER 5**

**Methodology**

**5.1 Introduction To Machine Learning**

Machine learning is a branch of computer science that employs statistical techniques to enable computer systems to "learn" (i.e., progressively improve performance on a specific task) from data without being explicitly programmed. Arthur Samuel coined the term "machine learning" in 1959. Machine learning, which evolved from the study of pattern recognition and computational learning theory in artificial intelligence, investigates the study and construction of algorithms that can learn from and make predictions on data – such algorithms overcome strictly static programme instructions by making data-driven predictions or decisions, by building a model from sample inputs. Machine learning is used in a variety of computing tasks where designing and programming explicit algorithms with high performance is difficult or impossible; examples include email filtering, network intruder detection, and computer vision. Machine learning is closely related to (and frequently overlaps with) computational statistics, which is also concerned with making predictions using computers. It has strong ties to mathematical optimization, which provides the field with methods, theory, and application domains. Machine learning is frequently confused with data mining, the latter of which focuses on exploratory data analysis and is referred to as unsupervised learning.Machine learning is a method used in the field of data analytics to create complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. Through learning from historical relationships and trends in the data, these analytical models enable researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights."

**5.2 Training The Data**

There are basically two widely-used types of training that can be done to create a model:

1. Supervised Learning
2. Unsupervised Learning
   * 1. **Supervised Learning**

The machine learning task of learning a function that maps an input to an output based on example input-output pairs is known as supervised learning. It derives a function from labelled training data, which consists of a set of training examples. Each example in supervised learning is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm examines the training data and generates an inferred function that can be used to map new examples. In an ideal scenario, the algorithm will be able to correctly determine the class labels for unseen instances. This necessitates that the learning algorithm generalize from the training data to previously unseen situations in a "reasonable" manner.

**5.3.2 Unsupervised Learning**

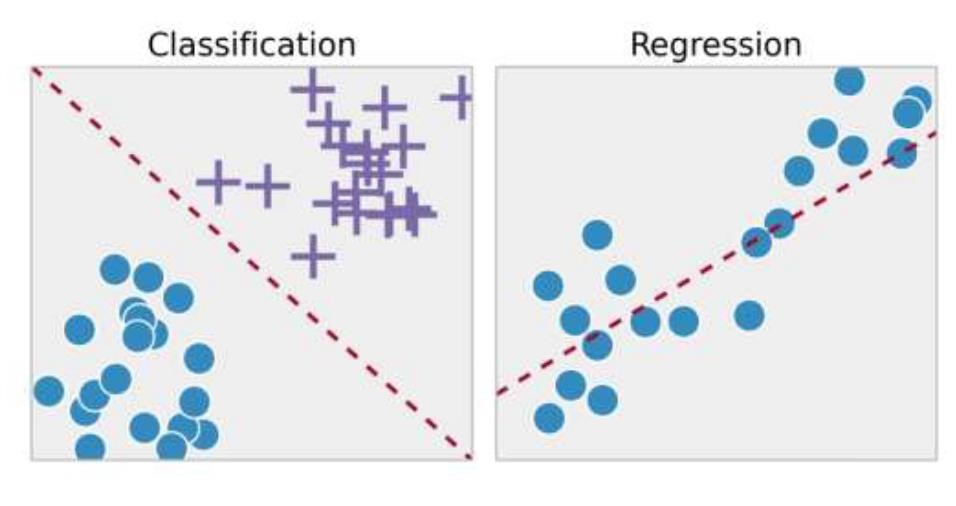
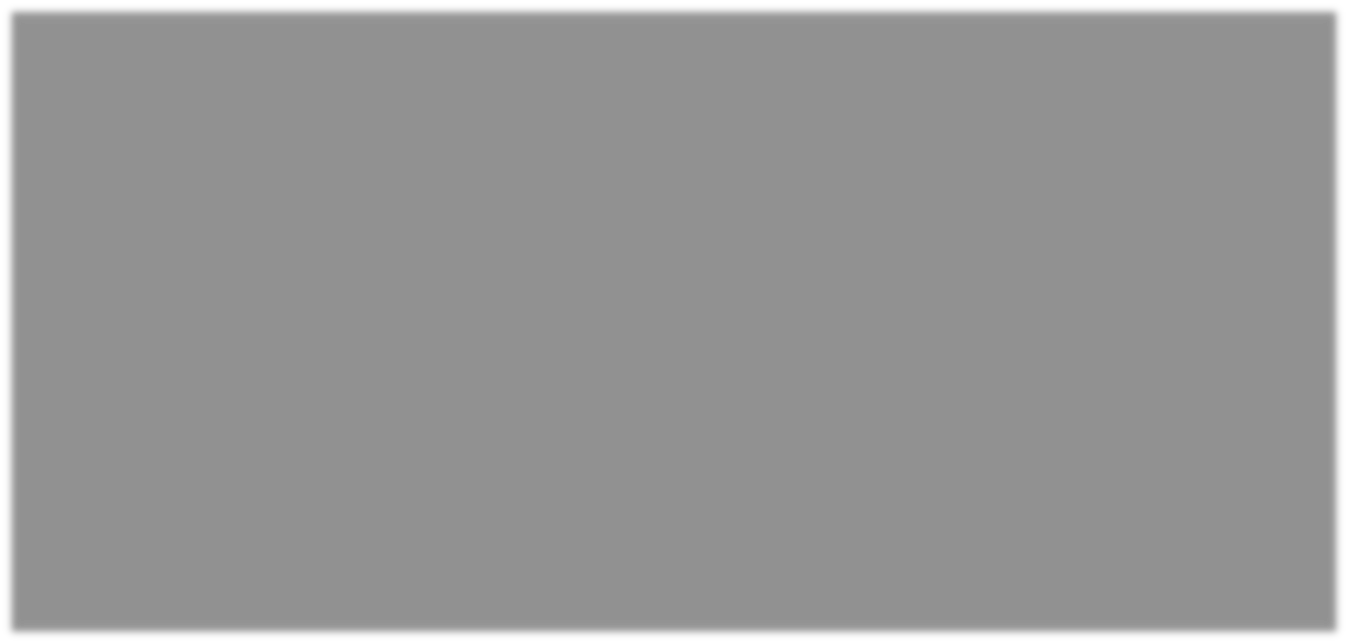
The machine learning task of inferring a function that describes the structure of "unlabeled" data is known as unsupervised machine learning (i.e., data that has not been classified or categorized). Because the examples provided to the learning algorithm are unlabeled, there is no simple way to assess the accuracy of the structure produced by the algorithm—a feature that distinguishes unsupervised learning from supervised learning and reinforcement learning.

The type of training used in this model is **SUPERVISED LEARNING.**

**5.4 Method in Supervised**

Supervised Learning mainly consists of two methods:

* Classification
* Regression



*Fig 5.7.1 Classification vs Regression*

* + 1. **Classification**

Classification is the problem in machine learning of determining which of a set of categories (sub-populations) a new observation belongs to, based on a training set of data containing observations (or instances) whose category membership is known.

Examples include categorizing an email as "spam" or "nonspam," and assigning a diagnosis to a patient based on observed characteristics (gender, blood pressure, presence or absence of certain symptoms, etc.). Pattern recognition is demonstrated by classification. A classifier is an algorithm that implements classification, particularly in a concrete implementation. Clustering is the corresponding unsupervised procedure, and it involves categorizing data based on some measure of inherent similarity or distance.

* + 1. **Regression**

Regression analysis calculates the dependent variable's conditional expectation given the independent variables – that is, the average value of the dependent variable when the independent variables are held constant. Less frequently, the emphasis is on a quantile or other location parameter of the dependent variable's conditional distribution given the independent variables. The regression function, which is a function of the independent variables, must be estimated in all cases. Regression analysis is widely used for forecasting and prediction. It is also used to determine which independent variables are related to the dependent variable and to investigate the nature of these relationships. Regression analysis can be used to infer causal relationships between independent and dependent variables in limited circumstances. This, however, can lead to illusions or false relationships, so exercise caution; for example, correlation does not prove causation.

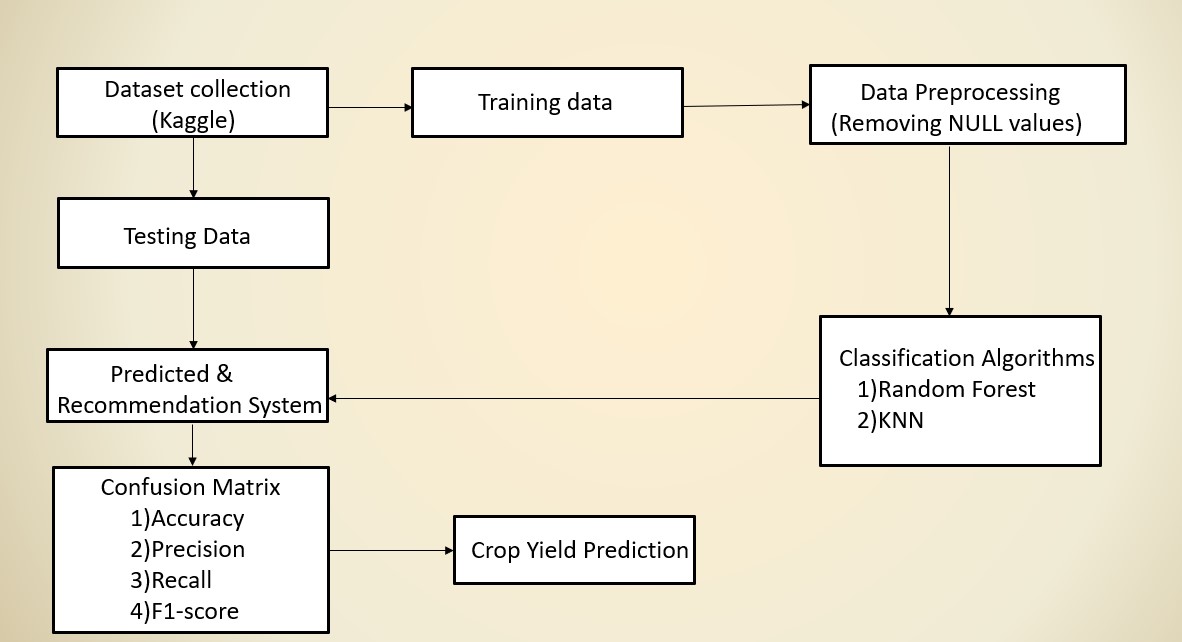
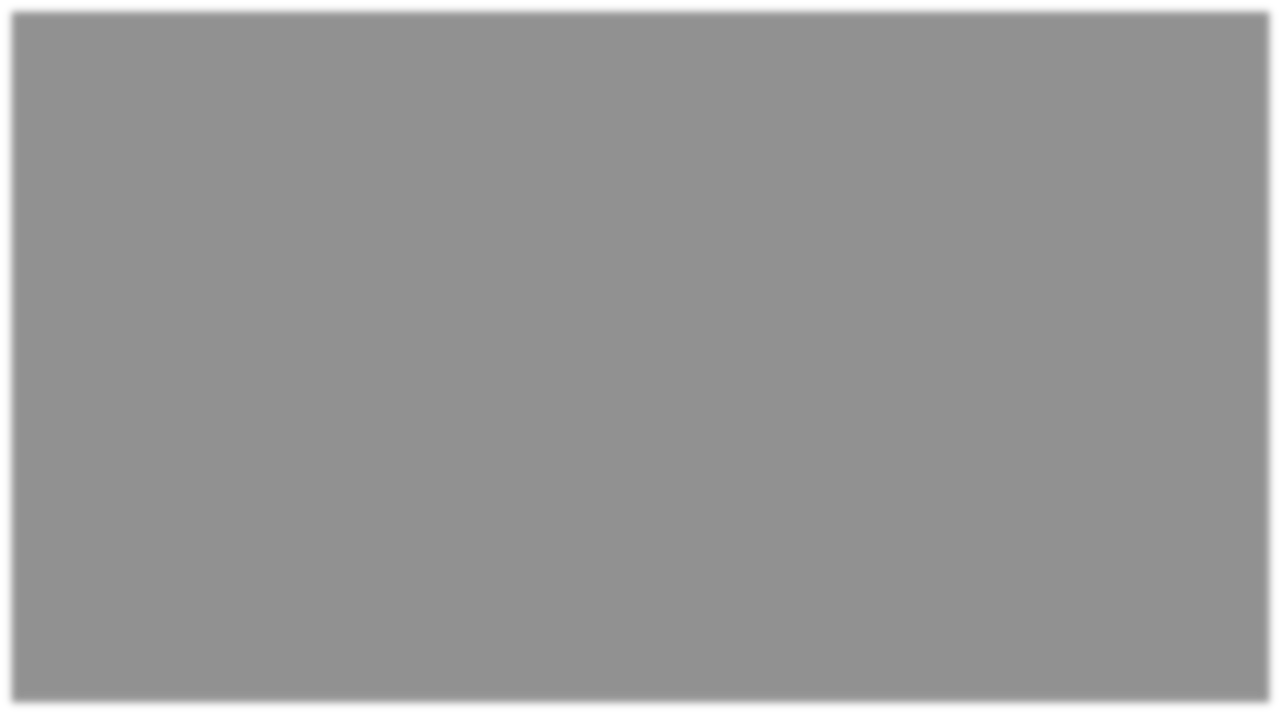
The type used in this model is **CLASSIFICATION** and so, more focus will be given on it.

1. **System Architecture**

Crop yield is extremely useful information for farmers. Understanding the yield can help you save money by lowering your losses. Crop yields were previously predicted by experienced farmers. The proposed system works in a similar manner. It uses previous data to forecast future yields. Crop productivity is most affected by weather and fertilizers. The accuracy of this prediction is determined by the accuracy of the information provided.

As a result, the proposed method predicts yield and reduces loss. The expected system assumes the role of an experienced farmer. It is, however, more precise and takes into account a number of additional parameters. There are several factors to consider, including soil condition, Temperature, pH, humidity etc.

*Fig 5.7.2 System Architecture*



**CHAPTER 6**

**Algorithms**

* 1. **Random Forest Classifier**

Random Forest is a supervised machine learning technique that can be applied to Classification and Regression problems. It is based on the concept of ensemble learning, which is the process of combining multiple classifiers to solve a complex problem and improve the model's performance. Random Forest is a classifier that uses the average of a number of decision trees from a given dataset to improve the predictive accuracy of the data set. Instead of relying on a single decision tree, the random forest takes the predictions from each tree and predicts the final output based on the majority vote of predictions.

**.**

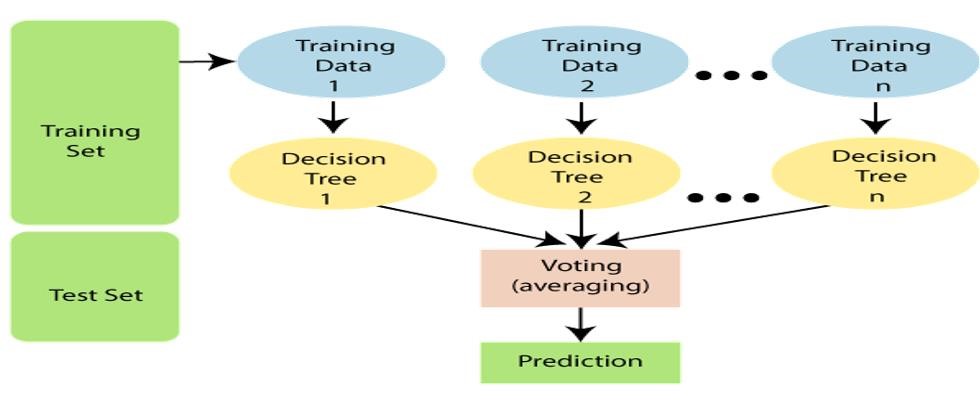
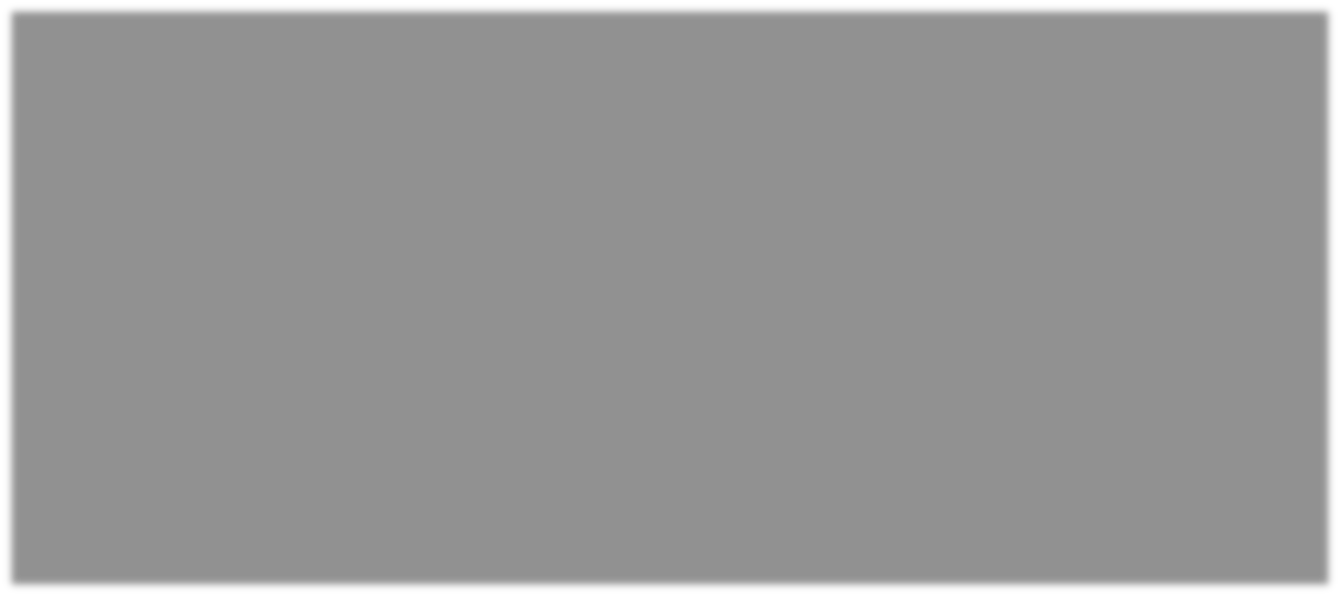
*Fig*

*6…..*

*.*

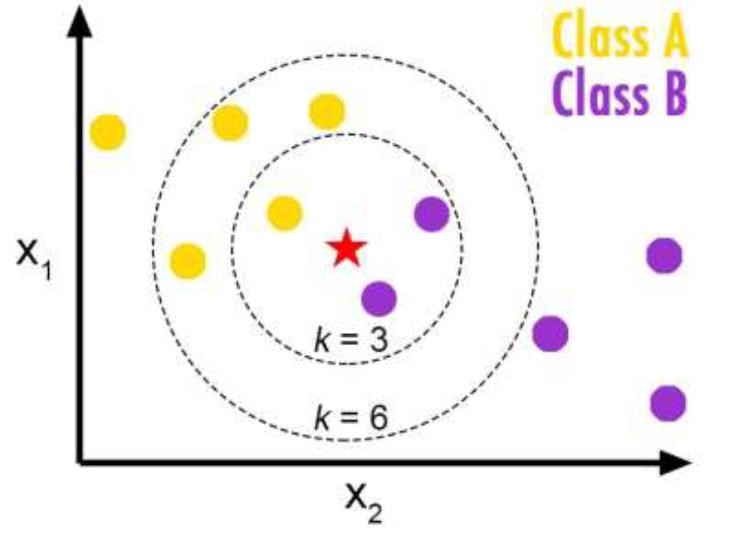
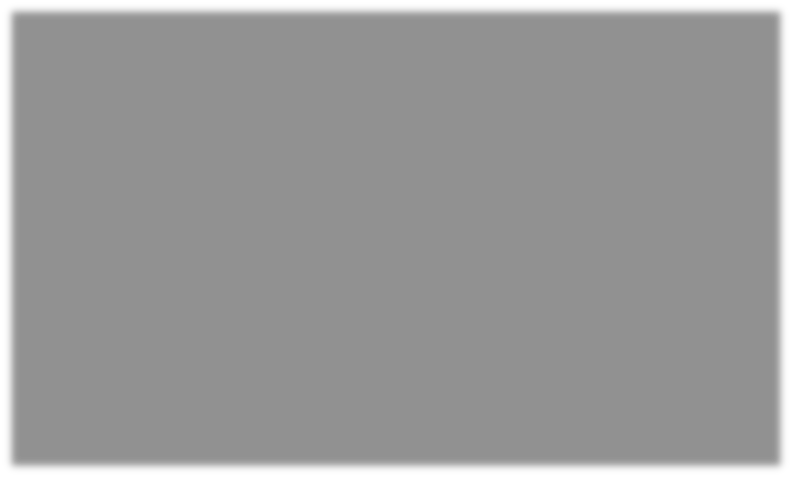
*1*

*Random Forest*



* 1. **K Nearest Neighbours**

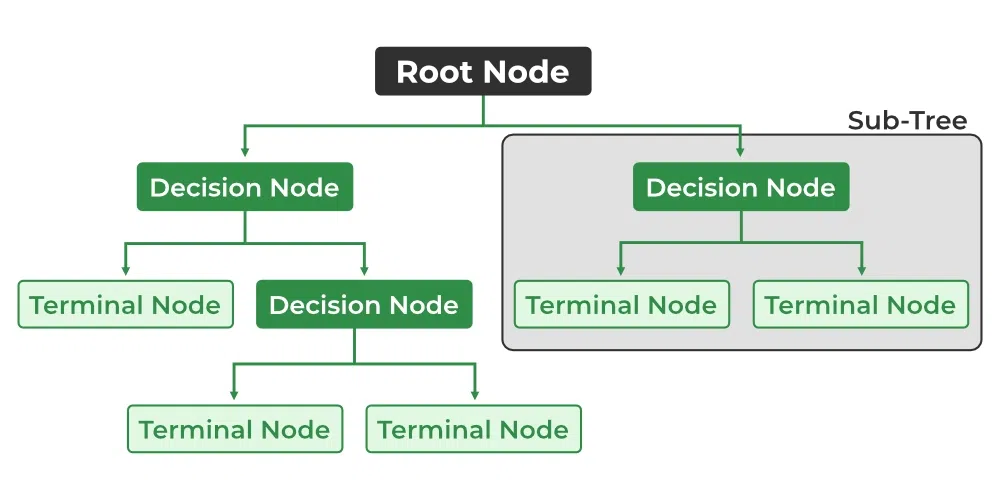
The k-nearest neighbors‘ algorithm (K-NN) is a nonparametric method for classification that is used in N pattern recognition. The result is class membership. A majority vote of its neighbors classifies an object, with the object assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, the object is simply assigned to the class of the object's single nearest neighbor.



*Fig 6.2 KNN*

* 1. **Decision Tree Classifier**

A decision tree is a flowchart-like tree structure where each internal node denotes the feature, branches denote the rules and the leaf nodes denote the result of the algorithm. It is a versatile supervised machine-learning algorithm, which is used for both classification and regression problems. It is one of the very powerful algorithms. And it is also used in Random Forest to train on different subsets of training data, which makes random forest one of the most powerful algorithms in machine learning

 *Fig 6.3 Decision Tree*

* 1. **Data Sets**

The dataset comprising the soil specific attributes which are collected from Kaggle. In addition, similar online sources of general crop data were also used. The crops considered in our model include rice, maize, chickpea, kidney beans, pigeon peas, moth beans, mungbean, black gram, lentil, pomegranate, banana, mango, grapes, watermelon, muskmelon, apple, orange, papaya, coconut, cotton, jute, coffee gives an analysis of the dataset. The number of instances of each crop available in the training dataset is depicted. The attributes considered where Nitrogen(N), Potassium(K), Phosphorus(P), Temperature, Humidity, Ph and Rainfall.

The above stated parameters of soil play a major role in the crop's ability to extract water and nutrients from the soil. For crop growth to their fullest potential, the soil must provide a satisfactory environment for it. Soil is the anchor of the roots. Nitrogen is largely responsible for the growth of leaves on the plant. Phosphorus is largely responsible for root growth and flower and fruit development. Potassium is a nutrient that helps the overall functions of the plant perform correctly. Temperature is a key factor in plant growth and development. Along with the levels of light, carbon dioxide, air humidity, water and nutrients, temperature influences plant growth and ultimately crop yields. Humidity directly influences the water relations of plant and indirectly affects leaf growth, photosynthesis, pollination, occurrence of diseases and finally economic yield. The level of acidity or alkalinity (Ph) is a master variable which affects the availability of soil nutrients. The activity of microorganisms presents in the soil and also the level of exchangeable aluminum can be affected by PH. rainfall can also determine how fast a crop will grow from seed, including when it will be ready for harvesting. A good balance of rain and proper irrigation can lead to faster-growing plants, which can cut down on germination time and the length between seeding and harvest. Hence for the following reasons the above stated parameters are considered for choosing a crop.

* 1. **Packages**

The packages used in this model include:

* Pandas
* Scikit-learn
* Numpy
* Matplotlib‘s pyplot
* Seaborn
  + 1. **Data Manipulation Packages**

 ***Pandas***

* Pandas is a Python package that provides fast, flexible, and expressive data structures that make it simple and intuitive to work with structured (tabular, multidimensional, potentially heterogeneous) and time series data. It intends to be the fundamental high-level building block for performing practical, real-world data analysis in Python. Furthermore, it aspires to be the most powerful and adaptable open-source data analysis and manipulation tool available in any language. It is already well on its way to accomplishing this goal.

* Pandas' two primary data structures, Series (1-dimensional) and Data Frame (2dimensional), handle the vast majority of common use cases in finance, statistics, social science, and many fields of engineering. Data Frame gives R users access to all of R's data. Frame offers and much more. Pandas is built on top of NumPy and is designed to work well in a scientific computing environment alongside many other third-party libraries.

* Written in: Python, Cython and C.

 ***NumPy***

* NumPy is a Python library that adds support for large, multidimensional arrays and matrices, as well as a large collection of high-level mathematical functions for working with these arrays. NumPy is open-source software with numerous contributors.

* NumPy is designed to work with Python's CPython reference implementation, which is a non-optimizing bytecode interpreter. Algorithms written for this version of Python are frequently much slower than compiled equivalents.
* NumPy addresses the slowness issue in part by providing multidimensional arrays as well as functions and operators that operate efficiently on arrays, which necessitates rewriting some code, primarily inner loops, in NumPy.

* Because they are both interpreted, NumPy in Python provides functionality comparable to MATLAB, and they both allow the user to write fast programmes as long as most operations work on arrays or matrices rather than scalars.

* In comparison, MATLAB has a plethora of additional toolboxes, most notably Simulink, whereas NumPy is inextricably linked with Python, a more modern and comprehensive programming language. Additionally, there are complementary Python packages available; SciPy is a library that adds more MATLAB-like functionality, and Matplotlib is a plotting package that provides MATLAB-like plotting functionality.

* Written in: Python and C
  + 1. **Data Building Packages**

###  Scikit-learn

* Scikit-learn (formerly scikits. learn) is a free software machine learning library written in Python. It includes support vector machines (svm), random forest, gradient boosting, kmeans, and DBSCAN as classification, regression, and clustering algorithms, and is designed to work with the Python numerical and scientific libraries NumPy and SciPy.
* It was built on NumPy, SciPy, and Matplotlib.
* Written in: Python, Cython, C and C++.
  + 1. **Data Visualization Packages**

###  Scikit-plot

* Scikit-plot is the result of a dreadful realization by an unartistic data scientist that visualization is one of the most important components of the data science process, not just an afterthought.

* When you're looking at a colored heatmap of a confusion matrix complete with class labels rather than a single-line dump of numbers enclosed in brackets, it's much easier to gain insights. Furthermore, if you ever need to present your results to someone 15 (virtually any time anyone hires you to do data science), you show them visualizations, not a bunch of Excel numbers. Overall, it is a simple library for adding plotting functionality to a scikit-learn object.

* Written in: Python, Cython, C and C++.

###  Matplotlib’s pyplot

* Matplotlib is a Python 2D plotting library that generates high-quality figures in a variety of hardcopy and interactive formats across platforms. Matplotlib can be used in Python scripts, as well as the Python and IPython libraries.

* Shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits are all available.

* It provides an object-oriented API for integrating plots into applications that use general-purpose GUI toolkits such as Tkinter, wxPython, Qt, or GTK+.

* matplotlib. Pyplot provides a MATLAB-like plotting framework

* Pylab is a namespace that combines pyplot and NumPy. This is convenient for interactive work, but it is recommended that the namespaces be kept separate for programming.

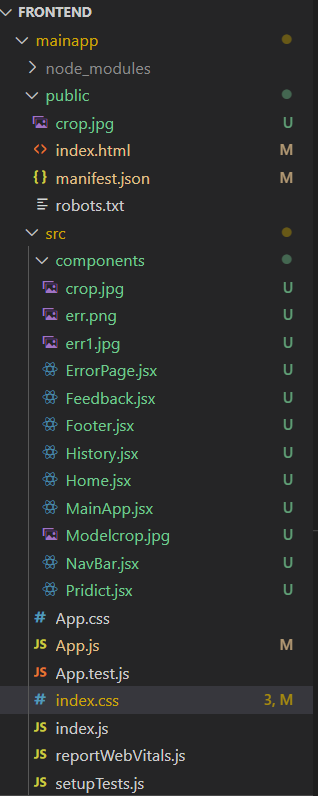
* Written in: Python.

**CHAPTER 7**

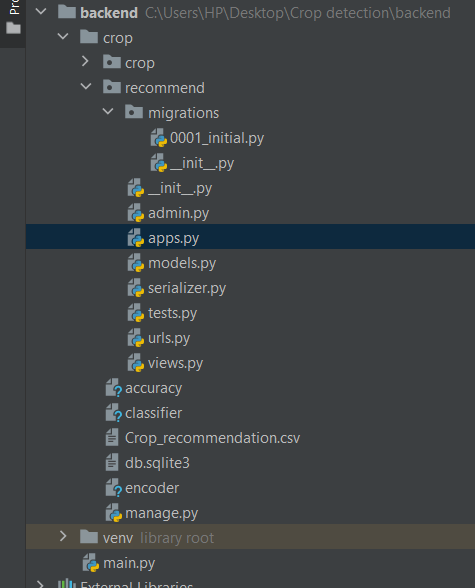
**SYSTEM DEMONSTRATION**

**7.1 Snapshots**

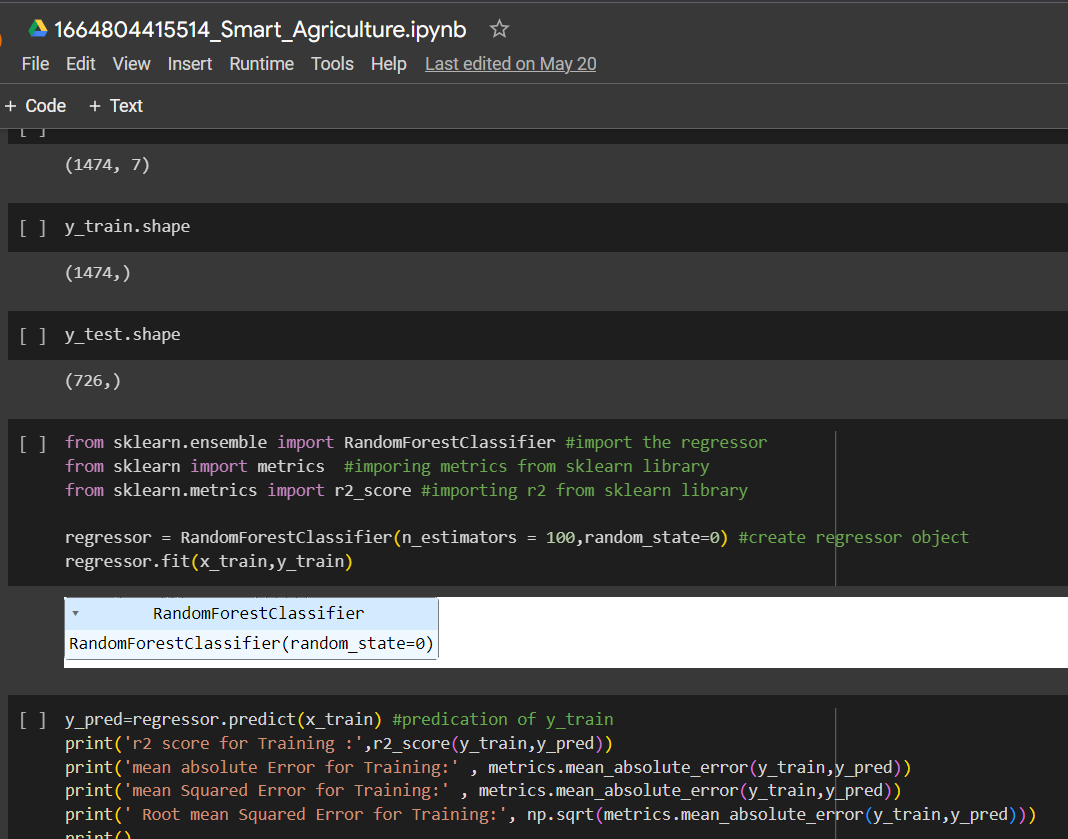
* **Frontend File Structure**

****

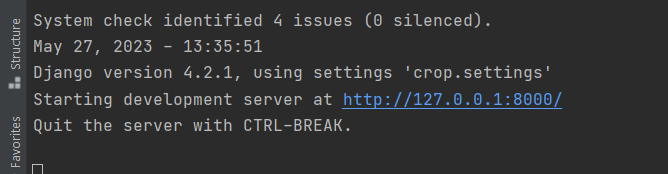
* **Backend File Structure**

****

* **Pre-processing and Model Building on Google Colab**

****

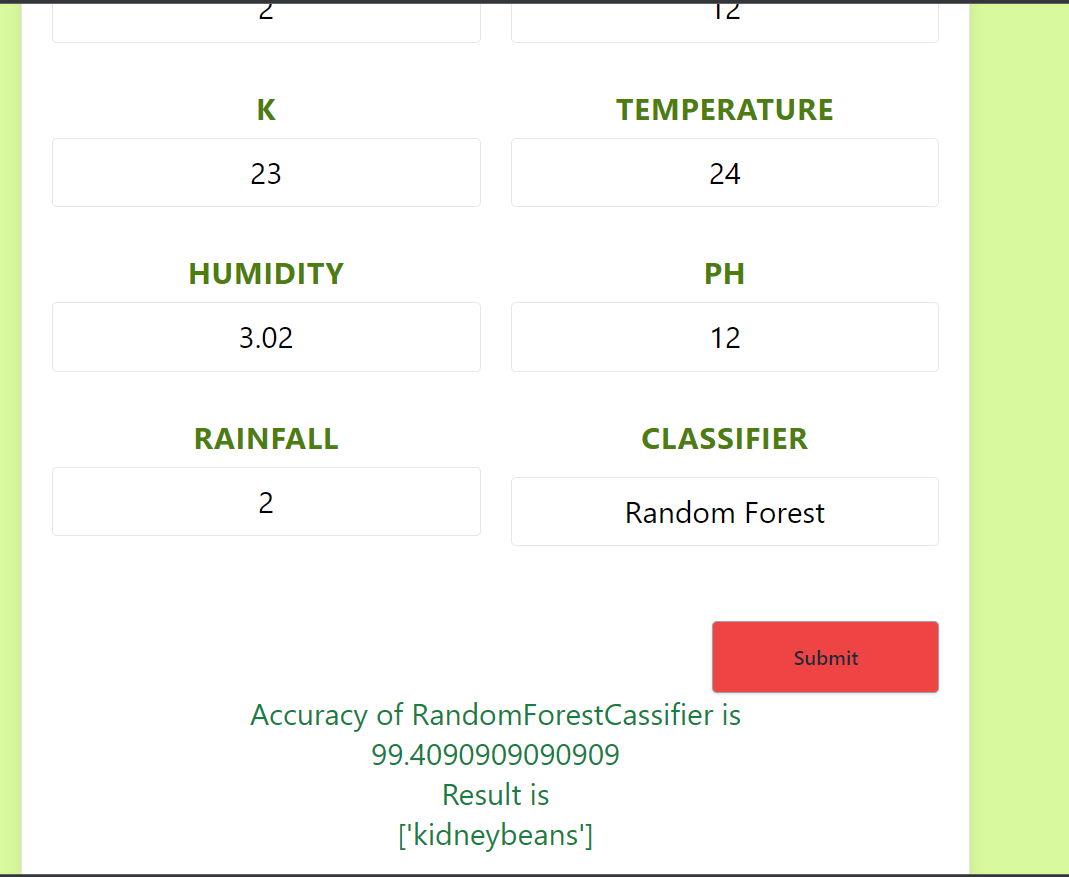
* **Running Server**

****

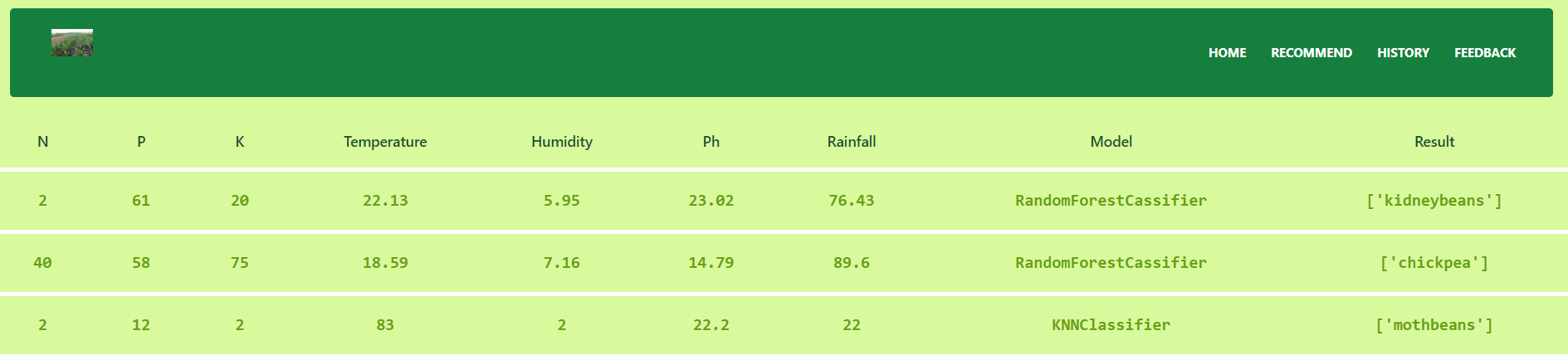
* **Homepage**

****

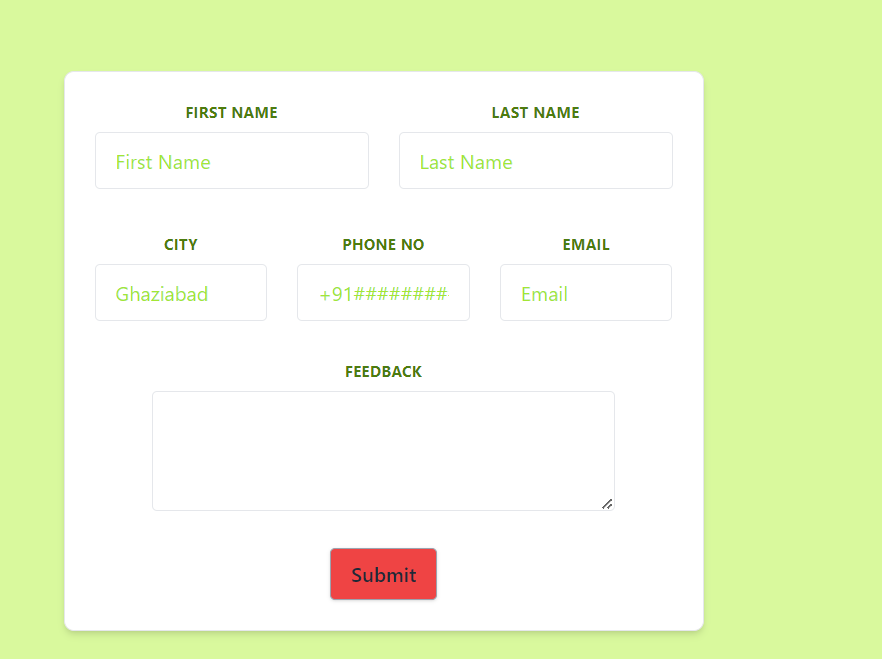
* **Recommend Page**

****

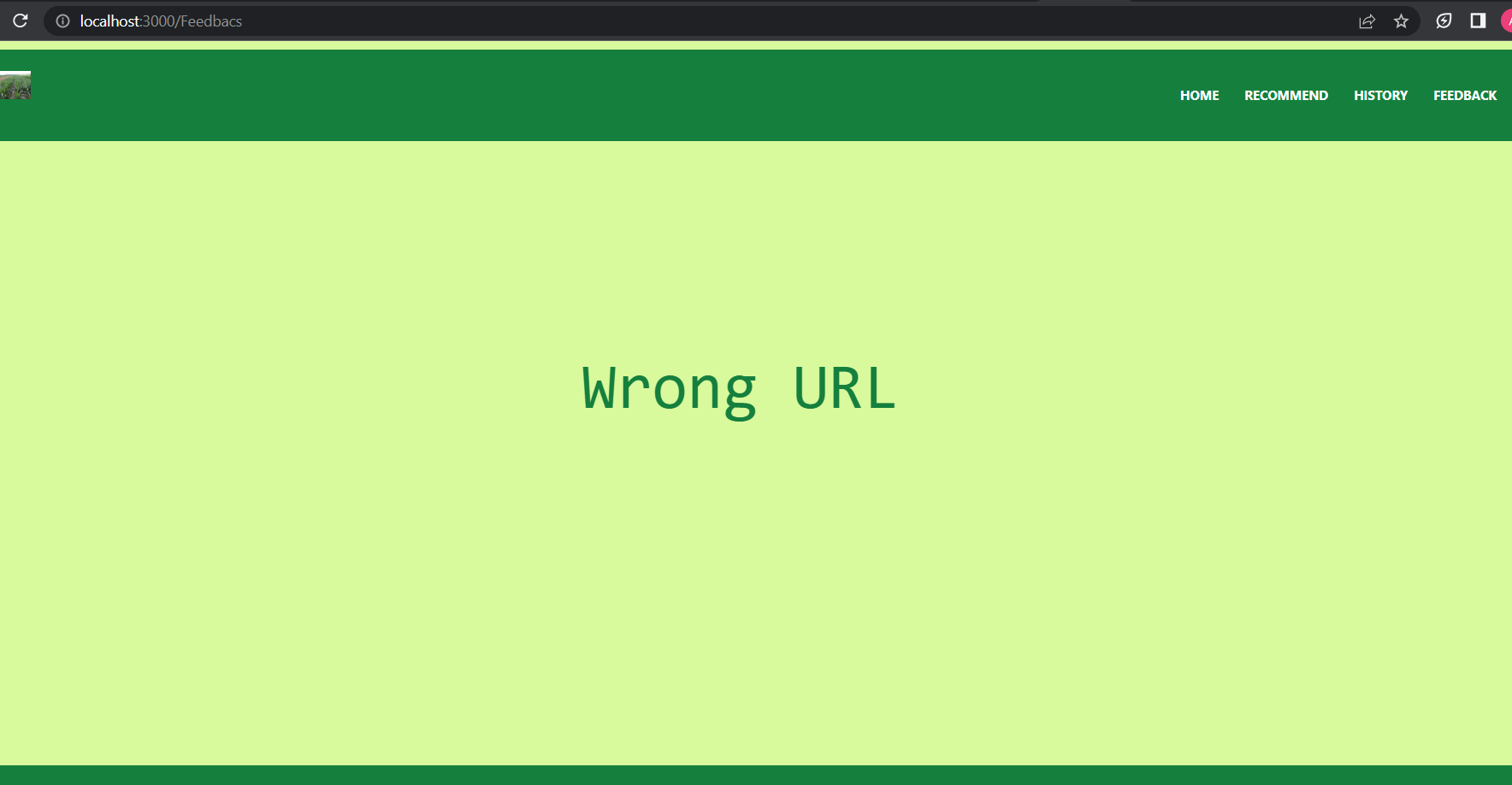
* **History Page**

****

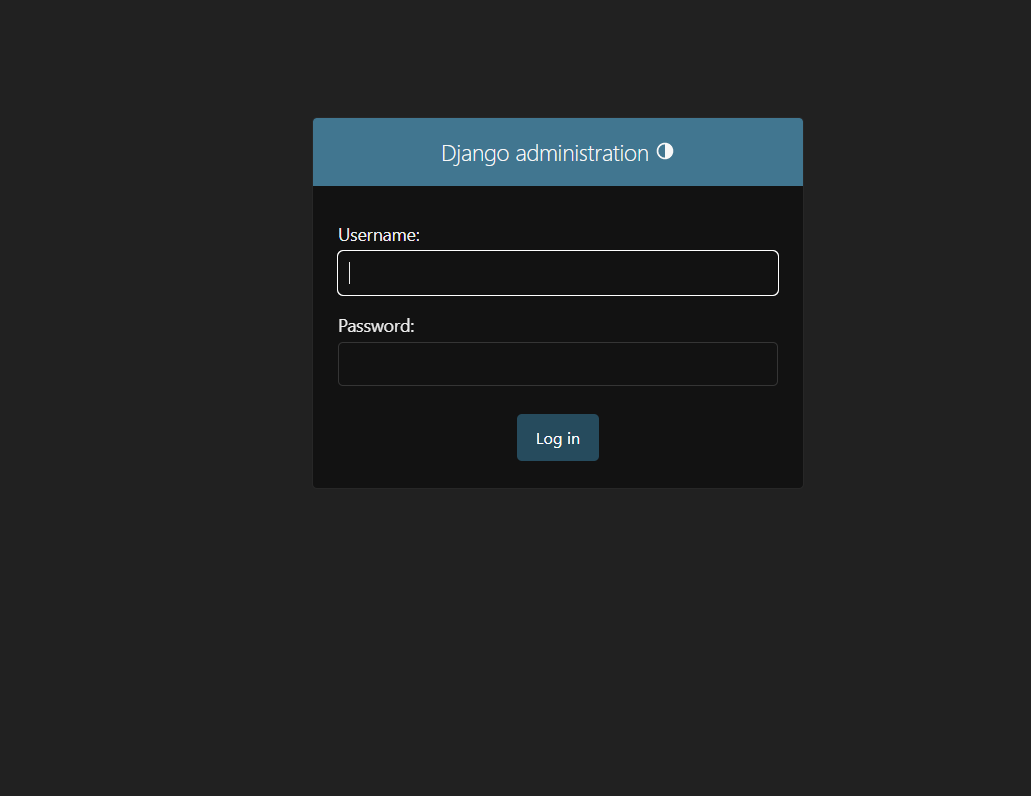
* **Feedback Page**

****

* **ErrorPage**

****

* **Admin Page**

****

**7.2 Code**

* **Frontend**
* **Index.js**

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

* **App.css**

.App {

text-align: center;

}

.App-logo {

height: 40vmin;

pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

.App-logo {

animation: App-logo-spin infinite 20s linear;

}

}

.App-header {

background-color: #282c34;

min-height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-size: calc(10px + 2vmin);

color: white;

}

.App-link {

color: #61dafb;

}

@keyframes App-logo-spin {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

* **App.js**

import MainApp from './components/MainApp.jsx';

function App() {

return (

<div>

<MainApp/>

</div>

);

}

export default App;

* **MainApp**

import React, { Component } from 'react';

import Footer from './Footer.jsx';

import NavBar from './NavBar.jsx';

class MainApp extends Component {

state = { }

render() {

return (<>

<NavBar/>

<Footer/>

</>);

}

}

export default MainApp;

* **Home.jsx**

//import { useNavigate } from 'react-router-dom';

//import pridict from '/pridict.jsx';

import img from './Modelcrop.jpg';

const Home=()=> {

return (<>

<div className='App'>

<br/>

<div className='flex justify-center'>

<div className="text-center p-6 max-w-5/6 w-4/5 bg-white relative rounded-lg border border-gray-200 shadow-lg dark:bg-white dark:border-white-700 text-left text-lime-700"><h1 className='text-begin font-bold text-3xl font-serif text-green-700' >Introduction</h1><br/><p className='font-serif'>Agriculture in India plays a predominant role in economy and employment. The common problem existing among the Indian farmers are they don't choose the right crop based on their soil requirements. Due to this they face a serious setback in productivity. This problem of the farmers has been addressed through precision agriculture. Precision agriculture is a modern farming technique that uses research data of soil characteristics, soil types, crop yield data collection and suggests the farmers the right crop based on their site-specific parameters. This reduces the wrong choice on a crop and increase in productivity.<br/>

<br/>&emsp;&emsp;&emsp;&emsp;<span className='bg-lime-50 'href='/'>There are several factors that effect crops good growth we will use these parameters to recommend the crop <br/><li><ul>N - ratio of Nitrogen content in soil</ul>

<ul>P - ratio of Phosphorous content in soil</ul>

<ul>K - ratio of Potassium content in soil</ul>

<ul>temperature - temperature in degree Celsius</ul>

<ul>humidity - relative humidity</ul>

<ul>ph - ph value of the soil</ul>

<ul>rainfall - rainfall in mm</ul>

</li></span>

</p></div>

</div>

<br/>

<br/>

<br/>

<div className='flex justify-center'>

<div className="p-6 max-w-5/6 w-4/5 bg-white relative mb-16 rounded-lg border border-gray-200 shadow-lg dark:bg-white dark:border-white-700 text-left text-lime-700">

<h1 className='text-center text-3xl text-green-700 font-serif font-bold'>Work flow of Model</h1>

<img src={img} alt="model"/>

</div>

</div>

</div>

</>);

}

export default Home;

* **Navbar.jsx**

import React, { Component } from 'react';

import img from './crop.jpg';

import Feedback from './Feedback';

import Home from './Home';

import Pridict from './Pridict';

import History from './History.jsx';

import {BrowserRouter as Router

,Route,Routes,Link} from 'react-router-dom';

import ErrorPage from './ErrorPage';

class NavBar extends Component {

state = { }

// prevent=(event)=>{

// event.preventDefault();

// }

render() {

return (<>

<Router>

<div className='App bg-lime-200'>

<div className="flex flex-wrap py-2">

<div className="w-full px-4">

<nav className="relative flex flex-wrap items-center justify-between px-2 py-3 bg-green-700 rounded">

<div className="container px-4 mx-auto flex flex-wrap items-center justify-between">

<div className="w-full relative flex justify-between lg:w-auto px-4 lg:static lg:block lg:justify-start">

<Link to={"/Home"}

className="text-sm font-bold leading-relaxed inline-block mr-4 py-2 whitespace-nowrap uppercase text-white"

><div className='object-cover h-10 w-10'>

<img src={img} alt="crop"/>

</div>

</Link>

<button

className="text-white cursor-pointer text-xl leading-none px-3 py-1 border border-solid border-transparent rounded bg-transparent block lg:hidden outline-none focus:outline-none"

type="button"

>

<i className="fas fa-bars"></i>

</button>

</div>

<div

id="example-navbar-info"

>

<ul className="flex flex-col lg:flex-row list-none lg:ml-auto">

<li className="nav-item">

<Link to="/Home"

className="px-3 py-2 flex items-center text-xs uppercase font-bold leading-snug text-white hover:opacity-75"

>

Home

</Link>

</li>

<li className="nav-item">

<Link to="./Pridict"

className="px-3 py-2 flex items-center text-xs uppercase font-bold leading-snug text-white hover:opacity-75"

>

Recommend

</Link>

</li>

<li className="nav-item">

<Link to="./History"

className="px-3 py-2 flex items-center text-xs uppercase font-bold leading-snug text-white hover:opacity-75"

>

History

</Link>

</li>

<li className="nav-item">

<Link to="./Feedback"

className="px-3 py-2 flex items-center text-xs uppercase font-bold leading-snug text-white hover:opacity-75"

>

FeedBack

</Link>

</li>

</ul>

</div>

</div>

</nav>

</div>

</div>

<Routes>

<Route exact path='/' element ={<Home/>}></Route>

<Route exact path='/Home' element ={<Home/>}></Route>

<Route exact path='/FeedBack' element={<Feedback/>}></Route>

<Route exact path='/Pridict' element={<Pridict/>}></Route>

<Route exact path='/History' element={<History/>}></Route>

<Route exact path='\*' element={<ErrorPage/>}></Route>

</Routes>

</div>

</Router>

</>

);

}

}

export default NavBar;

* **Footer**

import React, { Component } from 'react';

class Footer extends Component {

state = { }

render() {

return (

<>

<div className=''>

<footer className="p-4 bg-green-700 rounded-lg shadow md:flex md:items-center md:justify-between md:p-6 ">

<span className="text-sm text-white sm:text-center text-font-sherif dark:text-white">© 2022 <a href="https://www.linkedin.com/in/abhinav-choudhary-a1071a1b0/" className="hover:underline">Abhinav ch™</a>. All Rights Reserved.

</span>

</footer>

</div>

</>

);

}

}

export default Footer;

* **Feedback**

import React, { Component } from 'react';

import axios from 'axios';

class Feedback extends Component {

state = {

first:"",

last:"",

city:"",

phone:"",

email:"",

feedback:"",

}

handleOnChange = (event) => {

const { name, value } = event.target;

this.setState({

[name]: value,

});

};

prevent=(response)=>{

response.preventDefault();

const data=this.state;

console.log(data)

axios.post("http://127.0.0.1:8000/feedback",data).then(()=>{

alert("success")

}).catch((error)=>{

alert(error)

})

alert("form submited");

}

render() {

const {fi,la,ci,ph,em,fe}=this.state;

return (<>

<div className='App'>

<div className='text-center flex flex-col justify-center items-center border-gray-300 pt-24 pb-28'>

<div className="block p-6 max-w-lg bg-white rounded-lg border border-gray-200 shadow-md dark:bg-white dark:border-gray-200 dark:hover:border-lime-500 ">

<form method="POST" action="/feedback" className="w-full max-w-lg " name="feedback" onSubmit={this.prevent} >

<div className="flex flex-wrap -mx-3 mb-6 ">

<div className="w-full md:w-1/2 px-3 mb-6 md:mb-0">

<label className="block uppercase tracking-wide text-lime-700 text-xs font-bold mb-2" htmlFor="grid-first-name">

First Name

</label>

<input className="appearance-none block w-full text-lime-600 placeholder-lime-400 border hover:shadow-xl rounded py-3 px-4 mb-3 leading-tight focus:outline-none focus:bg-white focus:border-lime-400" id="grid-first-name" type="text" placeholder='First Name' name="first" onChange={this.handleOnChange} value={fi} required/>

</div>

<div className="w-full md:w-1/2 px-3">

<label className="block uppercase tracking-wide text-lime-700 text-xs font-bold mb-2" htmlFor="grid-last-name">

Last Name

</label>

<input className="appearance-none block w-full hover:shadow-md border border-gray-200 rounded py-3 px-4 leading-tight focus:outline-none focus:bg-white focus:border-lime-400 text-lime-600 placeholder-lime-400" id="grid-last-name" name="last" type="text" placeholder="Last Name" onChange={this.handleOnChange} value={la} required/>

</div>

</div>

<div className="flex flex-wrap -mx-3 mb-2">

<div className="w-full md:w-1/3 px-3 mb-6 md:mb-0">

<label className="block uppercase tracking-wide text-lime-700 text-xs font-bold mb-2" htmlFor="grid-city">

City

</label>

<input className="appearance-none block w-full hover:shadow-md text-lime-600 placeholder-lime-400 border border-gray-200 rounded py-3 px-4 leading-tight focus:outline-none focus:bg-white focus:border-lime-400" id="grid-city" name="city" type="text" onChange={this.handleOnChange} value={ci} placeholder="Ghaziabad" required/>

</div>

<div className="w-full md:w-1/3 px-3 mb-6 md:mb-0">

<label className="block uppercase tracking-wide text-lime-700 text-xs font-bold mb-2" htmlFor="grid-phone">

Phone no

</label>

<input className="appearance-none block w-full hover:shadow-md text-lime-600 placeholder-lime-400 border border-gray-200 rounded py-3 px-4 leading-tight focus:outline-none focus:bg-white focus:border-lime-400" id="grid-phone" name="phone" type="tel" onChange={this.handleOnChange} value={ph} placeholder="+91##########" required/>

</div>

<div className="w-full md:w-1/3 px-3 mb-6 md:mb-0">

<label className="block uppercase tracking-wide text-lime-700 text-xs font-bold mb-2" htmlFor="grid-email">

Email

</label>

<input className="appearance-none block w-full text-lime-600 placeholder-lime-400 hover:shadow-md border border-gray-200 rounded py-3 px-4 leading-tight focus:outline-none focus:bg-white focus:border-lime-400" id="grid-email" type="email" onChange={this.handleOnChange} value={em} name="email" placeholder="Email" required/>

</div>

</div>

<br/>

<label className="block uppercase tracking-wide text-lime-700 text-xs font-bold mb-2 h-4" htmlFor="grid-feedback">

Feedback

</label>

<textarea className="text-lime-600 placeholder-lime-400 selection: resize border rounded focus:outline-none hover:shadow-md focus:shadow-outline focus:border-lime-400 h-24 w-4/5" onChange={this.handleOnChange} value={fe} name="feedback" required id="grid-feedback"></textarea>

<br/>

<br/>

<button className="bg-red-500 hover:bg-rose-400 text-gray-800 font-semibold py-2 px-4 border border-gray-400 rounded shadow" onClick={this.handleOnChange}>

Submit

</button>

</form>

</div>

</div>

</div>

</>);

}

}

export default Feedback;

* **ErrorPage**

import React, { Component } from 'react';

class ErrorPage extends Component {

state = { }

render() {

return (<>

<div className='h-96 text-center'>

<p className='text-green-700 mt-48 font-mono text-6xl'>Wrong URL</p>

</div>

</>);

}

}

export default ErrorPage;

* **Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<link rel="icon" href="%PUBLIC\_URL%/crop.jpg" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<meta name="theme-color" content="#000000" />

<meta

name="description"

content="Web site created using create-react-app"

/>

<link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

<!--

manifest.json provides metadata used when your web app is installed on a

user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/

-->

<link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

<!--

Notice the use of %PUBLIC\_URL% in the tags above.

It will be replaced with the URL of the `public` folder during the build.

Only files inside the `public` folder can be referenced from the HTML.

Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

work correctly both with client-side routing and a non-root public URL.

Learn how to configure a non-root public URL by running `npm run build`.

-->

<title>Crop detection</title>

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="root"></div>

<!--

This HTML file is a template.

If you open it directly in the browser, you will see an empty page.

You can add webfonts, meta tags, or analytics to this file.

The build step will place the bundled scripts into the <body> tag.

To begin the development, run `npm start` or `yarn start`.

To create a production bundle, use `npm run build` or `yarn build`.

-->

</body>

</html>

* **Backend**
* **Admin.py**

from django.contrib import admin

from .models import Data,FeedBack

# Register your models here.

class FeedBackAdmin(admin.ModelAdmin):

list\_display = ( "FirstName",

"LastName",

"City",

"Phone",

"Email",

"FeedBack")

admin.site.register(Data)

admin.site.register(FeedBack,FeedBackAdmin)

* **Apps.py**

from django.apps import AppConfig

class RecommendConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'recommend'

* **Models.py**

from django.db import models

# Create your models here.

class Data(models.Model):

N = models.IntegerField(max\_length=3,null=False)

P = models.IntegerField(max\_length=3,null=False)

K = models.IntegerField(max\_length=3,null=False)

temperature = models.FloatField(max\_length=4 , null=False)

humidity = models.FloatField(max\_length=4 , null=False)

ph = models.FloatField(max\_length=4 , null=False)

rainfall = models.FloatField(max\_length=4 , null=False)

result = models.CharField(max\_length=255)

ml= models.CharField(max\_length=255,null=False)

def \_\_str\_\_(self):

return f"{self.result}"

class FeedBack(models.Model):

FirstName= models.CharField(max\_length=255,null=False)

LastName=models.CharField(max\_length=255,null=False)

City=models.CharField(max\_length=255,null=False)

Phone=models.IntegerField(max\_length=10)

Email=models.EmailField(null=False)

FeedBack=models.CharField(max\_length=255,null=False)

def \_\_str\_\_(self):

return f"{self.FirstName} {self.LastName}"

* **Serializer.py**

from rest\_framework import serializers

from .models import Data,FeedBack

class DataSerializer(serializers.ModelSerializer):

class Meta:

model=Data

fields=('id','N','P','K','temperature','humidity','ph','rainfall', 'result','ml')

class CreateDataSerializer(serializers.ModelSerializer):

class Meta:

model=Data

fields=('id','N','P','K','temperature','humidity','ph','rainfall', 'result','ml')

class CreateFeedBackSerializer(serializers.ModelSerializer):

class Meta:

model=FeedBack

fields=('id','FirstName','LastName','city','phoneno','Email','Feedback')

* **Urls.py**

from django.urls import path

from .views import DataView,CreateDataView,CreateFeedBackView

urlpatterns = [

path('history',DataView.as\_view()),

path('recommend',CreateDataView.as\_view()),

path('feedback',CreateFeedBackView.as\_view())

]

* **Views.py**

from django.http import JsonResponse

from rest\_framework import generics,status

from .models import Data,FeedBack

from .serializer import DataSerializer,CreateDataSerializer,CreateFeedBackSerializer

from rest\_framework.response import Response

from rest\_framework.views import APIView

import pickle

import sklearn

import numpy as np

import pandas as pd

# Create your views here.

class DataView(generics.ListAPIView):

queryset = Data.objects.all()

serializer\_class = DataSerializer

class CreateDataView(APIView):

serializer=CreateDataSerializer

def post(self,request):

infile = open('encoder', 'rb')

encoder = pickle.load(infile)

infile.close()

infile = open('accuracy', 'rb')

accuracy = pickle.load(infile)

infile.close()

infile = open('classifier', 'rb')

classifier = pickle.load(infile)

infile.close()

request.data['N']=int(request.data['N'])

request.data['P'] = int(request.data['P'])

request.data['K'] = int(request.data['K'])

request.data['temp'] = float(request.data['temp'])

request.data['Ph'] = float(request.data['Ph'])

request.data['humidity'] = float(request.data['humidity'])

request.data['rain'] = float(request.data['rain'])

print(classifier)

pred=classifier[ request.data['classifier']]

print(pred)

data = [request.data['N'],request.data['P'],request.data['K'],request.data['temp'],request.data['Ph'],request.data['humidity'],request.data['rain']]

input = np.array(data)

input = input.reshape(1, 7)

feature\_name = ["N", "P", "K", "temperature", "humidity", "ph", "rainfall"]

c = pd.DataFrame(input, columns=feature\_name)

from sklearn.preprocessing import LabelEncoder

file=pd.read\_csv('Crop\_recommendation.csv')

label=LabelEncoder()

label.fit\_transform(file['label'])

file['label']=label.fit\_transform(file['label'])

print(encoder.inverse\_transform(pred.predict(c)))

print(label.inverse\_transform(pred.predict(c)))

ans = label.inverse\_transform(pred.predict(c))

request.data['result'] = ans

model={'rf':"RandomForestCassifier",

'knn':"KNNClassifier",

'dt':"DecissionTreeClassifier"}

data=Data(N=request.data['N'],P=request.data['P'],K=request.data['K'],temperature=request.data['temp'],humidity=request.data['humidity'],ph=request.data['Ph'],rainfall=request.data['rain'],result=ans,ml=model[request.data['classifier']])

data.save()

d=[accuracy[request.data['classifier']],str(ans),model[request.data['classifier']]]

return JsonResponse(d,safe=False)

class CreateFeedBackView(APIView):

def post(self,request):

model=FeedBack(FirstName=request.data['first'], LastName=request.data['last'] ,City= request.data['city'], Phone=request.data['phone'],Email= request.data['email'], FeedBack=request.data['feedback'])

model.save()

return JsonResponse({'as':2})

* **Settings.py**

"""

Django settings for crop project.

Generated by 'django-admin startproject' using Django 4.2.1.

For more information on this file, see

https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/4.2/ref/settings/

"""

from pathlib import Path

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = 'django-insecure-5y!\_n%!sc3!&)nf1nknq+m==qzrg%1!rzq21qg#dofu@z-&jq6'

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'recommend.apps.RecommendConfig',

'rest\_framework',

'corsheaders',

]

MIDDLEWARE = [

''

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

"corsheaders.middleware.CorsMiddleware",

"django.middleware.common.CommonMiddleware",

]

CORS\_ALLOW\_ALL\_ORIGINS =True

ROOT\_URLCONF = 'crop.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'crop.wsgi.application'

# Database

# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

# Password validation

# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/4.2/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC\_URL = 'static/'

# Default primary key field type

# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

* **\_Init\_.py**

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('recommend.urls'))

]

* **Model.py**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

#Read the data set

from google.colab import files

uploaded = files.upload()

data = pd.read\_csv("Crop\_recommendation.csv")

data

data.shape

data.dtypes

data.head()

data.info()

data.describe()

data.isnull().sum()

# Getting numerical data to seperate columns

df\_num = data.select\_dtypes(include=[np.number])

df\_num

df = data['label'].unique()

print(len(df))

print(df)

print(pd.value\_counts(data['label']))

from scipy.stats import skew

for col in df\_num:

print(skew(df\_num[col]))

plt.figure()

sns.distplot(df\_num[col])

plt.show()

#checking the outliers in column N using boxplot

sns.boxplot(data['N'])

#checking the outliers in column P using boxplot

sns.boxplot(data['P'])

#calculating the quantiles of P

q3=data['P'].quantile(0.75)

q1=data['P'].quantile(0.25)

q2=data['P'].quantile(0.50)

IQR = q3-q1

#calculating the upperwhisker and lowerwhisker

ub = q3+1.5\*IQR

print(ub)

lb = q1-1.5\*IQR

print(lb)

data[(data.P<lb) | (data.P>ub)]

data['P'] = data['P'][(data['P'] >= lb) & (data['P'] <= ub)]

sns.boxplot(data['P'])

#checking the outliers in column K using boxplot

sns.boxplot(data['K'])

#calculating the quantiles of K

q3=data['K'].quantile(0.75)

q1=data['K'].quantile(0.25)

q2=data['K'].quantile(0.50)

IQR = q3-q1

#calculating the upperwhisker and lowerwhisker

ub = q3+1.5\*IQR

print(ub)

lb = q1-1.5\*IQR

print(lb)

data[(data.K<lb) | (data.K>ub)]

data['K'] = data['K'][(data['K'] >= lb) & (data['K'] <= ub)]

sns.boxplot(data['K'])

#checking the outliers in column temperature using boxplot

sns.boxplot(data['temperature'])

#calculating the quantiles of temperature

q3=data['temperature'].quantile(0.75)

q1=data['temperature'].quantile(0.25)

q2=data['temperature'].quantile(0.50)

IQR = q3-q1

#calculating the upperwhisker and lowerwhisker

ub = q3+1.5\*IQR

print(ub)

lb = q1-1.5\*IQR

print(lb)

data[(data.temperature<lb) | (data.temperature>ub)]

data['temperature'] = data['temperature'][(data['temperature'] >= lb) & (data['temperature'] <= ub)]

sns.boxplot(data['temperature'])

#checking the outliers in column humidity using boxplot

sns.boxplot(data['humidity'])

#calculating the quantiles of humidity

q3=data['humidity'].quantile(0.75)

q1=data['humidity'].quantile(0.25)

q2=data['humidity'].quantile(0.50)

IQR = q3-q1

#calculating the upperwhisker and lowerwhisker

ub = q3+1.5\*IQR

print(ub)

lb = q1-1.5\*IQR

print(lb)

data[(data.humidity<lb) | (data.humidity>ub)]

data['humidity'] = data['humidity'][(data['humidity'] >= lb) & (data['humidity'] <= ub)]

sns.boxplot(data['humidity'])

#checking the outliers in column ph using boxplot

sns.boxplot(data['ph'])

#calculating the quantiles of ph

q3=data['ph'].quantile(0.75)

q1=data['ph'].quantile(0.25)

q2=data['ph'].quantile(0.50)

IQR = q3-q1

#calculating the upperwhisker and lowerwhisker

ub = q3+1.5\*IQR

print(ub)

lb = q1-1.5\*IQR

print(lb)

data[(data.ph<lb) | (data.ph>ub)]

data['ph'] = data['ph'][(data['ph'] >= lb) & (data['ph'] <= ub)]

sns.boxplot(data['ph'])

#checking the outliers in column rainfall using boxplot

sns.boxplot(data['rainfall'])

#calculating the quantiles of rainfall

q3=data['rainfall'].quantile(0.75)

q1=data['rainfall'].quantile(0.25)

q2=data['rainfall'].quantile(0.50)

IQR = q3-q1

#calculating the upperwhisker and lowerwhisker

ub = q3+1.5\*IQR

print(ub)

lb = q1-1.5\*IQR

print(lb)

data[(data.rainfall<lb) | (data.rainfall>ub)]

data['rainfall'] = data['rainfall'][(data['rainfall'] >= lb) & (data['rainfall'] <= ub)]

sns.boxplot(data['rainfall'])

data.isnull().sum()

#replacing missing values of P with the median of that column.

data['P']=data['P'].fillna(data['P'].median())

#replacing missing values of K with the median of that column.

data['K']=data['K'].fillna(data['K'].median())

#replacing missing values of temperature with the median of that column.

data['temperature']=data['temperature'].fillna(data['temperature'].median())

#replacing missing values of humidity with the median of that column.

data['humidity']=data['humidity'].fillna(data['humidity'].median())

#replacing missing values of ph with the median of that column.

data['ph']=data['ph'].fillna(data['ph'].median())

#replacing missing values of rainfall with the median of that column.

data['rainfall']=data['rainfall'].fillna(data['rainfall'].median())

data.isnull().sum()

#here performing labelEncoder using LabelEncoder to convert categorical variables into

numerical variables.

from sklearn.preprocessing import LabelEncoder

import pickle

labelEncoder= LabelEncoder()

data['label']=labelEncoder.fit\_transform(data['label'].values)

filename = 'encoder'

outfile = open(filename,'wb')

pickle.dump(labelEncoder,outfile)

outfile.close()

data.info()

# Model building

from sklearn.model\_selection import train\_test\_split

x = data.drop('label', axis = 1)

y = data['label']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.33, random\_state = 1)

x\_test.shape

x\_train.shape

y\_test.shape

y\_train.shape

from sklearn.ensemble import RandomForestClassifier #import the regressor

from sklearn import metrics #imporing metrics from sklearn library

from sklearn.metrics import r2\_score #importing r2 from sklearn library

regressor = RandomForestClassifier(n\_estimators = 100,random\_state=0) #create regressor object

regressor.fit(x\_train,y\_train)

y\_pred=regressor.predict(x\_train) #predication of y\_train

print('r2 score for Training :',r2\_score(y\_train,y\_pred))

print('mean absolute Error for Training:' , metrics.mean\_absolute\_error(y\_train,y\_pred))

print('mean Squared Error for Training:' , metrics.mean\_absolute\_error(y\_train,y\_pred))

print(' Root mean Squared Error for Training:', np.sqrt(metrics.mean\_absolute\_error(y\_train,y\_pred)))

print()

y\_pred=regressor.predict(x\_test)

print('r2 score for Training :',r2\_score(y\_test,y\_pred))

print('r2 score for Testing :',(y\_test,y\_pred)) #print of r2 score for test

print('mean absolute Error for Testing:',metrics.mean\_absolute\_error(y\_test,y\_pred)) #print Mse value for test

print('mean Squared Error for Testing:',metrics.mean\_absolute\_error(y\_test,y\_pred))#print Mse value for test

print(' Root mean Squared Error for Testing:',np.sqrt(metrics.mean\_absolute\_error(y\_test,y\_pred)))

# Cross Validation

from sklearn.model\_selection import ShuffleSplit

model= RandomForestClassifier()

ssplit=ShuffleSplit(n\_splits=10,test\_size=0.20)

from sklearn.model\_selection import cross\_val\_score

results=cross\_val\_score(model,x,y,cv=ssplit)

print(results\*100)

print(np.mean(results)\*100)

RF\_ACC=np.mean(results\*100)

# Decision TreeClassifier

from sklearn.tree import DecisionTreeClassifier #import the regressor

from sklearn import metrics #imporing metrics from sklearn library

from sklearn.metrics import r2\_score #importing r2 from sklearn library

classifier = DecisionTreeClassifier(max\_depth=3)

classifier.fit(x\_train,y\_train)

print(classifier.score(x\_train,y\_train))

print(classifier.score(x\_test,y\_test))

y\_pred=classifier.predict(x\_train) #predication of y\_train

print('r2 score for Training :',r2\_score(y\_train,y\_pred))

print('mean absolute Error for Training:' , metrics.mean\_absolute\_error(y\_train,y\_pred))

print('mean Squared Error for Training:' , metrics.mean\_absolute\_error(y\_train,y\_pred))

print(' Root mean Squared Error for Training:', np.sqrt(metrics.mean\_absolute\_error(y\_train,y\_pred)))

print()

y\_pred=classifier.predict(x\_test)

print('r2 score for Training :',r2\_score(y\_test,y\_pred))

print('r2 score for Testing :',(y\_test,y\_pred)) #print of r2 score for test

print('mean absolute Error for Testing:',metrics.mean\_absolute\_error(y\_test,y\_pred)) #print Mse value for test

print('mean Squared Error for Testing:',metrics.mean\_absolute\_error(y\_test,y\_pred))#print Mse value for test

print(' Root mean Squared Error for Testing:',np.sqrt(metrics.mean\_absolute\_error(y\_test,y\_pred)))

from sklearn import tree

plt.figure(figsize=(15,5))

tree.plot\_tree(classifier,feature\_names=x\_train.columns,max\_depth=3,filled=True)

plt.show()

#Cross Validation

* **Data set sample**

N P K temperature humidity ph rainfall label

90 42 43 20.87974371 82.00274423 6.502985292 202.9355362 rice

85 58 41 21.77046169 80.31964408 7.038096361 226.6555374 rice

60 55 44 23.00445915 82.3207629 7.840207144 263.9642476 rice

74 35 40 26.49109635 80.15836264 6.980400905 242.8640342 rice

78 42 42 20.13017482 81.60487287 7.628472891 262.7173405 rice

69 37 42 23.05804872 83.37011772 7.073453503 251.0549998 rice

69 55 38 22.70883798 82.63941394 5.70080568 271.3248604 rice

94 53 40 20.27774362 82.89408619 5.718627178 241.9741949 rice

89 54 38 24.51588066 83.5352163 6.685346424 230.4462359 rice

68 58 38 23.22397386 83.03322691 6.336253525 221.2091958 rice

91 53 40 26.52723513 81.41753846 5.386167788 264.6148697 rice

90 46 42 23.97898217 81.45061596 7.50283396 250.0832336 rice

78 58 44 26.80079604 80.88684822 5.108681786 284.4364567 rice

93 56 36 24.01497622 82.05687182 6.98435366 185.2773389 rice

94 50 37 25.66585205 80.66385045 6.94801983 209.5869708 rice

60 48 39 24.28209415 80.30025587 7.042299069 231.0863347 rice

85 38 41 21.58711777 82.7883708 6.249050656 276.6552459 rice

91 35 39 23.79391957 80.41817957 6.970859754 206.2611855 rice

77 38 36 21.8652524 80.1923008 5.953933276 224.5550169 rice

88 35 40 23.57943626 83.58760316 5.85393208 291.2986618 rice

89 45 36 21.32504158 80.47476396 6.442475375 185.4974732 rice

76 40 43 25.15745531 83.11713476 5.070175667 231.3843163 rice

67 59 41 21.94766735 80.97384195 6.012632591 213.3560921 rice

83 41 43 21.0525355 82.67839517 6.254028451 233.1075816 rice

98 47 37 23.48381344 81.33265073 7.375482851 224.0581164 rice

66 53 41 25.0756354 80.52389148 7.778915154 257.0038865 rice

97 59 43 26.35927159 84.04403589 6.286500176 271.3586137 rice

97 50 41 24.52922681 80.54498576 7.070959995 260.2634026 rice

60 49 44 20.77576147 84.49774397 6.244841491 240.0810647 rice

84 51 35 22.30157427 80.64416466 6.043304899 197.9791215 rice

73 57 41 21.44653958 84.94375962 5.824709117 272.2017204 rice

92 35 40 22.17931888 80.33127223 6.357389366 200.0882787 rice

85 37 39 24.52783742 82.73685569 6.364134968 224.6757231 rice

98 53 38 20.26707606 81.63895217 5.01450727 270.4417274 rice

88 54 44 25.7354293 83.88266234 6.149410611 233.1321372 rice

95 55 42 26.79533926 82.1480873 5.950660556 193.3473987 rice

99 57 35 26.75754171 81.17734011 5.960370061 272.2999056 rice

95 39 36 23.86330467 83.15250801 5.561398642 285.2493645 rice

60 43 44 21.01944696 82.95221726 7.416245107 298.4018471 rice

63 44 41 24.17298839 83.7287574 5.583370042 257.0343554 rice

62 42 36 22.78133816 82.06719137 6.430010215 248.7183228 rice

64 45 43 25.62980105 83.52842314 5.534878156 209.9001977 rice

83 60 36 25.59704938 80.14509262 6.903985986 200.834898 rice

82 40 40 23.83067496 84.81360127 6.271478838 298.5601175 rice

85 52 45 26.31355498 82.36698992 7.224285503 265.5355937 rice

91 35 38 24.8972823 80.52586088 6.13428721 183.6793207 rice

76 49 42 24.958779 84.47963372 5.206373153 196.9560008 rice

74 39 38 23.24113501 84.59201843 7.782051313 233.0453455 rice

79 43 39 21.66628296 80.70960551 7.062779015 210.8142087 rice

88 55 45 24.63544858 80.41363018 7.730367824 253.7202781 rice

60 36 43 23.43121862 83.06310136 5.286203711 219.9048349 rice

76 60 39 20.0454142 80.3477562 6.766240045 208.5810155 rice

93 56 42 23.85724032 82.22572988 7.382762603 195.0948311 rice

65 60 43 21.97199397 81.89918197 5.658169482 227.3637009 rice

95 52 36 26.22916897 83.83625819 5.543360238 286.5083725 rice

75 38 39 23.44676801 84.79352417 6.215109715 283.9338466 rice

74 54 38 25.65553461 83.47021081 7.120272972 217.3788583 rice

91 36 45 24.44345477 82.45432595 5.950647577 267.9761948 rice

71 46 40 20.2801937 82.1235421 7.236705436 191.9535738 rice

99 55 35 21.7238313 80.2389895 6.501697816 277.9626192 rice

72 40 38 20.41447029 82.20802629 7.592490617 245.1511304 rice

83 58 45 25.75528612 83.51827127 5.875345751 245.6626799 rice

93 58 38 20.61521424 83.77345559 6.932400225 279.5451717 rice

70 36 42 21.84106875 80.72886384 6.946209881 202.3838319 rice

76 47 42 20.08369642 83.29114712 5.739175027 263.6372176 rice

99 41 36 24.45802087 82.74835604 6.738652179 182.5616319 rice

99 54 37 21.14347496 80.33502926 5.594819626 198.6730942 rice

86 59 35 25.78720567 82.11124033 6.946636369 243.5120414 rice

69 46 41 23.64124821 80.28597873 5.012139669 263.1103304 rice

91 56 37 23.43191632 80.56887849 6.363472208 269.5039162 rice

61 52 41 24.97669518 83.891805 6.880431223 204.8001847 rice

67 45 38 22.72791041 82.1706881 7.300410836 260.8875056 rice

79 42 37 24.87300744 82.84022551 6.587918708 295.6094492 rice

78 43 42 21.32376327 83.00320459 7.283736617 192.3197536 rice

75 54 36 26.29465461 84.56919326 7.023936392 257.4914906 rice

97 36 45 22.2286982 81.85872947 6.939083505 278.0791793 rice

67 47 44 26.73072391 81.78596776 7.868474653 280.4044392 rice

73 35 38 24.88921174 81.97927117 5.005306977 185.9461429 rice

77 36 37 26.88444878 81.46033732 6.136131869 194.5766559 rice

81 41 38 22.67846116 83.72874389 7.524080076 200.9133156 rice

68 57 43 26.08867875 80.37979919 5.706943251 182.9043504 rice

72 45 35 25.42977518 82.94682591 5.758506323 195.3574542 rice

61 53 43 26.40323239 81.05635517 6.349606327 223.3671883 rice

67 43 39 26.04371967 84.96907151 5.999969026 186.7536773 rice

67 58 39 25.2827223 80.54372813 5.453592032 220.1156708 rice

66 60 38 22.08576562 83.47038318 6.372576327 231.7364957 rice

82 43 38 23.28617173 81.43321641 5.105588355 242.3170629 rice

84 50 44 25.48591986 81.40633547 5.935344406 182.6549356 rice

81 53 42 23.67575393 81.03569343 5.17782304 233.7034975 rice

91 50 40 20.82477109 84.1341879 6.462391607 230.2242223 rice

93 53 38 26.92995077 81.91411159 7.069172227 290.6793783 rice

90 44 38 23.83509503 83.88387074 7.473134377 241.2013513 rice

81 45 35 26.52872817 80.12267476 6.158376967 218.9163567 rice

78 40 38 26.46428311 83.85642678 7.549873681 248.2256491 rice

60 51 36 22.69657794 82.81088865 6.028321558 256.9964761 rice

88 46 42 22.68319059 83.46358271 6.604993475 194.2651719 rice

93 47 37 21.53346343 82.14004101 6.500343222 295.9248796 rice

60 55 45 21.40865769 83.3293191 5.935745417 287.5766935 rice

78 35 44 26.54348085 84.67353597 7.072655622 183.6222657 rice

65 37 40 23.35905428 83.59512273 5.333322606 188.413665 rice

71 54 16 22.61359953 63.69070564 5.749914421 87.75953857 maize

61 44 17 26.10018422 71.57476937 6.931756558 102.2662445 maize

80 43 16 23.55882094 71.59351368 6.657964753 66.71995467 maize

73 58 21 19.97215954 57.68272924 6.596060648 60.65171481 maize

61 38 20 18.47891261 62.69503871 5.970458434 65.43835393 maize

68 41 16 21.77689322 57.80840636 6.158830619 102.0861694 maize

93 41 17 25.6217169 66.50415474 6.047906679 105.4654703 maize

89 60 19 25.19192419 66.6902901 5.913664501 78.06639649 maize

76 44 17 20.41683147 62.5542482 5.855442401 65.27798457 maize

67 60 25 24.92162194 66.78627406 5.750254943 109.2162279 maize

70 44 19 23.31689124 73.4541537 5.852607099 94.29712821 maize

90 49 21 24.84016732 68.3584573 6.472523287 74.05474936 maize

62 52 16 22.27526694 58.84015925 6.967057762 63.87020584 maize

92 44 16 18.87751445 65.76816093 6.082973754 94.76189431 maize

66 54 21 25.19008683 60.2001687 5.919045532 72.12375573 maize

63 58 22 18.25405352 55.28220433 6.204747653 63.72358154 maize

70 47 17 24.6129118 70.4162444 6.600827017 104.1626147 maize

61 41 17 25.1420613 65.26185135 6.021902237 76.68456006 maize

66 53 19 23.09348056 60.1159381 6.033550195 65.49730729 maize

74 55 19 18.05033737 62.89366992 6.28886807 84.23613484 maize

77 57 21 24.9321581 73.80435276 6.550563823 79.74078719 maize

99 50 15 18.14710054 71.09445342 5.573286437 88.07753741 maize

74 56 22 18.28362235 66.65952796 6.829199275 80.97573281 maize

83 45 21 18.83344471 58.75082029 5.716222912 79.7532896 maize

100 48 16 25.71895816 67.22190688 5.54990242 74.51490791 maize

79 51 16 25.33797709 68.49835977 6.586244581 96.46380213 maize

94 39 18 23.89114571 57.48775781 5.893093135 102.8301942 maize

75 49 15 21.53574127 71.50905983 5.918263801 102.4852929 maize

78 48 22 23.08974909 63.10459626 5.588650585 70.43473609 maize

87 54 20 25.61707368 63.4711755 6.576418207 108.8303762 maize

87 35 25 21.44526922 63.1621551 6.178056304 65.88951188 maize

63 43 19 18.51816776 55.53128131 6.641906353 90.988051 maize

84 57 25 22.53510514 67.99257471 6.489040367 64.40866039 maize

64 35 23 23.02038334 61.89472002 5.680361038 63.03843397 maize

60 46 22 24.89364635 65.61418761 6.625404348 87.9298085 maize

98 44 21 25.77175115 74.089114 6.524478032 107.4931917 maize

75 56 18 19.39851734 62.35750641 5.696205468 60.95197486 maize

86 55 21 21.54156232 59.64024162 6.803931519 109.7515385 maize

98 35 18 23.79746068 74.82913698 6.252797548 91.76337172 maize

76 57 18 18.9802729 74.52600826 6.092725883 94.26249353 maize

99 56 17 24.10859207 73.13112261 6.234330356 71.07562236 maize

60 44 23 24.7947077 70.04556743 5.722579819 76.72860067 maize

74 48 17 21.63162756 60.27766379 6.430616465 69.21803098 maize

89 60 17 25.37548751 57.21025565 5.983952675 101.7004306 maize

69 51 23 22.21738222 72.85462807 6.80163854 106.6213157 maize

96 46 22 20.58314011 69.00128641 6.499936446 66.29390357 maize

61 60 15 24.87502824 68.74248334 6.265564338 91.26056654 maize

74 58 18 20.03728219 56.35606753 6.727303282 109.024141 maize

74 43 23 25.95263264 61.89082199 6.325235159 99.57981207 maize

63 43 17 19.28889933 65.47050802 6.807487794 71.3195307 maize

99 36 20 20.57981887 65.34583901 6.671085817 78.34604471 maize

77 36 23 24.71417533 56.73426469 6.648725327 88.45361858 maize

87 60 23 20.27317074 63.91281869 6.439071996 62.50351892 maize

60 38 17 18.41932981 64.23580251 6.474476516 76.41312437 maize

94 54 17 23.39128187 61.74427165 5.871647806 107.3198135 maize

95 38 22 19.84939404 61.24500053 5.730617109 100.7689246 maize

84 44 21 21.869274 61.91044947 5.850439831 107.2681929 maize

77 58 19 22.8056033 56.50768935 5.791649933 101.5952794 maize

66 44 20 19.0781471 69.02298571 6.740000688 80.72515943 maize

63 35 16 22.02720976 65.35549924 6.272417541 83.73280082 maize

79 45 20 23.80546189 59.24537979 5.715208817 89.9622014 maize

72 60 25 18.52510753 69.0276233 5.773454729 88.10234397 maize

67 51 24 23.50297882 61.32026065 5.584171461 64.77791424 maize

86 36 24 26.54986394 72.89187265 5.787268394 73.33636055 maize

76 48 18 19.29563411 69.63481219 5.77597783 83.21030571 maize

75 53 18 20.68899915 59.4375337 6.864793607 103.651438 maize

81 45 23 19.32666088 68.034493 6.192360003 84.22969177 maize

73 45 21 24.60532218 73.58868502 6.636803223 96.59195302 maize

71 35 24 22.27373646 59.52193158 5.826426917 67.96704792 maize

96 54 22 25.70196694 61.33450447 6.960358276 83.20711308 maize

99 39 18 19.20129357 68.30578978 6.11275104 87.85092352 maize

62 48 20 21.70181447 60.47470519 6.708446922 95.71388473 maize

86 37 16 20.51716779 59.21235483 5.561510732 67.61013737 maize

94 50 19 23.30355338 73.62548442 5.873242491 97.59081274 maize

76 39 24 24.2547451 55.64709899 6.995843776 64.23845455 maize

77 52 17 24.86374934 65.7420046 5.714799723 75.82270467 maize

74 39 23 22.6265115 65.77472881 6.78073637 88.17251033 maize

81 49 20 18.04185513 60.61494304 5.513697923 104.2321615 maize

63 42 21 23.26237612 72.33125523 5.798423908 67.10225139 maize

99 38 21 22.88330922 71.59722446 6.352471866 67.72777298 maize

90 52 25 25.97482359 69.36385721 6.822586546 103.2234212 maize

68 40 19 26.14384005 66.20569924 6.655426355 107.2361366 maize

60 57 24 18.66116213 61.55327249 6.121294041 75.03247667 maize

71 52 18 25.10787449 55.97732754 5.790770203 78.16077693 maize

61 59 17 23.33844615 59.24580604 6.47444292 105.0083144 maize

88 38 15 25.08239719 65.92195844 6.455116637 62.49190812 maize

65 60 22 25.36768364 72.52054555 6.606984086 107.9124111 maize

78 37 22 25.34217103 63.31801994 6.330554389 74.52082026 maize

78 58 15 25.00933355 67.816568 6.528631266 62.91359494 maize

92 60 23 18.66746724 71.516474 5.721667141 69.93293255 maize

79 59 17 20.37999665 63.73849998 6.644205485 108.5054416 maize

91 55 15 18.09300227 72.61024172 6.376651091 78.96159541 maize

76 51 18 26.16985907 71.96246617 6.247040422 79.84925393 maize

87 48 25 18.65396672 61.37879671 6.656730008 93.62039175 maize

71 60 22 26.07470121 59.37147589 6.2048017 85.75692395 maize

90 57 24 18.92851916 72.80086137 6.158860284 82.34162918 maize

67 35 22 23.30546753 63.24648023 6.385684214 108.7603001 maize

60 54 19 18.74826712 62.49878458 6.417820493 70.23401597 maize

83 58 23 19.74213321 59.66263104 6.381201909 65.50861389 maize

83 57 19 25.73044432 70.74739256 6.877869005 98.73771338 maize

32 76 15 28.05153602 63.49802189 7.604110177 43.35795377 lentil

13 61 22 19.44084326 63.27771461 7.728832424 46.83130119 lentil

38 60 20 29.84823072 60.63872613 7.491217102 46.80452595 lentil

11 74 17 21.36383757 69.92375891 6.633864582 46.6352865 lentil

37 71 16 26.28663931 68.51966729 7.324863481 46.13833007 lentil

29 71 18 22.17499963 62.13873825 6.410441476 53.46622584 lentil

2 72 18 26.57597546 60.97876599 7.836719564 50.89110726 lentil

6 59 21 26.58972517 66.14007674 6.139215944 50.90994463 lentil

13 64 20 19.1345771 62.57526895 6.590571088 36.46946971 lentil

8 58 17 28.75273118 69.15640149 7.286049978 35.15426171 lentil

6 77 20 25.78746268 60.2816298 6.058306161 49.14337177 lentil

2 75 22 23.89271875 61.78779413 6.658605362 52.55730112 lentil

3 69 23 28.67408774 63.18832976 7.299360767 42.96018627 lentil

27 80 24 28.42062847 61.77336343 7.815210661 49.02366803 lentil

39 78 15 21.35499456 62.60136323 5.925391795 41.78219834 lentil

40 79 17 21.12695586 63.18738532 6.403683619 38.71834464 lentil

37 62 22 24.02037872 61.62313345 7.397546271 49.78102578 lentil

31 60 24 25.40474421 65.8567539 7.722335992 51.92057267 lentil

22 67 22 29.03017561 64.49166566 7.475926645 54.9393771 lentil

3 78 18 20.21368219 68.65257685 6.887130053 50.89732989 lentil

4 80 16 29.19585548 68.01965728 7.441976825 44.93261911 lentil

13 61 24 18.29783597 69.6897615 7.629910253 49.39111479 lentil

12 66 20 27.41434987 63.41785982 7.336117221 44.43177543 lentil

4 61 21 24.84063998 60.09116626 6.75020529 48.77790371 lentil

9 60 21 29.94413861 67.31323084 7.52178027 40.37113729 lentil

18 66 22 25.87990287 67.55109024 6.347379185 47.89645224 lentil

32 56 18 20.0467711 65.84395319 7.135251532 46.05333124 lentil

6 72 15 22.99451999 66.70897237 7.670178119 54.49044154 lentil

15 77 20 25.13163619 66.92642362 7.399749291 49.04015558 lentil

0 65 24 28.49584395 62.44616219 7.841496029 53.14531023 lentil

30 79 22 18.28766124 69.48515056 6.254216611 48.60449438 lentil

3 63 16 24.38041875 61.18458224 6.868881708 53.13946695 lentil

2 78 23 21.31852148 66.43934593 7.320514721 45.42616802 lentil

10 78 18 18.54141834 62.70637578 6.296976913 44.07819743 lentil

14 67 25 25.28710601 60.85993533 7.241151936 49.37369982 lentil

39 65 23 25.43459777 69.12613376 7.685959305 41.02682925 lentil

19 72 15 28.83600962 69.76112921 6.890760124 44.08562546 lentil

18 57 21 27.37659643 63.93927841 6.155915975 49.47371773 lentil

31 58 15 28.31886863 60.19461399 6.167855382 45.36521251 lentil

28 58 25 27.4818649 62.04814951 6.861640036 37.81123974 lentil

5 65 19 18.28072173 68.10365387 6.978361689 48.80253285 lentil

16 65 19 27.61204997 69.29786244 7.043160241 42.72374404 lentil

34 65 19 23.43974653 63.22011726 5.94239222 45.40277297 lentil

14 69 19 20.95628486 63.68128841 7.239455147 52.39881209 lentil

22 55 16 23.7937153 68.03209183 6.516317561 49.73922097 lentil

24 61 17 22.6371424 65.44544859 6.233269045 38.30411077 lentil

2 79 15 21.53577883 65.47227704 7.505283615 35.75107592 lentil

26 63 17 29.87854588 65.73085206 6.950300686 44.95654782 lentil

27 61 15 25.2653291 67.10004577 6.958054839 48.33941188 lentil

24 70 16 25.17885316 68.93307305 6.54803469 35.03484812 lentil

13 74 25 24.12192608 61.09533545 6.461618577 44.23629285 lentil

6 64 23 23.33565221 67.40460704 7.065264073 36.18678721 lentil

12 58 23 21.74600081 63.39503184 6.765091462 50.43306085 lentil

32 79 22 27.60195453 63.46170674 5.91645379 54.37814199 lentil

6 68 18 24.388717 62.50453062 6.711341147 47.26052494 lentil

10 79 20 24.98287462 66.895409 6.379881442 38.21370568 lentil

38 77 22 28.234829 69.3159965 6.313284268 35.36831423 lentil

17 74 17 26.03026959 69.55863145 7.393210848 37.11395801 lentil

26 68 24 28.04849594 64.07691942 7.504930973 37.15824966 lentil

23 75 17 24.87425505 64.00213929 7.198076286 48.28137482 lentil

32 78 22 23.97081395 62.35557553 7.007037515 53.40906048 lentil

19 79 19 20.06003985 67.76252583 6.677262562 42.89509057 lentil

22 60 18 19.59221047 61.28633405 6.74398035 41.7704893 lentil

28 69 16 29.77013109 66.29327012 6.547361618 35.69674138 lentil

1 67 21 27.52135365 60.53657684 6.551577598 48.06491307 lentil

12 67 23 25.62896213 63.14909763 6.585020303 45.49683991 lentil

36 67 20 20.39078312 60.47528931 6.924042372 53.31508572 lentil

28 70 21 25.39038396 60.4989659 7.437373666 39.18374505 lentil

12 71 19 24.91079638 60.71367427 7.142611056 42.19740397 lentil

22 68 16 27.70496805 63.20915034 7.74672376 37.46160727 lentil

26 66 22 18.06486101 65.1034354 6.300479414 51.54922825 lentil

16 65 16 18.13027797 62.45851612 6.078724107 50.6128521 lentil

14 59 22 23.82723528 67.89815262 6.76660668 46.90725077 lentil

33 59 19 23.19305333 62.74710773 7.641024177 49.55213308 lentil

21 63 17 25.08966129 68.17543102 6.559681838 41.45486619 lentil

0 69 21 25.86928193 61.88321072 7.072923306 36.68284038 lentil

10 75 17 18.43966037 68.05394959 7.732194788 39.00992137 lentil

30 61 18 27.14911056 67.02664337 6.157782589 52.50812701 lentil

0 74 17 23.33375853 64.50515776 7.240988401 47.01510708 lentil

35 74 22 26.7230014 62.96841833 6.898905799 42.87274897 lentil

7 63 24 19.55750776 64.45268309 6.818681086 53.04669416 lentil

9 56 17 26.13708256 66.7729209 6.261937875 46.48280681 lentil

14 74 15 27.99990346 65.57653373 6.493036868 49.94043064 lentil

14 76 20 29.05941162 62.10652364 7.042474679 36.5011366 lentil

36 65 16 25.71269843 64.1123333 7.692013657 50.17067771 lentil

28 67 21 21.79792649 63.73086065 6.250994223 46.62370222 lentil

28 79 16 24.70626432 60.26854183 6.052184881 53.12442925 lentil

40 61 22 20.94981756 65.8108757 7.002216044 44.23913012 lentil

10 70 19 24.84918386 68.98088448 7.272427638 41.61080544 lentil

12 80 19 21.91041045 65.21662467 5.962001484 36.10211371 lentil

37 77 20 25.93381964 68.70533022 7.080506001 51.02372773 lentil

0 67 22 29.82112112 69.4073209 6.593798387 51.56461082 lentil

7 73 25 27.52185591 63.13215259 7.28805662 45.20841071 lentil

10 56 18 27.99627907 68.6428593 7.32710972 46.10585191 lentil

39 70 15 20.76774783 63.90164154 6.366355781 47.9271552 lentil

26 56 22 23.05276444 60.424786 7.011121216 52.60285259 lentil

9 77 17 21.65845777 63.58337146 6.280725549 38.07659414 lentil

4 59 19 26.25070298 67.62779652 7.621494566 40.8106299 lentil

34 73 15 20.97195263 63.83179889 7.630424083 53.10207889 lentil

33 77 15 23.89736406 66.32102048 7.802212437 40.74536757 lentil

2 40 27 29.73770045 47.54885174 5.954626604 90.09586854 mango

39 24 31 33.55695561 53.72979826 4.757114897 98.67527561 mango

21 26 27 27.00315545 47.67525434 5.699586972 95.85118326 mango

25 22 25 33.56150184 45.53556603 5.977413803 95.70525913 mango

0 21 32 35.89855625 54.25964196 6.430139436 92.19721736 mango

20 19 35 34.17719782 50.62161586 6.113935087 98.00687989 mango

19 21 34 30.01592643 53.19212381 5.074272692 97.72843182 mango

18 17 31 31.74592134 45.16127859 5.667507706 93.75441586 mango

11 36 33 35.99009679 52.22780489 5.978634285 95.3713484 mango

30 28 30 31.86641378 52.19331595 5.064613314 98.46768642 mango

18 19 27 27.75518664 52.34605806 4.772385986 94.11213345 mango

23 23 27 34.72413192 51.4271781 5.161148592 97.31258083 mango

37 30 34 27.53907547 53.63549533 6.797779227 99.35408185 mango

11 27 30 27.69637763 48.5622488 6.39474303 89.85646496 mango

12 19 31 27.25373364 52.66319725 5.566704378 91.87312479 mango

3 28 33 30.33723921 48.88704844 5.755049971 94.42850522 mango

37 38 32 31.85744939 45.53106268 5.417340525 91.55845821 mango

26 37 30 35.39986338 49.45962621 6.166173834 97.41054011 mango

14 18 30 29.80747243 52.13797867 5.191265116 95.74606104 mango

40 16 35 34.16438906 54.16482251 4.954739564 98.33351125 mango

4 20 25 28.93270187 47.94053996 5.664587011 99.9834242 mango

36 25 33 27.98392787 53.33018851 5.548584852 99.61465679 mango

30 17 31 31.20478173 54.49960506 6.804437106 94.62954663 mango

28 37 28 32.13409675 50.52559148 6.097869767 98.63333684 mango

38 15 30 28.91862016 48.13974548 5.075504537 97.01331604 mango

12 37 30 31.09779147 47.41196659 4.546466109 90.28624348 mango

38 19 31 34.73823882 49.08864345 5.855119268 90.65022183 mango

8 33 29 29.98080499 49.48613279 6.442393461 91.82271568 mango

15 27 28 33.80398664 46.12866113 4.507523551 90.82549241 mango

34 16 25 30.07202564 50.96040505 6.10729559 92.09609766 mango

11 36 31 27.92063282 51.77965917 6.47544932 100.2585673 mango

33 29 34 31.40948821 49.21729127 6.832979509 92.99739415 mango

12 31 26 35.7877738 51.94190321 5.395275719 100.2160615 mango

12 34 28 33.36140093 45.02236377 6.13526938 98.81596545 mango

5 16 31 35.96054636 48.69677802 4.555688532 98.00644238 mango

1 30 29 28.33333307 51.39586505 6.434197756 91.67241761 mango

16 35 31 32.27652024 50.19368841 5.316875978 95.99487068 mango

35 18 26 31.99490489 50.84881347 5.279388967 97.38741498 mango

4 40 26 27.58258929 48.56916221 6.720041791 95.8445641 mango

9 29 34 29.38471637 45.88744691 5.72742254 100.8124659 mango

2 38 33 32.38697531 53.2328243 4.691396195 90.21633216 mango

26 32 32 30.91471455 49.92963856 6.810186079 90.14047759 mango

34 38 31 35.37775595 45.58110023 6.454045329 97.41586402 mango

5 32 33 32.32362177 52.5896771 5.842763773 93.36718816 mango

31 29 26 28.22373428 47.40519056 5.024124684 97.76832322 mango

34 34 35 27.27433181 47.16808054 6.422710539 95.257992 mango

36 19 32 27.10710832 50.70880979 4.94295037 92.37238878 mango

7 17 26 34.89226666 48.75613373 6.414526606 91.63074547 mango

38 15 27 33.7462686 48.50387598 6.777788126 92.26439205 mango

5 19 25 27.3511056 54.43945147 6.441328044 96.27792547 mango

37 36 26 32.89300162 52.61323969 4.650536197 94.49161372 mango

21 31 32 35.38598705 51.42664176 5.254532213 90.29643888 mango

37 36 27 27.5529736 47.90859131 5.910634533 90.40332704 mango

23 23 30 32.82141065 47.45553843 4.755273631 90.89173106 mango

36 26 26 30.17294105 51.0845903 6.814630246 95.23444287 mango

24 33 35 29.26382931 54.82257868 5.342866119 100.7586226 mango

26 18 30 32.06097197 51.08494181 6.336234624 96.59816497 mango

22 17 26 28.69818144 47.71875722 4.754435025 99.642454 mango

11 34 32 29.14305008 49.40983294 6.831706773 97.55155537 mango

29 35 28 28.3471611 53.53903102 6.967417766 90.40260445 mango

22 28 26 27.67256197 45.41692012 4.947683034 92.84991507 mango

23 24 32 28.1218093 46.16888595 5.630619901 93.30247448 mango

1 35 34 30.79375683 46.69536813 6.27339822 92.21318555 mango

2 24 34 28.89409382 54.80750249 6.472774648 94.76322976 mango

39 37 25 33.33024826 45.61143594 6.953246506 98.28583013 mango

15 36 27 27.78912455 53.96886679 5.643710216 91.01152997 mango

3 18 31 31.65333432 48.20662669 6.392313973 91.09745581 mango

8 38 32 29.75150773 46.73723302 4.981816523 91.405983 mango

33 31 34 31.32995611 50.22287593 5.421265283 89.78216168 mango

14 29 32 35.63627319 48.97047762 6.942520105 97.51952041 mango

18 20 26 31.66524687 51.98594645 5.435840509 89.98024312 mango

9 21 32 32.26935342 53.56092806 5.870116071 95.94035356 mango

20 30 27 27.81005614 51.59445462 4.74910393 95.89898581 mango

9 38 25 34.58561471 50.34035336 5.497946899 100.3060719 mango

26 24 34 31.27180992 52.23810152 6.811291098 89.74409017 mango

31 36 29 33.93679864 52.72170281 6.460542749 97.4611918 mango

14 18 35 31.09154239 47.02058367 4.791146778 91.46664318 mango

40 16 35 31.89356292 49.02450149 6.4841522 89.59371481 mango

28 27 34 32.45465292 50.69693751 6.526654345 95.04871605 mango

0 17 30 35.47478322 47.97230503 6.279133738 97.79072474 mango

1 29 29 27.32961444 49.30347234 6.052026047 93.53197359 mango

2 36 31 30.90225239 49.95955487 5.73171945 91.77522598 mango

12 27 26 29.09382275 45.5661059 5.32307197 96.23520043 mango

7 28 35 30.02086169 46.78393776 4.66910839 96.63721027 mango

0 36 26 34.13072188 51.25786185 5.101206389 96.38808001 mango

26 35 31 33.44619894 53.05980465 5.339556562 98.05089394 mango

27 21 30 35.3915464 52.48823147 5.061081874 91.22881052 mango

22 38 31 31.53356352 53.06009323 5.821106036 98.57025046 mango

22 18 31 30.7645515 47.93791463 5.956027059 90.38503469 mango

28 23 28 30.01821337 50.0983181 5.676032581 96.08745082 mango

7 31 27 31.32863689 47.59319575 6.524114355 94.67344737 mango

29 34 26 33.88004781 54.39416048 6.273953676 89.29147581 mango

8 37 33 28.07802689 54.9640534 6.128167757 97.45373619 mango

39 16 27 35.53845018 52.94641947 4.934964765 91.54560427 mango

40 24 25 28.70595247 50.44030129 5.445008416 95.8946444 mango

19 38 26 31.48451729 48.77926304 4.525722333 93.17221967 mango

21 21 30 27.69819273 51.41593238 5.403908328 100.7720705 mango

22 18 33 30.41235793 52.48100602 6.621623545 93.92375879 mango

31 20 30 32.17752026 54.01352682 6.207495815 91.88766069 mango

18 26 31 32.6112614 47.74916499 5.418475257 91.10190759 mango

24 130 195 29.99677232 81.54156612 6.112305667 67.12534492 grapes

13 144 204 30.7280404 82.42614055 6.092241627 68.38135469 grapes

22 123 205 32.44577836 83.88504863 5.896343436 68.73932528 grapes

36 125 196 37.46566825 80.65968681 6.15526103 66.83872293 grapes

24 131 196 22.03296178 83.74372787 5.732453638 65.34440794 grapes

2 123 198 39.64851881 82.21079946 6.253034534 70.39906054 grapes

35 140 197 16.77557314 82.75241875 6.106190557 66.76285469 grapes

11 122 195 12.14190714 83.56812483 5.647202395 69.63122027 grapes

6 123 203 12.7567962 81.62497448 6.130310493 66.77844567 grapes

17 134 204 39.04071989 80.18393287 6.499604931 73.88467027 grapes

25 130 197 39.70772192 82.68593454 5.554831977 74.91506217 grapes

27 145 205 9.467960445 82.29335466 5.800242694 66.02765219 grapes

9 122 201 29.58748357 80.91934392 5.570290539 68.06417307 grapes

16 139 203 17.82803682 80.96093443 6.27564088 65.84748763 grapes

32 141 204 8.825674745 82.89753705 5.536645599 67.235765 grapes

22 138 195 27.83487131 83.51444973 6.208196881 73.02882766 grapes

31 144 202 11.02105378 80.55557235 5.870600622 68.23963161 grapes

3 136 205 17.5862944 80.84806564 6.334771461 71.4065452 grapes

28 122 197 19.89363946 82.73366439 5.856575335 69.66256816 grapes

4 136 204 29.93707596 81.77713468 5.898944282 65.52279323 grapes

39 145 201 36.73126647 80.58931938 5.775600435 72.24230804 grapes

38 132 197 20.42094753 81.54185044 5.931101816 66.93065667 grapes

36 133 198 25.51939719 83.98351748 6.2286454 69.17281221 grapes

25 121 201 30.50734778 82.71775569 5.594240603 70.08200379 grapes

15 125 199 18.4269936 80.55625868 5.569230319 69.75734306 grapes

24 140 205 12.087022 83.59398734 5.93202852 68.66813363 grapes

13 132 203 23.60115364 82.48336987 6.423216506 73.23901752 grapes

5 126 197 12.80000387 81.20876367 6.417500829 67.10439401 grapes

30 120 200 38.06099482 82.24729637 6.234904253 65.70148216 grapes

23 142 197 39.06555518 82.03812973 6.000573725 69.30772897 grapes

26 135 203 33.78372897 81.16314317 5.685102769 74.53557341 grapes

7 126 203 16.76201707 82.00335557 5.662140095 73.28712806 grapes

32 139 198 35.89307536 82.66850729 6.358186848 66.53946559 grapes

9 141 202 21.01245395 81.17931863 6.119495295 66.38448261 grapes

20 142 196 10.89875873 80.01639435 6.207600783 68.69420397 grapes

32 129 201 16.36251869 83.00471609 6.48754639 71.55665483 grapes

3 134 199 20.28370163 81.32235739 5.81717753 71.06611222 grapes

38 138 204 25.11108456 83.25447587 6.325480034 73.01026829 grapes

14 131 198 33.4641162 83.86742974 5.562790949 67.92204319 grapes

20 122 204 11.7976469 80.86325389 6.487369687 65.06962486 grapes

40 126 201 11.36300891 80.03100049 6.116982944 71.18289431 grapes

36 128 204 25.23542319 80.68700527 5.695792761 67.03840888 grapes

11 132 197 15.99050693 81.23966573 5.734317007 74.40198861 grapes

0 137 195 22.4359017 80.18612085 6.329499832 65.3973168 grapes

19 123 200 34.76086052 81.03544763 6.167013532 65.70430027 grapes

31 136 197 31.11047251 83.34010951 5.653776058 71.43001582 grapes

4 134 200 28.57828803 80.95628959 5.840256272 73.34232097 grapes

39 139 201 41.18664903 81.01783402 5.539980812 68.68895899 grapes

8 127 196 27.02766138 83.17093908 5.833302165 70.95666003 grapes

39 138 203 21.19339319 82.33098331 6.399433771 74.62834921 grapes

32 120 204 10.38004759 83.44518113 6.138958698 67.3917379 grapes

12 142 203 31.3115978 82.56407013 5.972850838 65.01095312 grapes

8 133 195 20.46657776 80.97598029 6.456079585 71.29813872 grapes

8 139 199 29.36947679 81.53996362 6.336426667 66.13442813 grapes

21 134 202 10.72302459 80.02130636 6.425419926 65.2982112 grapes

40 140 195 14.97846952 80.49979873 6.294395676 71.63437433 grapes

39 127 202 15.3246651 81.67215994 6.477768039 71.60102999 grapes

19 120 195 18.73932187 81.12109244 5.931538447 73.55807954 grapes

21 139 201 19.3642553 83.36094029 5.980598579 67.15094741 grapes

17 136 195 41.20733624 81.61051026 6.389783283 65.90227462 grapes

33 139 203 33.34214482 82.51034633 5.693287415 70.68098614 grapes

22 133 201 23.81995682 80.12211649 6.00299607 67.2739864 grapes

32 130 196 40.66012294 81.24995984 6.372959542 74.03030056 grapes

37 135 205 11.82768186 80.2827185 5.510924849 74.10225057 grapes

15 140 195 13.28504331 83.54193816 5.69945282 65.80006004 grapes

39 132 196 35.83089092 83.32560104 5.778594403 73.67984885 grapes

40 121 199 26.18159716 81.03886263 6.315586313 66.05911698 grapes

40 132 202 24.57558351 80.70695797 5.971813006 69.706113 grapes

29 142 203 29.67229086 83.71498986 5.891195653 66.48490371 grapes

32 121 199 39.37102553 81.25353895 6.129812716 74.08101744 grapes

6 140 205 17.66558428 82.92903419 6.313085601 69.8671263 grapes

8 120 196 24.06679352 82.66396666 6.053662544 69.81855775 grapes

34 133 202 15.31413469 80.09711412 5.804799142 74.82144653 grapes

35 135 199 21.77466746 80.54942557 6.400719746 69.39630398 grapes

16 145 199 26.91624843 80.76838926 5.953966361 69.30927185 grapes

8 136 201 41.65602996 82.22118237 5.609255992 74.19664838 grapes

25 129 195 17.98667801 81.17712085 5.777271492 72.37127689 grapes

16 130 201 29.12033769 82.79092939 5.682395429 68.8503047 grapes

39 129 203 34.38922481 83.18392806 5.863996687 71.03001556 grapes

38 135 203 41.36106301 82.79782954 6.444373116 69.92107482 grapes

33 120 205 35.12158265 82.26890793 5.550832178 69.71518491 grapes

35 125 204 19.6491772 80.15215777 6.107741788 73.69529586 grapes

1 132 200 16.27852801 82.94270065 5.620745638 66.57462809 grapes

39 140 203 21.11903604 80.63399198 6.349875906 69.27779761 grapes

28 145 202 19.2077707 82.9042841 6.484323189 66.83113717 grapes

6 128 200 25.96308415 82.57813624 5.838748311 70.31782647 grapes

6 139 199 25.67385024 81.6212135 6.29099842 74.10919422 grapes

29 122 196 41.94865736 81.15595212 5.638328481 73.06862952 grapes

37 144 197 11.18994268 80.8084305 6.415555956 66.34234944 grapes

38 120 197 17.5438296 82.94703302 6.323722572 73.77063744 grapes

38 141 198 13.05809741 80.28297993 5.757009965 70.75633584 grapes

14 121 203 9.724457611 83.74765639 6.158689406 74.46411148 grapes

6 125 204 27.92004934 82.93262435 5.733539807 69.92092839 grapes

32 138 197 9.535585543 80.73112694 5.908724337 69.44115171 grapes

11 124 204 13.42988625 80.06633966 6.361141107 71.40043037 grapes

23 138 200 9.851242629 80.22631717 5.96537863 68.42802444 grapes

40 143 201 24.97256132 82.72828653 6.476757723 66.70016285 grapes

6 142 202 27.23708304 82.94573346 6.224542938 70.42508897 grapes

37 124 195 18.70679077 83.4795292 6.209928251 66.5964488 grapes

35 134 204 9.949929082 82.55138983 5.841138354 66.00817551 grapes

24 128 196 22.75088787 90.69489172 5.521466996 110.4317855 apple

7 144 197 23.8494014 94.34814995 6.133220586 114.0512495 apple

14 128 205 22.60800988 94.58900601 6.226289556 116.0396587 apple

8 120 201 21.18667419 91.13435689 6.321152192 122.233323 apple

20 129 201 23.41044706 91.69913296 5.587905967 116.0777931 apple

32 137 204 22.86006627 93.12859895 5.824151693 117.7296726 apple

27 139 205 22.48403042 93.40819246 5.772179946 105.5473627 apple

0 123 205 22.02775403 92.96129462 5.790993052 121.1349176 apple

22 144 196 21.91191314 91.68748063 6.499226821 117.0761277 apple

1 124 199 23.71059131 93.27392415 5.658473817 112.6676589 apple

30 122 197 21.37784654 92.72043743 5.573241391 106.1417017 apple

29 121 196 22.84852833 94.32130209 6.079497202 123.5977843 apple

13 126 204 23.1094265 92.79630809 6.383180271 108.183792 apple

9 139 199 23.25230817 94.54128292 5.867420996 105.3558408 apple

0 133 200 23.67287749 90.4935574 5.708418722 104.2298028 apple

30 143 199 23.76881552 90.59810302 5.7983508 102.2648546 apple

36 140 198 23.34386401 91.47684705 6.28188384 104.4267991 apple

37 137 199 22.63946441 90.18451645 5.697945522 108.3405879 apple

33 121 203 22.45696744 94.76285385 5.605934087 114.8407725 apple

7 144 195 22.96388477 93.58065995 5.85648105 104.6472986 apple

35 128 205 21.07273439 93.56585985 6.041053829 107.8737015 apple

29 128 198 22.44075021 92.70785115 5.685062404 121.4977331 apple

2 143 196 22.71271308 90.45261746 5.669489065 109.8852597 apple

34 140 198 21.70416965 93.44006288 5.751707342 115.1781396 apple

29 144 204 22.43324518 92.48667725 5.800448951 119.1025189 apple

32 141 203 21.25941052 92.84416234 5.821347769 109.0658471 apple

13 144 197 22.9215706 94.89613443 6.28022267 105.6941544 apple

25 143 198 22.81212536 91.51861705 6.027314401 107.855225 apple

9 137 200 21.12152071 90.6878768 5.636687393 102.8017203 apple

6 144 198 21.11478672 90.31528693 5.559363609 104.5086618 apple

37 126 196 23.59997268 90.97597665 5.596449493 107.1728191 apple

2 120 203 23.12652652 94.71203306 5.893492999 108.6211833 apple

11 143 197 22.98458907 93.3204487 5.875718516 122.1952483 apple

10 141 201 22.12659387 90.97818277 6.386021424 104.5412275 apple

24 142 202 22.53779727 91.48135786 5.710819862 101.8474768 apple

23 138 195 22.49095104 91.70292746 5.795985716 124.3915101 apple

18 125 204 22.35548159 94.47811755 6.046673619 116.7366261 apple

13 121 196 22.20700989 93.50574163 6.443382913 120.1593771 apple

26 122 202 22.44516988 94.73763514 5.617227184 107.1843273 apple

28 123 202 22.76643029 92.12438519 6.442289294 120.4359949 apple

26 121 201 22.19109412 90.02575116 6.162034371 112.3126628 apple

21 137 196 23.6119202 91.70293849 5.812781806 123.5900822 apple

21 135 198 23.86087054 94.92048112 5.765015126 105.0241329 apple

5 144 205 21.42177231 92.62665309 6.184922574 102.8045658 apple

2 123 205 22.36629253 90.78572467 5.739652177 124.9831618 apple

15 133 199 23.99686172 91.61001707 5.824778636 117.6102915 apple

31 130 198 21.80129837 92.73446667 5.554823557 120.0586671 apple

25 143 200 23.80436344 92.80441624 6.024248787 100.6192543 apple

16 143 204 23.71475278 91.53331177 5.631333387 121.8961665 apple

19 122 202 23.34467359 90.37981478 5.811975094 112.8954016 apple

10 125 196 22.31253665 90.03577124 5.730557448 113.0688155 apple

20 139 202 23.50201428 92.21083961 5.66999105 107.9868949 apple

28 123 198 23.46260321 91.45665004 5.682751473 111.7763395 apple

28 136 200 23.06204373 92.39544055 6.245858905 114.7399101 apple

2 131 199 22.47420512 91.22759742 6.017370134 124.2179699 apple

2 140 197 22.69780133 92.82223419 5.53456749 105.0508234 apple

27 138 201 23.66682067 93.90191078 5.952367662 105.4004751 apple

30 127 204 22.50050273 92.45878335 6.126436584 100.9343903 apple

32 145 203 23.83053666 90.84422164 6.406818518 109.5966791 apple

29 139 205 23.64142354 93.74461474 6.155939453 116.6912176 apple

26 126 195 21.41363812 92.99124545 5.878568981 118.3979065 apple

40 136 202 22.85267372 94.5764581 5.935336308 117.5314026 apple

6 124 200 22.98208095 93.84505029 5.971332179 109.5852253 apple

35 138 200 21.19909519 90.80819418 5.67130617 103.6838922 apple

17 136 196 23.87192332 90.49939035 5.882155988 103.0548094 apple

33 134 205 21.0365275 94.33919546 6.08551916 114.7412734 apple

16 143 197 22.61711614 93.51978375 5.90402645 116.9256766 apple

27 120 200 21.45278675 90.74531921 6.110218826 116.7036582 apple

29 145 205 22.81227579 92.12992101 6.212302608 109.3383552 apple

3 141 197 21.98141856 91.12719303 6.142803397 115.4789148 apple

15 123 204 22.52709326 92.54780429 6.365972688 115.3830068 apple

5 136 195 22.35628673 91.92360477 6.264202804 107.7697413 apple

10 136 204 21.19852186 92.15595143 6.276198595 105.8554351 apple

7 141 195 23.8812458 93.45067555 5.514253142 104.9116663 apple

2 129 201 22.78234161 94.36803516 5.682343744 122.1449949 apple

29 138 197 22.19055385 92.43764169 5.830892252 121.6622761 apple

30 137 200 22.91430043 90.70475565 5.603413172 118.6044645 apple

29 132 204 23.08950736 90.22507299 6.0967531 108.2166601 apple

14 139 197 21.72484506 92.83975602 6.056529526 121.6961761 apple

18 125 203 22.44307715 91.59234006 6.160267496 102.5565807 apple

33 143 204 21.1316077 91.95769858 5.814434775 122.5391946 apple

40 144 196 22.71750705 92.25479855 5.987262638 107.0289866 apple

9 143 197 23.75033085 92.88160462 5.570020684 117.6602827 apple

38 135 203 23.76121837 93.661643 5.965551311 100.825956 apple

28 130 196 22.13450646 94.67695747 6.062356467 112.9203223 apple

35 142 203 21.17089176 90.23730166 5.895319002 123.6495149 apple

12 129 205 22.36238282 91.15761594 6.119432215 118.6832725 apple

1 135 203 22.77856513 92.70124029 5.624203283 113.7759219 apple

0 145 205 21.22503442 90.09877774 5.52078314 113.9760462 apple

31 121 201 23.15791104 90.34396882 5.731535258 110.712841 apple

35 131 203 22.42776057 93.91722423 5.893490899 102.7230739 apple

29 140 195 23.64082979 90.95257927 5.560521058 116.7431319 apple

33 138 198 22.29423493 90.69033986 6.222390798 122.7418744 apple

14 140 197 23.35225078 90.90054697 6.071255131 113.0381382 apple

35 145 195 22.03911546 94.58075845 6.231950009 110.9804014 apple

40 120 197 23.80593812 92.48879468 5.889480679 119.6335548 apple

25 132 198 22.31944084 90.85174383 5.732757516 100.1173443 apple

31 137 196 22.14464104 93.82567435 6.400321212 120.6310784 apple

36 144 196 23.65167552 94.50528753 6.496934492 115.3611268 apple

10 140 197 22.16939473 90.27185592 6.229498836 124.4683112 apple

22 30 12 15.78144173 92.51077745 6.354006744 119.035002 orange

37 6 13 26.03097313 91.50819306 7.511755068 101.2847738 orange

27 13 6 13.36050601 91.35608208 7.335158382 111.2266885 orange

7 16 9 18.87957654 92.04304496 7.813916603 114.6659511 orange

20 7 9 29.47741671 91.57802915 7.129136941 111.1727497 orange

26 27 10 28.06903173 92.91487288 6.079998496 114.1339416 orange

5 23 15 25.66901098 92.04670813 7.408939392 112.5424199 orange

0 18 14 29.77149434 92.00719952 7.207991261 114.4161786 orange

39 24 14 30.55472573 90.90343769 7.189259647 106.0711985 orange

13 23 6 23.96147583 90.26408017 7.365338111 102.6958703 orange

21 17 15 23.98289638 91.5473145 7.455991072 118.4901697 orange

33 12 8 25.26052689 90.31153735 6.822282114 117.3695296 orange

6 9 12 31.08368929 90.14362642 7.028746406 109.6894658 orange

19 7 10 14.78003032 91.22062116 6.118430299 100.1961762 orange

24 18 6 26.56608303 94.45239715 6.285312759 116.3796525 orange

9 11 8 24.85903405 94.39000473 6.559236744 111.7803734 orange

31 8 7 34.51465139 93.63812684 7.163245982 103.5684926 orange

22 17 5 24.12188673 90.72351622 6.945562889 102.835632 orange

13 5 8 23.85340379 90.10522549 7.474710503 103.923226 orange

16 8 9 24.60297538 91.28408653 7.601189843 111.2948115 orange

4 13 6 15.63211033 94.25966183 7.561143224 101.4705704 orange

0 25 14 19.33516809 91.97978938 6.361671475 116.450422 orange

8 7 10 28.2620488 91.98317355 6.929216014 105.2132259 orange

4 23 5 22.67594476 93.36348717 7.477935216 110.3332655 orange

33 14 8 21.03200078 92.9641969 7.684420446 110.6823944 orange

30 7 15 33.23453301 91.06053924 7.825531916 115.7659902 orange

21 29 12 22.30318989 92.15987039 6.438668989 117.3688104 orange

11 14 5 11.50322938 94.8933184 6.946354724 115.5683776 orange

9 8 15 14.34320488 94.35734702 7.994465371 110.2223123 orange

5 18 14 33.1056981 93.48447453 7.434118807 119.1709113 orange

29 25 14 30.49183837 90.4582865 7.781988584 113.3302105 orange

33 12 15 30.25578031 92.03272799 6.052318465 116.7173125 orange

8 16 6 12.22816189 90.26457428 7.106650373 108.4161706 orange

15 14 8 10.01081312 90.22399223 6.22094286 119.3941064 orange

16 7 8 22.79196751 90.60901895 6.420457311 116.5084074 orange

0 12 7 20.18432263 90.65458473 6.969249676 116.8130969 orange

5 25 6 30.72119881 94.01331956 6.011302181 106.8118019 orange

6 8 11 24.35590861 92.39651663 6.600948788 119.6946577 orange

10 5 5 21.21306973 91.35349216 7.817846496 112.9834361 orange

1 17 6 10.78689755 91.38411917 6.8198271 117.5293447 orange

1 30 10 11.89925671 91.34663797 7.291405641 103.5771468 orange

0 23 15 22.56664172 93.37488907 7.598729065 109.8585753 orange

24 27 9 18.86883219 93.24688124 6.157135092 119.3936976 orange

36 11 13 17.34083741 93.04897191 7.1917274 112.7194284 orange

40 21 8 34.90665289 92.87820148 7.418761774 102.1906333 orange

40 22 6 24.53610067 91.90997228 6.488221135 115.9787989 orange

32 18 13 13.8377282 91.74780462 6.044167236 107.9873218 orange

9 10 10 22.3551049 93.52211892 6.010391864 101.5164589 orange

13 16 8 34.74004942 93.12316972 6.949838549 100.1967854 orange

15 9 11 11.54785707 94.14861001 7.907956251 108.8289171 orange

29 11 5 23.13338811 91.94670335 7.639788459 104.4224145 orange

1 15 9 29.98364695 94.55239717 7.53350946 115.3560318 orange

18 5 11 20.87947369 90.93756231 6.251586885 102.4550786 orange

14 22 9 17.24944623 91.13772765 6.543191814 112.5090516 orange

33 15 7 15.83388699 91.68293851 7.651225301 109.7571416 orange

4 6 7 23.01014302 91.11764246 6.708889665 112.6738296 orange

17 16 14 16.39624284 92.18151927 6.625538653 102.944161 orange

12 20 10 24.45132792 93.10527686 6.528354932 109.4711098 orange

34 29 8 31.87859192 91.15248149 6.450640306 105.3437825 orange

39 28 10 31.34920143 91.48247612 7.181907673 109.1549823 orange

31 25 12 18.05142392 90.03969587 7.016482298 111.7793889 orange

12 6 8 30.84835031 92.86773675 6.388617138 107.4142681 orange

12 29 13 22.45616931 91.52781832 7.57125447 118.0069295 orange

26 11 11 13.70319166 90.95589386 7.609348255 106.2944879 orange

19 24 15 20.48954522 93.72485075 7.137136973 111.8391951 orange

39 21 9 13.20844373 94.02769434 6.354022554 106.2696156 orange

16 29 13 32.31944397 93.67804556 6.196907944 117.6236473 orange

36 29 13 20.68185224 90.91510525 7.829507245 109.7513927 orange

37 23 12 31.52675982 90.50621806 6.395258356 113.1169398 orange

39 9 15 25.35467646 91.81183218 7.992041984 116.7555937 orange

31 5 14 17.66545409 91.69865887 6.583411671 110.6857506 orange

18 12 8 12.59093977 91.81668769 6.206053072 119.3916718 orange

20 20 10 11.86631922 93.68394562 6.976997772 106.060149 orange

5 8 5 11.03367937 92.22706805 6.562594972 112.7715925 orange

20 8 12 25.2990432 94.96419851 7.260416405 117.9733424 orange

25 21 11 32.23797837 90.15406807 6.460044778 104.7052254 orange

14 19 14 17.68408797 94.35815354 6.699164936 108.0638166 orange

37 18 12 10.2708877 90.19147747 7.401121811 106.6955204 orange

26 15 6 17.22034507 94.78797376 6.912033409 108.0054343 orange

13 22 5 19.667056 90.50096668 7.764040111 100.1737964 orange

32 25 9 10.35609594 93.75652041 7.796034006 101.1456947 orange

19 7 9 27.255435 91.71369387 6.969883483 101.139435 orange

28 7 9 34.5917846 92.13229786 6.730757538 115.5650287 orange

24 30 11 32.39523995 94.51768464 6.601395755 113.25373 orange

7 17 10 10.16431299 91.22320999 6.465913274 106.362551 orange

18 23 8 21.49118657 93.43949693 6.41354791 101.4819888 orange

7 20 12 16.53460397 94.76759975 6.475275337 110.0447896 orange

20 23 11 31.8520694 90.12220323 6.407715561 109.9455062 orange

18 14 11 28.04799508 90.00621688 6.550814117 117.1311498 orange

34 11 10 31.75048899 94.59551226 7.36220835 115.1989301 orange

20 29 10 29.07412717 93.27189064 7.36549204 100.7896871 orange

37 24 13 19.14381903 90.71037456 7.8546243 108.0230792 orange

12 8 10 16.14820285 91.4448027 7.995848977 107.4287664 orange

34 10 14 34.05296914 92.05811721 6.725600855 116.8020848 orange

6 13 9 34.51423957 90.56151463 7.786725333 118.3271968 orange

27 30 5 32.71748548 90.54608254 7.656978112 113.328978 orange

13 8 12 25.16296632 92.54736032 7.105904818 114.3117197 orange

6 7 7 27.68167318 94.47316879 7.199106204 113.9995146 orange

40 17 15 21.35093384 90.9492967 7.871063004 107.0862095 orange

31 26 9 11.69894639 93.25638873 7.566165721 103.2005992 orange

133 47 24 24.40228894 79.19732001 7.231324765 90.8022356 cotton

136 36 20 23.09595631 84.86275707 6.925412377 71.29581071 cotton

104 47 18 23.9656349 76.97696717 7.633437412 90.75616738 cotton

133 47 23 24.88738107 75.62137159 6.827354668 89.76050416 cotton

126 38 23 25.36243778 83.63276077 6.176716425 88.43618918 cotton

126 50 19 24.69457084 81.7358876 6.628722836 78.58494391 cotton

113 41 20 25.0017188 80.53965818 7.256877571 96.32600992 cotton

121 45 22 22.45942937 81.30681027 6.443785385 64.23026638 cotton

121 47 16 23.60564038 79.29573149 7.723240151 72.49800885 cotton

129 60 22 24.58453146 79.12404171 5.947448589 71.94608134 cotton

107 45 25 23.0865933 83.55546146 7.227745516 71.84080724 cotton

122 59 18 23.5000992 83.63488952 6.219469084 79.81328183 cotton

140 38 15 24.1472953 75.88298598 6.021439523 69.91563467 cotton

102 49 21 24.69315538 84.84422454 6.253343655 89.799462 cotton

111 40 25 24.484692 84.44932014 6.187455799 90.94342484 cotton

131 35 18 24.49112609 82.24415809 7.057693366 64.02949379 cotton

135 43 16 23.47986888 81.73049149 6.720449769 86.76287924 cotton

100 46 18 24.18586246 76.04203958 6.431689506 69.08056728 cotton

123 39 24 25.00755095 78.17952126 7.453106264 86.06411872 cotton

117 56 15 25.99237426 77.0543546 7.368258226 89.1188212 cotton

121 36 24 23.66457347 81.69105088 7.352401887 99.36898373 cotton

101 58 18 25.66891439 81.38103349 6.652143699 78.59595817 cotton

107 42 24 22.04612876 84.62978302 6.144631795 86.00758678 cotton

100 41 22 22.4204752 84.55794703 7.318802162 93.46595573 cotton

125 39 21 25.03149561 82.21276599 7.954629324 95.0191318 cotton

105 60 23 23.53371386 77.21705554 6.207652157 87.54004943 cotton

102 46 19 22.77076388 82.5993307 6.631005298 81.49543437 cotton

131 49 22 25.49848236 79.9751579 7.306918817 67.05961949 cotton

139 35 15 25.248679 83.4630147 5.898293044 86.55517751 cotton

108 36 19 22.78249615 77.51235009 7.238566893 64.61444234 cotton

118 45 23 23.37044424 77.43198948 7.977651226 71.67870701 cotton

107 51 22 24.86560781 78.22080815 5.983075895 79.56866268 cotton

125 60 17 24.14386157 84.51591287 6.785723961 80.36146974 cotton

113 37 20 25.03300222 79.04368718 7.393441155 97.10087029 cotton

131 52 16 23.65724079 84.47601498 6.486068274 88.54479121 cotton

115 48 16 25.54359718 84.09229796 7.175934962 88.94245493 cotton

113 38 25 22.00085141 79.47270984 7.388265888 90.42224164 cotton

111 41 18 23.64328417 78.1258666 6.10539819 80.96157332 cotton

111 53 19 23.96436009 78.02763149 6.419536555 84.63148859 cotton

122 48 16 24.65425757 75.6350708 6.307585854 61.82980133 cotton

108 46 17 24.3017998 84.87668973 6.93221485 65.0247867 cotton

132 41 22 24.29144926 81.02453404 7.810865753 90.41694635 cotton

103 42 17 24.29470232 84.61527627 6.527541661 81.05902285 cotton

133 50 25 25.72180042 81.19666206 7.569454601 99.93100821 cotton

127 37 18 24.87663664 76.30050373 7.041065585 91.9223468 cotton

110 39 25 22.60612115 77.34264002 7.208795456 75.13617229 cotton

131 38 19 23.86814008 75.68339729 6.814341946 90.4547185 cotton

108 38 24 23.41022496 76.43836957 7.442217061 78.82199603 cotton

122 40 17 24.96440768 81.31677618 6.854558957 80.03995829 cotton

111 50 15 25.16820129 80.30351815 7.884550475 84.62419032 cotton

140 40 17 22.72767171 77.07598065 6.006085786 77.55176318 cotton

100 40 20 22.45145981 76.25674874 7.432043735 86.84998693 cotton

123 50 16 23.04920461 75.53835214 6.498052108 70.65644296 cotton

107 36 21 25.29250148 75.66653335 6.205263534 62.64174227 cotton

118 50 19 22.95604064 82.33733678 6.360812227 66.48339303 cotton

103 51 20 22.80213132 84.14668447 7.046607434 91.6389565 cotton

133 57 19 23.54234715 75.98203329 7.947011366 84.12536744 cotton

129 47 20 24.41212325 80.80343786 6.281913858 98.60457373 cotton

116 52 19 22.94276687 75.37170612 6.114525877 67.08022574 cotton

114 40 23 25.53676123 81.13668716 6.753978061 95.4262599 cotton

131 60 17 25.32023717 81.79475917 7.425041316 83.46532547 cotton

107 43 18 22.426733 81.53480799 6.745104394 65.54475812 cotton

123 44 21 25.78544484 75.00539324 7.641116569 91.39578861 cotton

112 49 25 25.68959532 77.90621048 6.470135478 66.19426787 cotton

119 44 15 22.14593688 82.8597549 7.091992365 60.65381719 cotton

130 59 19 25.07278712 82.50257909 6.520403794 93.51042684 cotton

127 53 24 22.21506982 76.17851932 6.127939628 70.40557612 cotton

134 52 18 23.9643129 76.59175937 7.994679507 76.13090645 cotton

109 36 18 25.40059227 76.53237965 7.524707577 62.5138867 cotton

100 48 17 23.7805123 83.03878838 7.827877818 66.26555904 cotton

132 52 19 24.16402322 76.7433897 6.436691764 61.94626051 cotton

102 37 25 25.31468463 77.91757121 5.907930899 72.82902109 cotton

111 39 22 22.60361557 80.3509046 6.135025006 88.57395505 cotton

117 51 15 22.9535715 78.71555832 6.044556594 99.75336197 cotton

136 36 24 22.74446976 80.41198458 7.59781958 90.07326633 cotton

134 56 18 23.80834611 83.91902605 6.691268104 70.97358303 cotton

112 54 15 25.46228792 81.56641891 6.175492306 76.88582484 cotton

105 56 15 25.96779712 81.97904282 7.272316209 74.14169043 cotton

140 45 15 25.5308271 80.04662756 5.801047545 99.39557151 cotton

126 46 25 24.43847399 81.69801729 6.757457943 60.79645852 cotton

106 49 24 23.03887865 76.47039772 6.983395573 90.64770699 cotton

121 53 19 23.51308653 76.72621429 7.976889498 80.11272117 cotton

108 60 17 22.75805656 76.75768356 6.558902588 97.76600619 cotton

116 56 17 24.71252544 77.7293114 7.979090365 85.24963302 cotton

100 52 19 23.45969093 82.44777468 7.903528673 93.50153555 cotton

129 43 16 25.5503704 77.85055621 6.73210948 78.58488484 cotton

118 44 23 22.08458267 82.82904143 6.691690476 67.06459777 cotton

117 43 25 24.68854799 78.51206972 7.839849298 69.31153566 cotton

126 37 21 25.84997269 84.16855231 6.61448588 77.03421249 cotton

120 48 16 22.46054478 75.40989245 7.456971816 71.85436078 cotton

102 45 16 23.65629976 77.52425987 7.2942193 74.8984994 cotton

131 56 20 22.00817088 81.83896111 7.762647875 92.23645249 cotton

114 40 17 24.32630461 80.13456404 6.363406102 69.45072055 cotton

101 37 18 22.92360984 82.68738535 7.63737841 92.91915074 cotton

106 46 20 23.43821725 78.63388824 6.200671976 81.15072105 cotton

113 38 20 22.10718988 78.58320116 6.364729934 74.94136567 cotton

102 53 21 23.03814028 76.11021529 6.913678684 91.49697481 cotton

110 39 18 24.54795322 75.39752705 7.766259769 63.88079866 cotton

107 58 15 23.73868041 75.77503808 7.55606399 76.63669195 cotton

120 60 15 22.31871914 83.86129998 7.288377241 65.35747011 cotton

89 47 38 25.52468965 72.24850829 6.002524871 151.8869972 jute

60 37 39 26.59104992 82.94164078 6.033485257 161.2469997 jute

63 41 45 25.29781791 86.8870535 7.121933579 196.6249511 jute

86 40 39 25.72100868 88.16513579 6.207459637 175.6086697 jute

96 41 40 23.58419277 72.00460848 6.090060478 190.4242157 jute

100 35 36 25.31042337 72.01364411 6.346715209 190.5577618 jute

63 37 43 23.41798979 85.08640476 6.661957897 185.7446728 jute

70 43 40 24.35564134 88.80391021 6.176860192 169.1168028 jute

67 55 44 26.284017 75.14640198 7.251847296 182.2685447 jute

74 40 40 25.13842773 83.12053888 6.386259978 169.3388465 jute

89 53 44 24.88692811 71.91711523 7.319735475 150.2498675 jute

74 46 45 25.75734909 88.36668522 6.025028997 189.4263485 jute

89 41 38 23.12844351 74.68322732 6.344751947 199.8362913 jute

60 55 40 24.9949957 88.95692783 7.02777956 151.4935635 jute

67 43 38 25.21622704 70.88259632 7.299304715 195.8645552 jute

70 38 35 24.39736241 79.26861738 7.014063944 164.2697011 jute

74 49 38 23.31410442 71.4509053 7.488014404 164.4970373 jute

90 40 39 25.72668885 81.86171563 6.626503893 191.9649389 jute

82 35 44 26.96656378 78.21047693 6.239011 169.8391177 jute

73 45 37 23.70467146 74.63745355 6.742688094 181.2783964 jute

85 53 38 24.90075709 73.84186449 6.588017308 153.8990984 jute

81 56 36 23.39605743 72.60512854 7.097586415 174.7876411 jute

84 55 38 26.8748389 79.78725152 6.956682743 173.1017097 jute

80 45 42 23.1426498 74.99739774 7.380396262 151.9035477 jute

76 54 45 24.29496635 77.62976013 6.176618831 184.9800516 jute

76 56 39 24.39459498 89.89106506 6.551130445 197.1220049 jute

81 40 45 25.7629429 80.76238215 6.427726565 174.5071843 jute

76 44 45 25.4879684 84.48235878 6.740947635 168.7848886 jute

69 47 40 25.37122686 76.2403666 6.130136384 183.8270791 jute

82 40 45 26.21312799 81.70476368 6.667633355 180.1237765 jute

69 57 35 24.30748599 78.54340987 6.186814392 186.2337571 jute

81 36 38 23.76554749 87.98329901 6.334837865 150.3166152 jute

67 60 38 24.79853023 78.53037059 7.16214284 162.2847429 jute

72 51 40 23.20683504 74.09956958 7.422318499 199.4766779 jute

65 39 45 23.66805429 70.89000744 6.768001309 184.4633281 jute

78 50 43 25.12417673 85.72530641 6.348441469 159.5718087 jute

77 52 41 23.89069041 83.46409075 6.097294061 167.7230632 jute

89 52 42 23.09433785 81.45139295 6.14132902 196.6587013 jute

62 49 37 24.21744605 82.85284045 7.479248124 166.1365886 jute

90 48 45 24.06475727 71.31342851 6.509174789 153.6390212 jute

66 47 36 24.85441411 74.4407048 6.57256106 175.572958 jute

80 52 39 26.41915161 76.85691248 7.165696848 197.2101782 jute

89 52 45 24.89326318 77.01222585 7.207457208 196.469984 jute

77 51 44 23.25583402 82.7015932 7.124333547 166.2160846 jute

94 37 41 24.7634518 87.06071115 6.463538707 179.1630865 jute

75 41 35 24.97042599 78.62697699 6.856833064 166.6415254 jute

60 55 36 26.12797248 80.49172597 7.132389299 150.6326874 jute

62 56 35 25.97825807 81.65769588 6.235357638 163.3488091 jute

84 40 42 26.2830571 73.35763537 6.704273839 186.6898282 jute

100 56 40 26.38905406 83.31240346 7.433313409 176.1516409 jute

75 56 44 25.2746335 73.7459581 6.109478059 168.0432282 jute

78 46 42 23.09499564 78.45959697 7.095413294 155.3851533 jute

82 48 36 25.79351957 81.76904006 6.352076783 193.2418382 jute

100 58 41 23.17403323 87.88255345 6.658769991 160.6217342 jute

88 50 40 25.63215038 79.95150917 7.051822472 182.2582277 jute

67 41 40 25.848795 87.81661683 7.333143205 152.6194403 jute

72 42 43 26.56767277 80.90424543 6.352771037 181.2915605 jute

89 40 43 26.24532085 72.97198375 7.124050134 189.9711184 jute

89 57 43 26.91515043 73.19897535 6.998787171 177.2233048 jute

61 41 44 24.36972377 82.11319791 6.537914958 159.9210934 jute

79 45 43 25.71901283 79.15532398 7.171054239 187.1735424 jute

84 40 43 25.01157559 88.3313023 7.228268228 169.4168014 jute

98 43 35 25.40785911 76.44048625 7.319952206 188.6372826 jute

75 36 44 23.28081 74.27607475 6.613341343 153.7447398 jute

89 58 35 23.98651719 82.09053379 6.096838784 167.0576456 jute

91 41 37 24.48556447 83.20630007 6.132570523 192.2316221 jute

77 48 36 25.86705009 84.09985284 7.36008498 154.8390847 jute

66 58 35 23.5643831 79.46283115 7.321619041 185.25947 jute

62 59 41 24.2248758 74.89465426 7.175170657 192.4931257 jute

82 35 35 25.49386782 86.97061481 7.299076163 176.5268267 jute

61 41 35 24.97178693 79.47557931 6.842966479 195.7571622 jute

99 57 38 24.80624984 82.09281674 6.356295568 156.3616174 jute

70 42 43 23.16814977 76.66724969 6.508342839 157.1215052 jute

90 59 35 24.25133493 89.86454053 7.098227926 175.1742112 jute

73 43 42 26.58361011 78.00774772 6.310699968 154.8238864 jute

67 46 44 26.82489244 78.20392774 7.093328631 153.9199807 jute

84 37 42 25.49674786 81.13449097 6.691074249 169.9288234 jute

72 41 36 24.09874353 80.57226761 6.187746776 176.8604109 jute

71 56 37 23.18866654 86.20899734 6.491506245 176.103677 jute

64 53 38 26.24347471 78.51063754 6.855362875 183.4065252 jute

65 54 39 23.75091572 71.14782585 7.124571593 160.0889553 jute

60 58 37 26.13871511 79.1188943 6.067302109 171.4892533 jute

86 39 43 26.14576648 71.23690851 6.432051512 193.1007598 jute

90 50 44 26.91643698 73.48655995 6.253408852 171.4716375 jute

91 38 36 26.5232969 77.17331847 7.287318723 157.8548562 jute

87 48 38 23.81579631 80.94023552 7.161865733 190.312216 jute

72 41 36 26.50838667 86.84264005 6.065898283 152.9801697 jute

71 54 35 26.63952463 70.95705996 7.311077075 199.3355744 jute

82 46 41 23.3250131 79.79609448 6.581693772 187.3096148 jute

71 52 43 26.47549543 73.96164569 6.732826127 180.2513601 jute

80 43 43 23.78756036 74.36794079 6.014572075 172.6442654 jute

77 55 43 25.49941707 75.99987588 6.663559451 193.7141828 jute

95 57 41 23.24925555 73.65346838 6.434610995 184.7674863 jute

63 47 35 26.98582182 89.05587886 7.432768147 193.8778713 jute

93 43 38 23.61475336 86.14290267 6.987332927 150.2355238 jute

87 44 43 23.87484465 86.79261344 6.718725189 177.5147313 jute

88 52 39 23.92887902 88.07112278 6.880204617 154.6608736 jute

90 39 37 24.81441246 81.68688879 6.86106911 190.7886386 jute

90 39 43 24.44743944 82.286484 6.7693455 190.9684885 jute

84 38 43 26.57421679 73.81994896 7.26158085 159.3223075 jute

91 21 26 26.33377983 57.36469955 7.261313694 191.6549412 coffee

107 21 26 26.45288458 55.32222678 7.235070264 144.6861336 coffee

83 38 35 25.70822684 52.88667115 7.18915558 136.7325092 coffee

108 24 31 24.12832546 56.18107663 6.431899748 147.2757818 coffee

116 28 34 23.44372334 60.39523266 6.42321105 122.2103248 coffee

116 23 25 23.4123707 52.26994674 6.869720196 139.3670753 coffee

109 31 27 23.05951896 50.40609436 6.973839707 164.4971875 coffee

89 25 34 23.07895447 63.65861483 7.184801627 129.8765443 coffee

118 18 32 27.6496114 51.11044023 6.351823783 122.8392822 coffee

111 32 34 25.46743689 69.35161206 6.392048018 171.3764462 coffee

84 36 28 26.7350622 55.55164819 6.119892347 140.6305213 coffee

85 33 25 26.20811417 52.50987966 6.910823945 189.0944824 coffee

99 15 27 27.0424167 57.27927475 6.501157208 165.6872119 coffee

81 30 31 24.65090184 51.93952357 7.027585559 135.1386537 coffee

95 39 29 27.35152643 55.99375012 7.13411409 148.9812525 coffee

81 34 30 25.17787724 62.26244581 6.647765997 135.0119649 coffee

80 15 28 23.11438731 68.00096043 6.703270635 161.8944624 coffee

104 20 26 27.22783677 52.95261751 7.493191968 175.7260273 coffee

109 29 28 23.26316991 60.5160021 6.724688503 194.1755471 coffee

100 32 26 25.234661 57.53161469 6.043485685 124.2261737 coffee

100 24 28 25.59535262 57.72920846 7.101661011 195.7733251 coffee

83 21 28 25.5674832 60.49244602 7.466900683 190.2257843 coffee

120 23 28 25.67324193 51.29043632 6.877799264 196.2736367 coffee

104 26 30 24.40726724 62.65692638 6.410992833 148.6977358 coffee

108 33 31 23.69287069 66.76090123 7.393825704 144.6576424 coffee

91 25 26 24.53460016 66.99765375 7.482414225 180.5059257 coffee

86 26 27 27.13140403 52.89368299 6.081172981 192.4280381 coffee

98 18 27 27.56088634 68.49299897 6.516312148 167.4358075 coffee

111 27 31 23.59302313 55.27564977 6.043330951 191.3980675 coffee

84 39 35 23.17714381 52.13864034 6.959404135 117.3113562 coffee

98 27 27 24.71384065 51.29142534 7.238109556 197.6439711 coffee

118 21 34 24.38534644 64.72543073 7.234258375 119.6324109 coffee

103 27 31 27.15998538 51.59100753 6.691541233 126.1752206 coffee

82 24 33 26.53543168 67.09608099 6.809593554 120.6494434 coffee

86 31 35 27.01207284 60.76645256 6.485761419 191.4508931 coffee

88 35 35 27.55906475 58.45742907 6.784460602 117.9389993 coffee

84 27 29 23.32293161 53.00366334 7.167092586 168.2644287 coffee

120 40 33 24.23850608 54.30329632 6.73410539 115.1564012 coffee

106 40 30 23.42611644 64.10651528 6.779984384 122.6847408 coffee

113 21 33 26.02241444 55.83288958 7.277422738 176.9020924 coffee

117 34 25 24.83846178 56.7685316 7.21270048 124.4135035 coffee

80 30 25 26.24092174 65.64381357 7.487266991 148.3771202 coffee

88 21 27 24.43011925 66.02411187 7.231166546 181.6368274 coffee

113 33 34 26.00373964 62.1445102 6.559817161 153.477776 coffee

87 23 28 26.22367404 62.26594559 6.979590627 193.7461968 coffee

113 15 29 27.09617155 63.55324262 6.779230041 190.2440566 coffee

98 29 30 25.64004392 61.03273481 6.217974349 199.4735636 coffee

97 29 27 27.74576987 54.36976075 7.205078785 139.8619431 coffee

85 35 32 26.24928198 54.28617819 6.854011265 133.1120232 coffee

82 29 35 26.67377159 52.24226285 6.246872394 156.1543898 coffee

103 33 25 27.10210397 55.7497332 6.911066044 139.5013171 coffee

112 17 28 27.62975458 61.26002598 6.777417989 196.6492664 coffee

99 19 33 27.5364547 55.51673151 6.273741983 130.6377143 coffee

120 20 34 23.56960509 50.56339727 6.906124587 130.3797119 coffee

114 27 28 24.99451759 57.93250202 7.162802357 192.8736822 coffee

100 40 35 27.56441788 54.41094079 6.955787351 177.816092 coffee

108 35 25 23.98143338 61.10935084 6.971963169 161.5279095 coffee

115 31 30 24.22984659 67.37768353 6.840927967 122.4073418 coffee

87 28 30 25.60153969 68.66257977 6.536676653 168.8383605 coffee

82 24 26 24.31274458 53.57285558 6.089443603 184.4103931 coffee

94 26 27 26.36629861 52.25738495 7.456460375 177.3176161 coffee

87 28 35 26.5602777 57.1621814 6.759211911 152.0616227 coffee

118 40 35 26.35034208 58.50650238 7.460174812 121.5586297 coffee

87 38 29 25.20406808 57.88370456 6.652642579 156.1457255 coffee

92 40 30 23.35723208 55.18792166 6.026287448 171.6976946 coffee

97 22 26 23.60567546 59.68849145 6.074190142 185.1568059 coffee

99 40 32 24.18471151 69.94807345 7.045543056 163.2708732 coffee

89 28 33 26.44414097 53.83876189 6.993236001 175.3723314 coffee

112 39 29 26.12492233 63.37479229 6.726528895 147.8035305 coffee

111 28 26 27.77363343 64.47858698 6.937352845 192.7121236 coffee

114 20 26 25.55656667 62.67087838 7.27905689 193.5866233 coffee

117 26 30 27.92374437 67.96910852 7.079850922 115.2325531 coffee

111 29 31 26.05968403 52.31098539 6.136286518 161.3432535 coffee

119 30 28 26.35770906 64.57578034 6.505203696 163.6269496 coffee

116 40 33 24.91370487 54.15319242 7.042089492 129.5481144 coffee

95 37 35 27.31317116 68.4233391 6.348337519 192.4288139 coffee

86 40 33 26.1387869 52.26311691 7.432322234 136.3027766 coffee

117 37 32 23.1069385 67.06230539 6.787658922 162.5769606 coffee

105 18 35 23.52648086 68.44030686 6.743417121 171.8839938 coffee

109 23 25 25.11711046 68.48030408 7.00733163 194.8773479 coffee

80 18 31 24.02952505 58.84880599 7.303033217 134.6803969 coffee

101 31 26 26.70897548 69.71184111 6.861235184 158.8608887 coffee

103 33 33 26.71717393 50.50148528 7.131435858 126.8073984 coffee

93 26 27 24.59245684 56.46829641 7.288211994 137.7044047 coffee

104 35 28 27.51006055 50.66687215 6.983732393 143.9955548 coffee

116 36 25 27.57847581 58.52534263 6.172090205 156.6810374 coffee

107 38 29 26.65069302 57.56695719 6.35118177 145.105065 coffee

101 33 33 26.97251562 62.0183627 6.908671379 142.8610793 coffee

107 31 31 23.17124551 52.97841162 6.766184468 153.1201644 coffee

99 16 30 23.52652084 65.44340921 6.392791654 186.1728203 coffee

103 40 30 27.30901814 55.196224 6.348316257 141.4831644 coffee

118 31 34 27.54823036 62.88179198 6.123796057 181.4170812 coffee

106 21 35 25.627355 57.04151119 7.428523634 188.5506536 coffee

116 38 34 23.29250318 50.04557009 6.020947179 183.468585 coffee

97 35 26 24.91461008 53.74144743 6.334610249 166.2549307 coffee

107 34 32 26.77463708 66.4132686 6.78006386 177.7745075 coffee

99 15 27 27.41711238 56.63636248 6.086922359 127.92461 coffee

118 33 30 24.13179691 67.22512329 6.362607851 173.3228386 coffee

117 32 34 26.2724184 52.12739421 6.758792552 127.1752928 coffee

104 18 30 23.60301571 60.39647474 6.779832611 140.9370415 coffee

**CHAPTER 8**

**Modules Description**

**Frontend**

* **Home page – description of project**
* **Recommend page – to predict the crop and accuracy of model**
* **Feedback page – user can give the feedback**
* **History – show the history of the recommendations till now**

**Backend**

* **API in Django**
* **Pre-processing of Data set**
* **Training Models**
* **Testing Models**
* **Serializing the Models**
* **Accuracy of Models**

**CHAPTER 9**

**Conclusion**

In a modern environment with less space and less knowledge of agriculture, all the factors are considered from the perspective of farmer and plant, and the farmer is properly guided until the harvesting. Before selecting any plant to grow it is important to have the knowledge and an understanding of the factors that affect the cultivation and how to maintain or control them. From this system, these above-mentioned factors are automatically processed and select the crop type to be cultivated.

Once the plant is cultivated, the farmer is asked for feedback regularly with a time interval of one month. From this feedback taken, the system self-trained, and the accuracy is improved with time and data collected. From this system, the guidance of a specialist is not needed, and the maintenance is less. Thus, implementing this system will not have any additional monetary impact on the user.

**CHAPTER 10**

**Reference**

1. S. Das. *Filters, wrappers and a boosting-based hybrid for feature selection*. In International conference on machine learning 1 (2001), pp. 74–81. [[Google Scholar]](http://scholar.google.com/scholar_lookup?hl=en&publication_year=2001&pages=74-81&conference=In+International+conference+on+machine+learning&author=S.+Das&title=Filters%2C+wrappers+and+a+boosting-based+hybrid+for+feature+selection)
2. H. Liu and Y. Lei, *Toward integrating feature selection algorithms for classification and clustering*, IEEE Trans. Knowl. Data Eng. 17 (2005), pp. 491–502. doi:<https://doi.org/10.1109/TKDE.2005.66>. [[Crossref]](https://www.tandfonline.com/servlet/linkout?suffix=cit0002&dbid=16&doi=10.1080%2F13873954.2021.1882505&key=10.1109%2FTKDE.2005.66), [[Web of Science ®]](https://www.tandfonline.com/servlet/linkout?suffix=cit0002&dbid=128&doi=10.1080%2F13873954.2021.1882505&key=000226996100004), [[Google Scholar]](http://scholar.google.com/scholar_lookup?hl=en&volume=17&publication_year=2005&pages=491-502&journal=%00null%00&issue=%00null%00&issn=%00null%00&author=H.+Liu&author=Y.+Lei&title=Toward+integrating+feature+selection+algorithms+for+classification+and+clustering&pmid=%00empty%00&doi=10.1109%2FTKDE.2005.66)
3. A. Maruf, M. Abdullah, and S. Shatabda, *iRSpot-SF: Prediction of recombination hotspots by incorporating sequence based features into Chou’s Pseudo components*, Genomics 111 (2019), pp. 966–972. doi:<https://doi.org/10.1016/j.ygeno.2018.06.003>. [[Crossref]](https://www.tandfonline.com/servlet/linkout?suffix=cit0003&dbid=16&doi=10.1080%2F13873954.2021.1882505&key=10.1016%2Fj.ygeno.2018.06.003), [[PubMed]](https://www.tandfonline.com/servlet/linkout?suffix=cit0003&dbid=8&doi=10.1080%2F13873954.2021.1882505&key=29935224), [[Web of Science ®]](https://www.tandfonline.com/servlet/linkout?suffix=cit0003&dbid=128&doi=10.1080%2F13873954.2021.1882505&key=000475294900053), [[Google Scholar]](http://scholar.google.com/scholar_lookup?hl=en&volume=111&publication_year=2019&pages=966-972&journal=%00null%00&issue=%00null%00&issn=%00null%00&author=A.+Maruf&author=M.+Abdullah&author=S.+Shatabda&title=iRSpot-SF%3A+Prediction+of+recombination+hotspots+by+incorporating+sequence+based+features+into+Chou%E2%80%99s+Pseudo+components&pmid=%00empty%00&doi=10.1016%2Fj.ygeno.2018.06.003)
4. I. Kononenko, *Estimating attributes: Analysis and extensions of RELIEF*. European conference on machine learning 784 (1994), pp. 171–182. [[Crossref]](https://www.tandfonline.com/servlet/linkout?suffix=cit0004&dbid=16&doi=10.1080%2F13873954.2021.1882505&key=10.1007%2F3-540-57868-4_57), [[Google Scholar]](http://scholar.google.com/scholar_lookup?hl=en&publication_year=1994&pages=171-182&conference=European+conference+on+machine+learning&author=I.+Kononenko&title=Estimating+attributes%3A+Analysis+and+extensions+of+RELIEF)
5. J. Pavel Pudil, N.N. Choakjarernwanit, and J. Kittler, *Feature selection based on the approximation of class densities by finite mixtures of special type*, Pattern Recognit 28 (1995), pp. 1389–1398. doi:<https://doi.org/10.1016/0031-3203(94)00009-B>. [[Crossref]](https://www.tandfonline.com/servlet/linkout?suffix=cit0005&dbid=16&doi=10.1080%2F13873954.2021.1882505&key=10.1016%2F0031-3203%2894%2900009-B), [[Web of Science ®]](https://www.tandfonline.com/servlet/linkout?suffix=cit0005&dbid=128&doi=10.1080%2F13873954.2021.1882505&key=A1995RV43600006), [[Google Scholar]](http://scholar.google.com/scholar_lookup?hl=en&volume=28&publication_year=1995&pages=1389-1398&journal=%00null%00&issue=%00null%00&issn=%00null%00&author=J.+Pavel+Pudil&author=N.N.+Choakjarernwanit&author=J.+Kittler&title=Feature+selection+based+on+the+approximation+of+class+densities+by+finite+mixtures+of+special+type&pmid=%00empty%00&doi=10.1016%2F0031-3203%2894%2900009-B)
6. J. Novovicová, P. Pudil, and J. Kittler, *Divergence based feature selection for multimodal class densities*, IEEE Trans. Pattern Anal. Mach. Intell. 18 (1996), pp. 218–223. doi:<https://doi.org/10.1109/34.481557>. [[Crossref]](https://www.tandfonline.com/servlet/linkout?suffix=cit0006&dbid=16&doi=10.1080%2F13873954.2021.1882505&key=10.1109%2F34.481557), [[Web of Science ®]](https://www.tandfonline.com/servlet/linkout?suffix=cit0006&dbid=128&doi=10.1080%2F13873954.2021.1882505&key=A1996TV66900012), [[Google Scholar]](http://scholar.google.com/scholar_lookup?hl=en&volume=18&publication_year=1996&pages=218-223&journal=%00null%00&issue=%00null%00&issn=%00null%00&author=J.+Novovicov%C3%A1&author=P.+Pudil&author=J.+Kittler&title=Divergence+based+feature+selection+for+multimodal+class+densities&pmid=%00empty%00&doi=10.1109%2F34.481557)
7. H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*, 1st ed., Springer, US, 1998. [[Crossref]](https://www.tandfonline.com/servlet/linkout?suffix=cit0007&dbid=16&doi=10.1080%2F13873954.2021.1882505&key=10.1007%2F978-1-4615-5689-3), [[Google Scholar]](http://scholar.google.com/scholar_lookup?hl=en&publication_year=1998&pages=%00empty%00&author=H.+Liu&author=H.+Motoda&isbn=%00null%00&title=Feature+Selection+for+Knowledge+Discovery+and+Data+Mining)
8. M. Dash, K. Choi, P. Scheuermann, and H. Liu. F*eature selection for clustering-a filter solution*. IEEE International Conference on Data Mining, 2002, Proceedings., Maebashi City, Japan, pp. 115–122. [[Crossref]](https://www.tandfonline.com/servlet/linkout?suffix=cit0008&dbid=16&doi=10.1080%2F13873954.2021.1882505&key=10.1109%2FICDM.2002.1183893), [[Google Scholar]](http://scholar.google.com/scholar_lookup?hl=en&publication_year=2002&pages=115-122&conference=IEEE+International+Conference+on+Data+Mining&author=M.+Dash&author=K.+Choi&author=P.+Scheuermann&author=H.+Liu&title=Feature+selection+for+clustering-a+filter+solution)