GrowGuardian-AI For Farming

A report submitted for the course of

Application Development_ Deep Learning Explore

IV B. Tech I Semester

by

A.Ajithkumar-2011CS030004
A.Asmitha -2011CS030006
G.Mounika -2011CS030060

Under the esteemed guidance of

Ms. Divya Maheshwari

(Assistant Professor)

Department of Computer Science & Engineering (Data Science)

Malla Reddy University

Maisammaguda, Dulapally,
Hyderabad, Telangana 500100
www.mallareddyuniversity.ac.in
2023-24

CERTIFICATE

This is to certify that this bonafide record of the application development entitled **“GrowGuardian-AI For Farming”** submitted by Mr.A.Ajithkumar(2011CS030004), Ms.A.Asmitha(2011CS030006), Ms.G.Mounika(2011CS030060) of IV year I sem to the Malla Reddy University, Hyderabad.This bonafide record of work carried out by us under the guidance of our supervision.The contents of this report, in full or in parts, have not been submitted to any other Organization for the award of any Degree.

INTERNAL GUIDE	**HEAD OF THE DEPARTMENT**
Ms.Divya Maheshwari	**Dr.GS NaveenKumar**
Assistant Professor	**CSE (Data Science)**

External Examiner

Date:

ABSTRACT

Since there have been climate changes which have resulted in an increasing amount of unexpected rainfalls, par below temperatures and heatwaves in the region, which result in significant loss of ecosystem. Deep learning has helped develop various utility tools to tackle world problems. This problem of agriculture can be solved by using various DL algorithms. This project aims to create two things - a)A crop recommendation system and b) a Plant disease identification system embedded into a single website. The datasets were publicly available over the internet. Once the features for task one are extracted, then the dataset is trained on five different algorithms - logistic regression, decision tree, support vector machine(SVM), multilayer perceptron and random forest. For the second task, three CNN architectures, VGG16, ResNet50 and EfficientNetV2, are trained, and a comparative study is done between them. For the task one, random forest achieved an accuracy of 99.31%, and for the second task, EfficientNetV2 achieved an accuracy of 96.06%.

CONTENTS

Certificate		I
Abstract		II
Contents		III
List of Tables		IV
List of Figures		V
Abbreviations		VI
Chapter 1	Introduction	1
	1.1 Introduction to Climate-Driven Agricultural Challenges	1
	1.2 The Role of DeepLearning in Agricultural Solutions	1
	1.3 Bridging the Gap with a Unified Website	2
Chapter 2	Review of Relevant Literature	3
Chapter 3	Methodology	5
	3.1 Project Objectives	5
	3.2 System Architecture	5
	3.3 Datasets	6
	3.4 Crop Recommendation	7
	3.5 Plant Disease Identification	13
	3.6 Tools and Technologies	16
Chapter 4	Results and Discussions	18
	4.1 Model Results	18
	4.2 Deployed website	20
Chapter 5	Conclusion and Future Scope of Study	23
References		24
Appendix A	:Sample Source Code	25

LIST OF TABLES

Table	Title	Page
4.1	Crop recommendation task accuracy over various algorithm	18
4.2	Plant Disease Classification task accuracy over various architectures.	20

LIST OF FIGURES

Figure	Title	Page
3.1	System Architecture	6
3.2	Random data from the dataset for crop recommendation	7
3.3	Sigmoid Function	8
3.4	Decision Tree	10
3.5	Support Vector Machine	11
3.6	MLP	12
3.7	Random Forest	13
3.8	VGG 16 architecture	14
3.9	ResNet50 architecture	15
3.10	Efficient Net architecture	16
4.1	Histogram representation of the table 4.1	18
4.2	Accuracy and Loss graphs of the VGG model	19
4.3	Accuracy and Loss graphs of the ResNet50 model	19
4.4	Accuracy and Loss graphs of the EfficientNetV2S model	19
4.5	Landing page of the website	20
4.6	Plant disease detection landing page	21
4.7	Result of the plant disease detection	21
4.8	Crop recommendation system landing page	22
4.9	Result of crop recommendation	22

LIST OF ABBREVIATIONS

ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
VGG	Visual Geometry Group
ResNet	Residual Neural Network

CHAPTER – 1

INTRODUCTION

1.1 Introduction to Climate-Driven Agricultural Challenges

In recent years, the world has witnessed unprecedented changes in climate patterns, giving rise to extreme weather events such as unexpected rainfalls, below-average temperatures, and heatwaves. These climatic shifts pose significant challenges to agricultural ecosystems, leading to adverse impacts on crop yields and plant health. In response to this pressing issue, there is a critical need for innovative solutions that harness the power of technology to address the complex interplay between climate change and agriculture.

1.2 The Role of DeepLearning in Agricultural Solutions

Machine learning (ML) and Deep Learning (DL) has emerged as powerful tools in developing practical solutions to global challenges. Leveraging this potential, a comprehensive project has been undertaken to tackle the adverse effects of climate change on agriculture. The project comprises two major components: a Crop Recommendation System and a Plant Disease Identification System, both seamlessly integrated into a single website. By utilizing publicly available datasets, this initiative aims to harness the predictive capabilities of ML algorithms to enhance agricultural practices and mitigate the impact of climate-induced changes.

a) Crop Recommendation System

The first facet of the project focuses on developing a robust Crop Recommendation System. This system aims to assist farmers in making informed decisions about suitable crops based on the prevailing climatic conditions. The dataset, sourced from the internet, undergoes feature extraction to create a foundation for model training. Five diverse ML algorithms - Logistic Regression, Decision Tree, Support Vector Machine (SVM), Multi-layer Perceptron, and Random Forest - are employed to predict optimal crop choices. Notably, the Random Forest algorithm achieves an impressive accuracy of 99.31%, demonstrating its efficacy in providing accurate crop recommendations amidst changing climatic conditions.

b) Plant Disease Identification System

The second dimension of the project is dedicated to addressing the impact of climate change on plant health through a Plant Disease Identification System. Leveraging the power of

Convolutional Neural Networks (CNNs), specifically VGG16, ResNet50, and EfficientNetV2 architectures, the system can accurately identify and diagnose plant diseases. By training on a dataset of plant images, the models exhibit remarkable capabilities in distinguishing between healthy and diseased plants. Notably, EfficientNetV2 emerges as a standout performer, achieving an impressive accuracy of 96.06%. A comprehensive comparative study between the three CNN architectures sheds light on their respective strengths and weaknesses.

1.3 Bridging the Gap with a Unified Website

By integrating these two systems into a cohesive and user-friendly website, this project aims to provide farmers and agricultural stakeholders with a centralized platform for informed decision-making. The website serves as a dynamic resource, offering crop recommendations tailored to the changing climate and enabling early detection and management of plant diseases. Through the seamless synergy of ML algorithms and climate data, this initiative represents a significant stride towards sustainable and resilient agriculture in the face of evolving environmental challenges. As climate change continues to impact global ecosystems, the fusion of technology and agriculture becomes increasingly vital, and this project stands at the forefront of innovative solutions to address these challenges.

CHAPTER - 2

REVIEW OF RELEVENT LITERATURE

There are many attempts made to tackle the problem of crop recommendation and plant disease classification. G Chauhan and A. Chaudhary proposed the ways to recommend crops based on soil type and used random forest and decision trees to make the prediction. This showed that the task of predicting crops based on land patterns would be helpful via a random forest classifier.

For plant disease detection, S.P. Mohanty has an open-source dataset of plant leaf images along with their grayscale and segmented part. The paper published by him talks about the classification task done by using AlexNet. This paper talks about using the above-mentioned dataset on various DL classification models, and Inspired by this state of the art result, and I decided to use the CNN approach for plant disease detection.

In the paper ―Deep learning for Image-Based Plant detection" [1] the authors Prasanna Mohanty et al., has proposed an approach to detect disease in plants by training a convolutional neural network. The CNN model is trained to identify healthy and diseased plants of 14 species. The model achieved an accuracy of 99.35% on test set data. When using the model on images procured from trusted online sources, the model achieves an accuracy of 31.4%, while this is better than a simple model of random selection, a more diverse set of training data can aid to increase the accuracy. Also some other variations of model or neural network training may yield higher accuracy, thus paving path for making plant disease detection easily available to everyone.

Malvika Ranjan et al. in the paper ―Detection and Classification of leaf disease using Artificial Neural Network" proposed an approach to detect diseases in plant utilizing the captured image of the diseased leaf. Artificial Neural Network (ANN) is trained by properly choosing feature values to distinguish diseased plants and healthy samples. The ANN model achieves an accuracy of 80%.

According to paper ―Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features" [3] by S. Arivazhagan, disease identification process includes four main steps as follows: first, a color transformation structure is taken for the input RGB image, and then by means of a specific threshold value, the green pixels are detected and uninvolved, which is followed by segmentation process, and for obtaining beneficial segments the texture statistics are computed. At last, classifier is used for the features that are extracted to classify the disease..

Kulkarni et al. in the paper ―Applying image processing technique to detect plant diseases" [4], a methodology for early and accurately plant diseases detection, using artificial neural network (ANN) and diverse image processing techniques. As the proposed approach is based on ANN classifier for classification and Gabor filter for feature extraction, it gives better results with a recognition rate of up to 91%.

In paper ―Plant disease detection using CNN and GAN" [5], by Emaneul Cortes, an approach to detect plant disease using Generative Adversarial networks has been proposed. Background segmentation is used for ensuring proper feature extraction and output mapping. It is seen that using Gans may hold promise to classify diseases in plants, however segmenting based on background did not improve accuracy.

In the paper ―Convolutional Neural Network based Inception v3 Model for Animal Classification [6], Jyotsna Bankar et al. have proposed use of inception v3 model in classifying animals in different species. Inception v3 can be used to classify objects as well as to categorize them, this capability of inception v3 makes it instrumental in various image classifiers.

CHAPTER – 3

METHODOLOGY

3.1 Project Objectives

The project is divided into 3 parts crop recommendation, disease identification and deployment. The first two tasks are related to creating ML and DL models with higher accuracy and the last task is related to creating a web application using flask and also create a neat and robust system design which will help us to serve larger requests over the network.

3.2 System Architecture

The overall system can be divided into two parts a) Web server and b) a ML con- tainer. The whole web server can be connected to a database most probably a SQL database. The architecture is designed in a way to allow large number of requests. It is easily scalable and deployable.

The technologies used for this whole project are:

- Docker: Docker helps in containarization process. It creates a whole development environment which is easy to use and easily deployable across any server
- NginX: Nginx is used to create a load balancer. A load balancer help distrubute the traffic among multiple available servers using different algorithms.
- Tensorflow: It is an open-source machine learning framework provided by Google.
- Keras: Keras is a deep learning library which provides powerful deep learning solutions
- MySQL: A relational database system
- Flask: A light-weight python web server

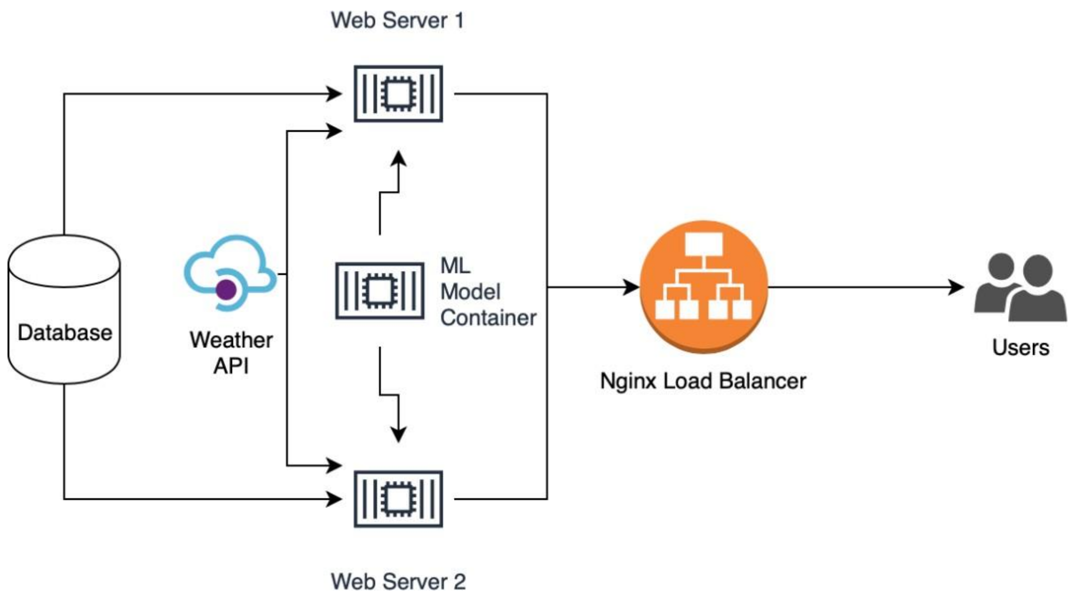


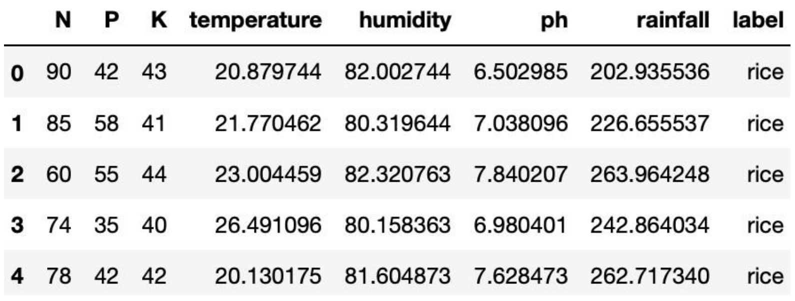
Fig:3.1 System Architecture

3.3 Datasets

The datasets for both the tasks were available online and were open sourced under open license to be used for anybody. The dataset for the first task consists of aCSV file which contains 2200 entries of the various factors such as soil condition,temperature, pH, humidity and rainfall and label output as the type of crop wellproduced in that type of conditions.

There are in total 22 types of crops available in the dataset and they are ’rice’ ’maize’, ’chickpea’, ’kidneybeans’, ’pigeonpeas’, ’mothbeans’, ’mungbean’, ’black-gram’, ’lentil’, ’pomegranate’, ’banana’, ’mango’, ’grapes’, ’watermelon’, ’muskmelon’,’apple’, ’orange’, ’papaya’, ’coconut’, ’cotton’, ’jute’ and ’coffee’.
The second dataset consists of 70,000 plant images having various diseases. It wasa 5GB data all of resolution 256x256. There are in total 38 classes available where there are 14 different plants and 26 diseases to be identified.

These datasets were used to train all the further mentioned ML algorithms and theone with best accuracy was chosen.



	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

Fig:3.2 Random data from the dataset for crop recommendation

3.4 Crop Recommendation

The crop recommendation is basically a classification task. Standard ML algorithms were used to classify various plants. The features trained for the classification were the NPK value of the soil, temperature of the surroundings, pH of the soil and the rainfall in a particular area.

This dataset was trained on the following algorithms:

- Logistic Regression
- Decision Tree
- Support Vector Machine (SVM)
- Multi Layer Perceptron
- Random Forest

- ## LOGISTIC REGRESSION

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning techniqueLogistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

Logistic Function (Sigmoid Function):

The sigmoid function is a mathematical function used to map the predicted values to probabilities.

It maps any real value into another value within a range of 0 and 1.

The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid functionor the logistic function.

In logistic regression, we use the concept of the threshold value, which defines the probabilityof either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

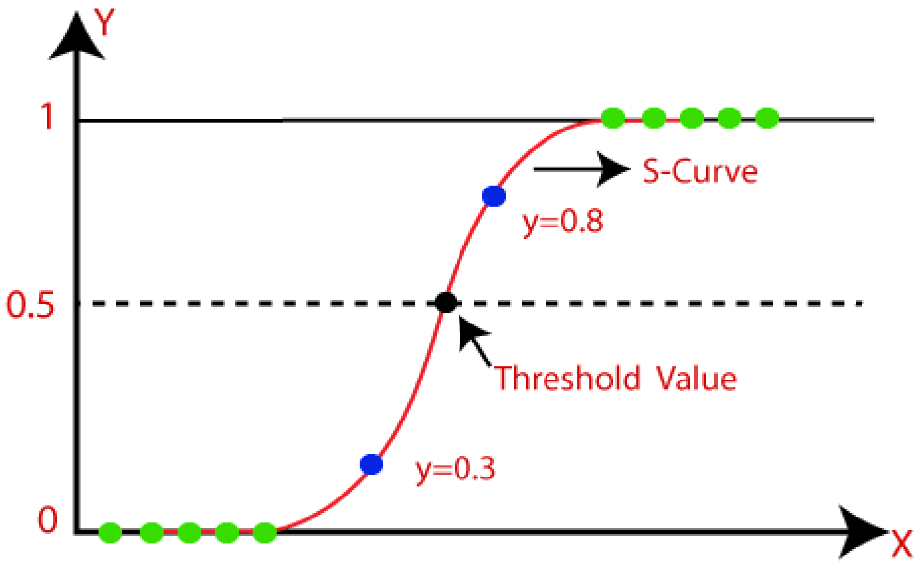


Fig:3.3 Sigmoid Function

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. Themathematical steps to get Logistic Regression equations are given below:

We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

In Logistic Regression y can be between 0 and 1 only, so for this let's divide the aboveequation by (1-y):

$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

But we need range between -[infinity] to +[infinity], then take logarithm of the equation itwill become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

DECISION TREE

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

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decisionbased on given conditions**.**

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the CART algorithm**,** which stands for Classification and Regression Tree algorithm.

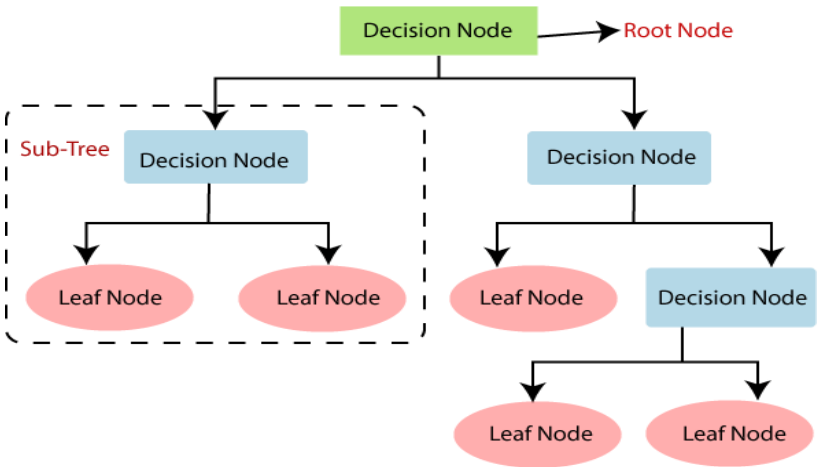


Fig:3.4 Decision Tree

Decision Trees usually mimic human thinking ability while making a decision, so it is easy tounderstand.

The logic behind the decision tree can be easily understood because it shows a tree-likes tructure.

How does the Decision Tree algorithm Work?

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in **Step -6**. Continue this process until a stage is reached where you cannot further classify the nodes.

SUPPORT VECTOR MACHINE

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

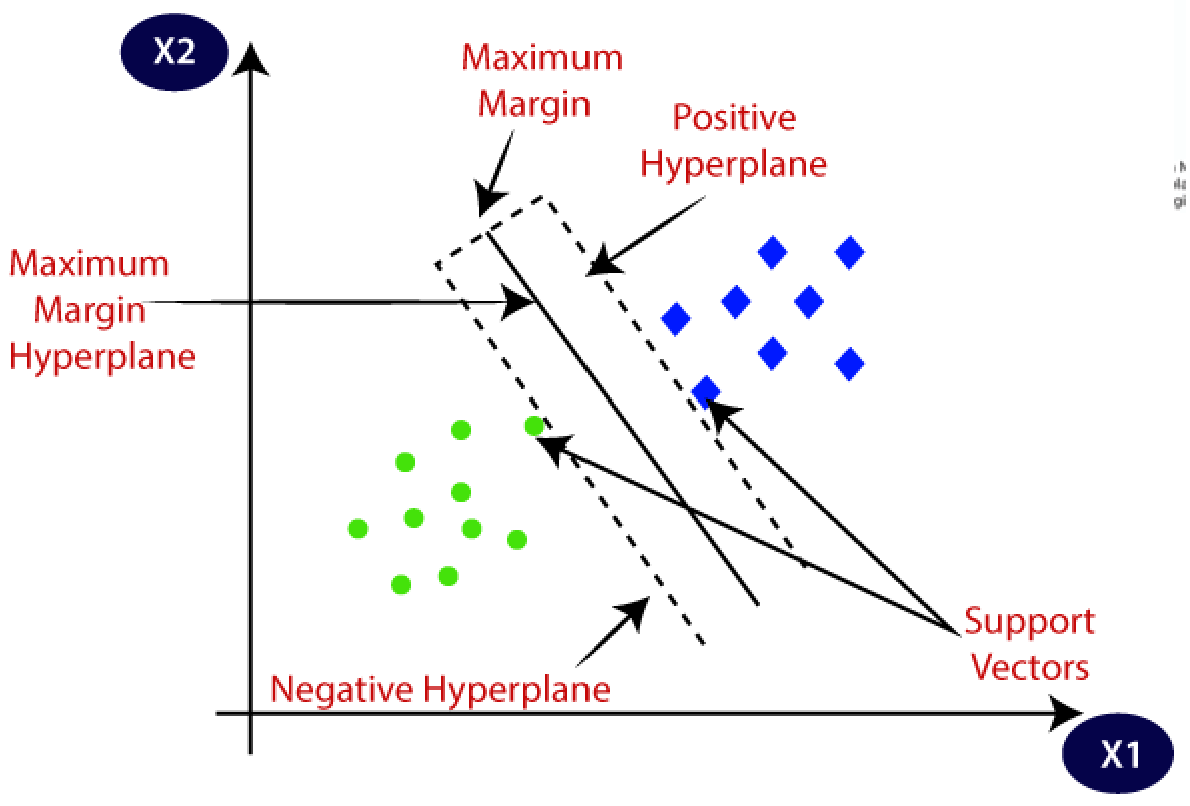


Fig:3.5 Support Vector Machine

MULTI-LAYER PERCEPTRON

Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension. A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.
A multi-layer perceptron has one input layer and for each input, there is one neuron(or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes. A schematic diagram of a Multi-Layer Perceptron (MLP) is depicted below.

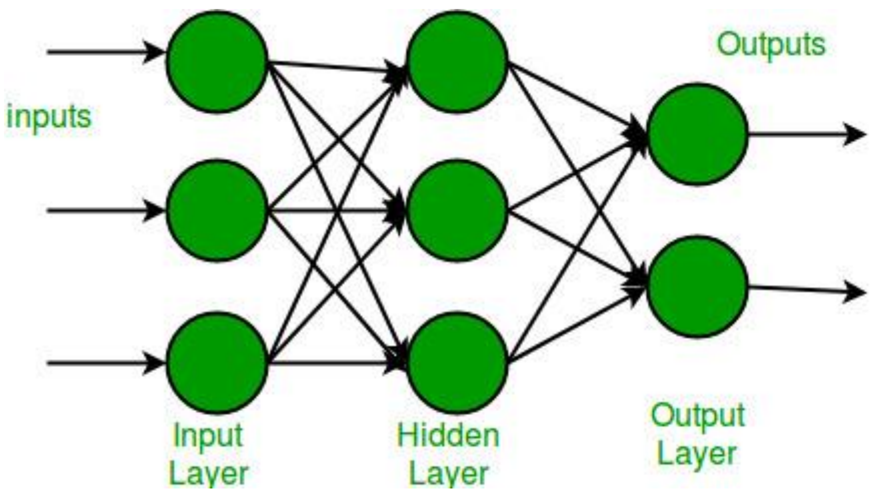


Fig:3.6 MLP

RANDOM FOREST

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the

random forest takes the prediction from each tree and based on the majority votes of predictions, andit predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem ofoverfitting.

The below diagram explains the working of the Random Forest algorithm:

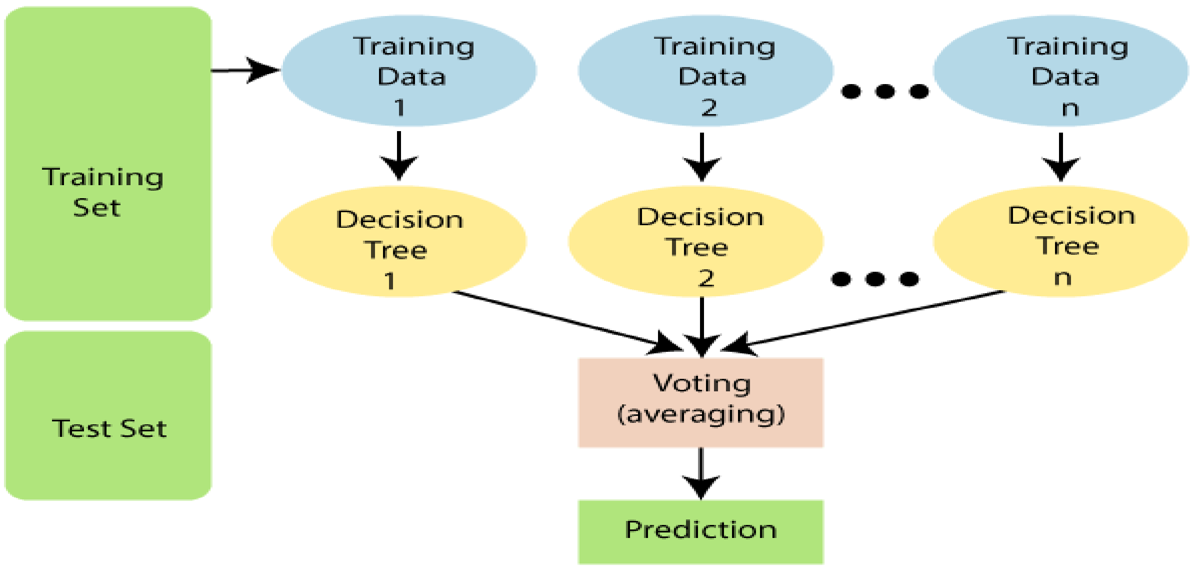


Fig:3.7 Random Forest

How does Random Forest algorithm work?

Random Forest works in two-phase first is to create the random forest by combining Ndecision tree, and second is to make predictions for each tree created in the first phase.

The Working process can be explained in the below steps and diagram:

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: Repeat Step 1 & 2.

Step-5: For new data points, find the predictions of each decision tree, and assign the newdata points to the category that wins the majority votes.

3.5 Plant Disease Identification

The dataset available for this task is a collection of 70,000 images in total. Image classification is a hard task for normal ML algorithms since the feature detectionis not easy. Hence for major image classification problems convolutional neural networks(CNNs) are used which can detect features while training and provide better accuracy over the traditional ML algorithms.

There are 3 different state-of-the-art CNN architecture available for the image classification task.

- VGG16
- ResNet50
- EfficientNet

While handling image data CNNs are the best type of approach to perform classification. The below mentioned models are used in plant disease detection task.

VGG16

The VGG-16 is one of the most popular pre-trained models for image classification. Introduced in the famous ILSVRC 2014 Conference, it was and remains THE model to beat even today. Developed at the Visual Graphics Group at the University of Oxford, VGG-16 beat the then standard of AlexNet and was quickly adopted by researchers and the industry for their image Classification Tasks. Figure:5 shows the VGG16 architecture.

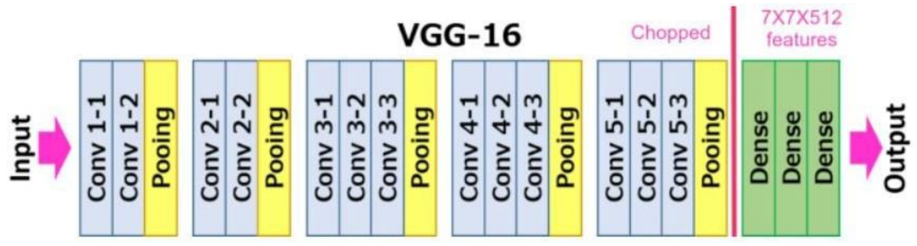


Fig:3.8 VGG 16 architecture

ResNet50

ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has 3.8 x $10\hat{9}$ Floating points operations. It is a widely used ResNet model and we have explored ResNet50 architecture in depth.This architecture can be used on computer vision tasks such as image classififcation,object localisation, object detection.

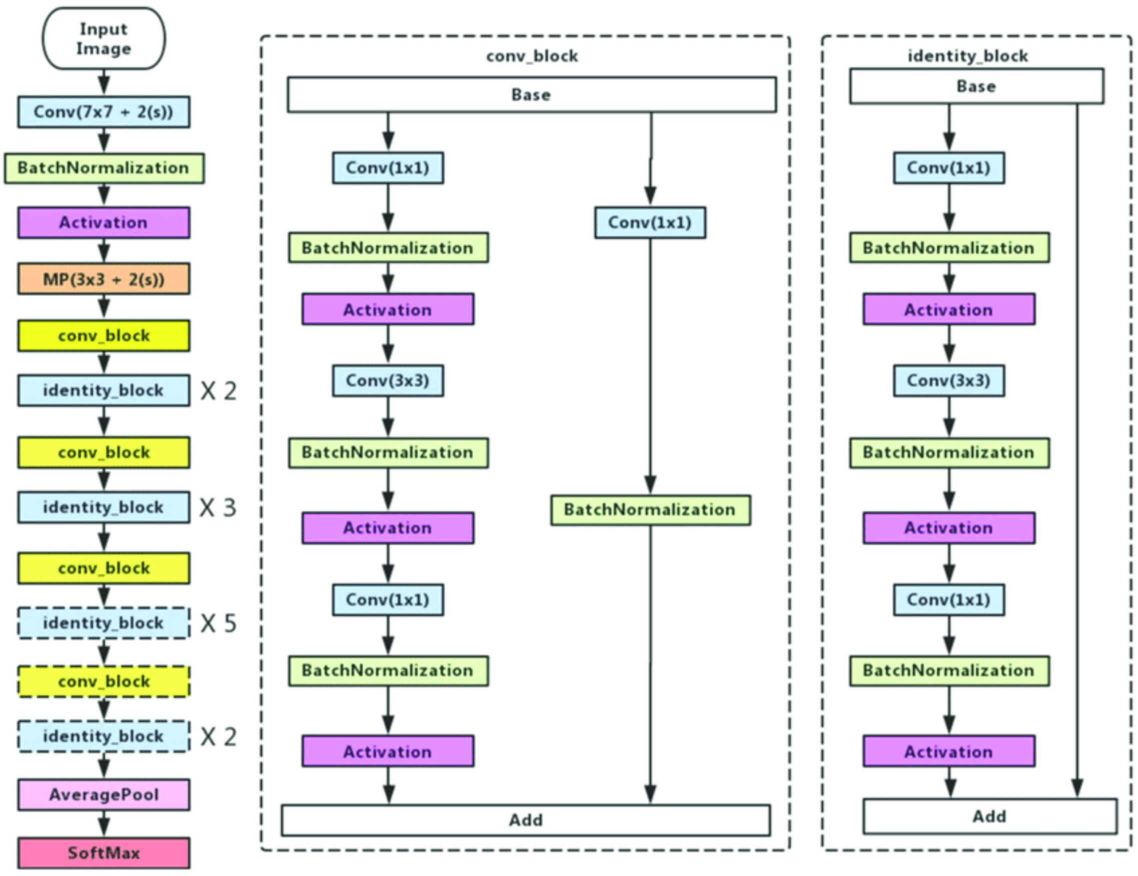


Fig:3.9 ResNet50 architecture

EfficientNet

EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. The compound scaling method is justified by the intuition that if the input image is bigger, then the network needs more layers to increase the receptive fieldand more channels to capture more fine-grained patterns on the bigger image. The base EfficientNet-B0 network is based on the inverted bottleneck residual blocks of MobileNetV2, in addition to squeeze-and-excitation blocks. In this project Ef- ficientV2S is used which the current SOTA of Google on the ImageNet dataset.

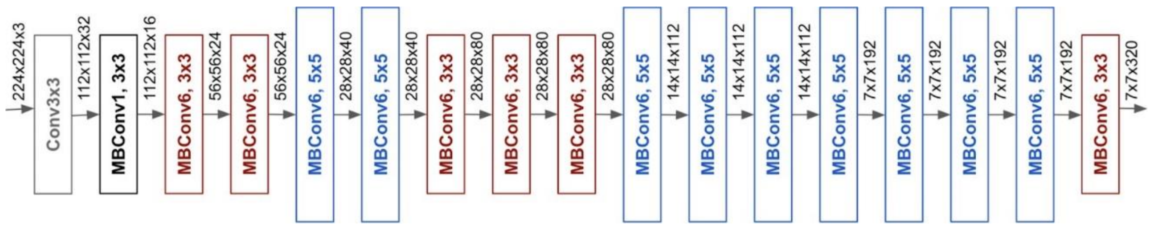


Fig:3.10 EfficientNet architecture

3.6 Tools and Technologies

As the project had two parts, web and ML two spectrums of tools are been used. Those are listed below:

Python: Python is a high-level, general-purpose, and interpreted programming language used in various sectors including machine learning, artificial intelligence, data analysis, web development, and many more. Python is known for its ease of use, powerful standard library, and dynamic semantics. It also has a large community of developers who keep on contributing towards its growth. The major focus behind creating it is making it easier for developers to read and understand, also reducing the lines of code.

Tensor Flow: TensorFlow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

Keras: Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. It was developed by one of the Google engineers, Francois Chollet. It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks. It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.

Flask: Flask is often referred to as a microframework. It is designed to keep the core of the application simple and scalable.Instead of an abstraction layer for database support, Flask supports extensions to add such capabilities to the application.

Docker: Docker is an open-source containerization platform by which you can pack your application and all its dependencies into a standardized unit called a container. Containers are light in weight which makes them portable and they are isolated from the underlying infrastructure and from each other container.

Nginx: Nginx is a dedicated web server that has solved efficiency issues and provided us with an optimum way to handle 1000s of requests concurrently. Web server for reverse proxy, caching, and load balancing.

HTML: HTML is an acronym which stands for Hyper Text Markup Language which is used for creating web pages and web applications.

CSS: CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

JavaScript : JavaScript (JS) is the most popular lightweight, interpreted compiled programming language. It can be used for both Client-side as well as Server-side developments. JavaScript also known as a scripting language for web page.

CHAPTER-4

RESULTS AND DISCUSSIONS

4.1 Model Results

For the first task, 5 classification algorithms were taken and then the accuracy overeach of them were measured. For the task 2 the best image classification CNN architectures were trained. VGG16, ResNet50 and EfficientNetv2 are trained. The tables below show the results 2.

Table:4.1 Crop recommendation task accuracy over various algorithm

Algorithm	Accuracy	Precision	Recall	F1-Score
Logistic Regression	94.54	0.95	0.95	0.94
Decision Tree	97.72	0.98	0.98	0.98
SVM	9.09	0.59	0.09	0.11
Multi-Layer Perceptron	95.22	0.96	0.95	0.95
Random Forest	99.31	0.99	0.99	0.99

This table concludes that the random forest outperforms every other algorithm achieving 99.92% accuracy.

A histogram representation of the accuracy vs models is also shown for the task 1of the project.

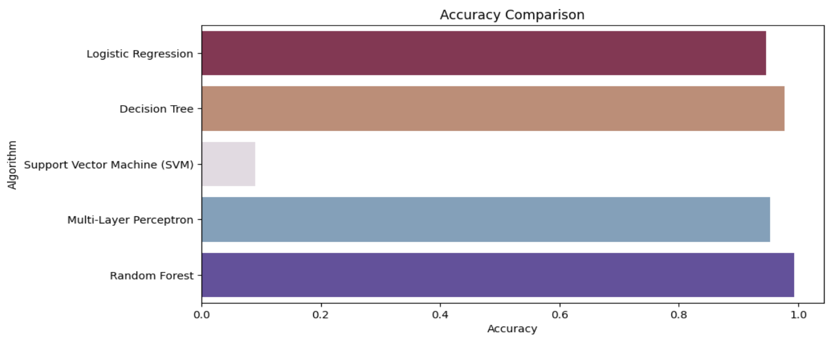


Fig:4.1 Histogram representation of the table 4.1

For the task 2 i.e. the plant disease identification task three models are trained VGG16, ResNet50 and EfficientNetV2S. The accuracy and loss function graph overvarious epochs of both the models is shown below.

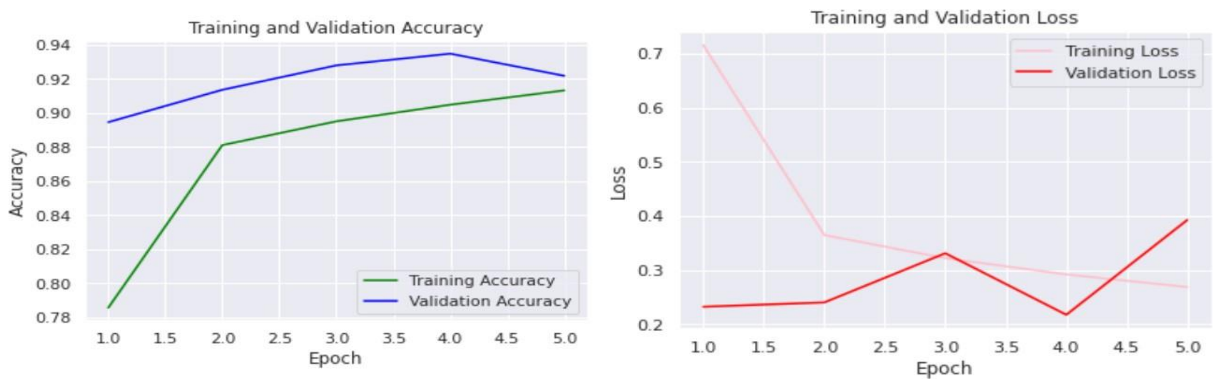


Fig: 4.2 Accuracy and Loss graphs of the VGG model

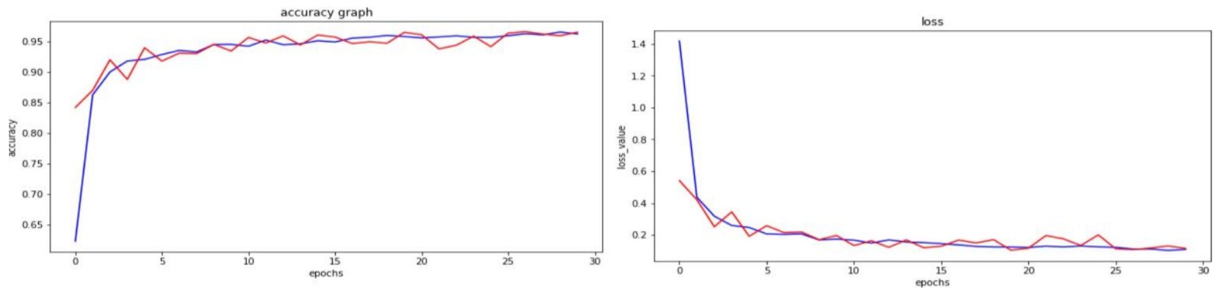


Fig:4.3 Accuracy and Loss graphs of the ResNet50 model

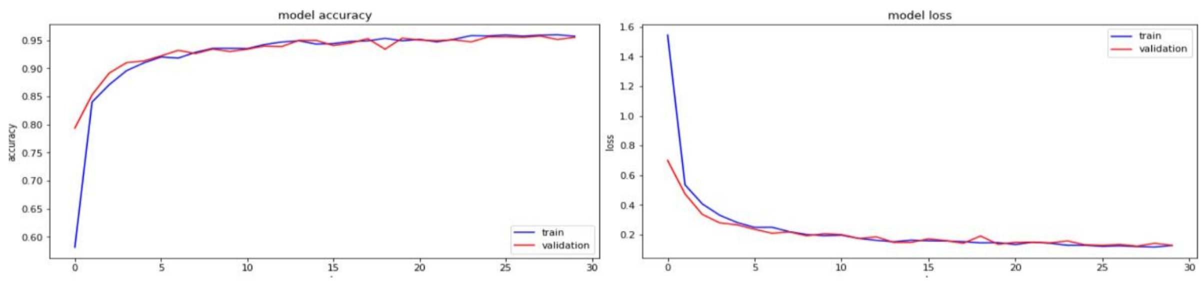


Fig:4.4 Accuracy and Loss graphs of the EfficientNetV2S model

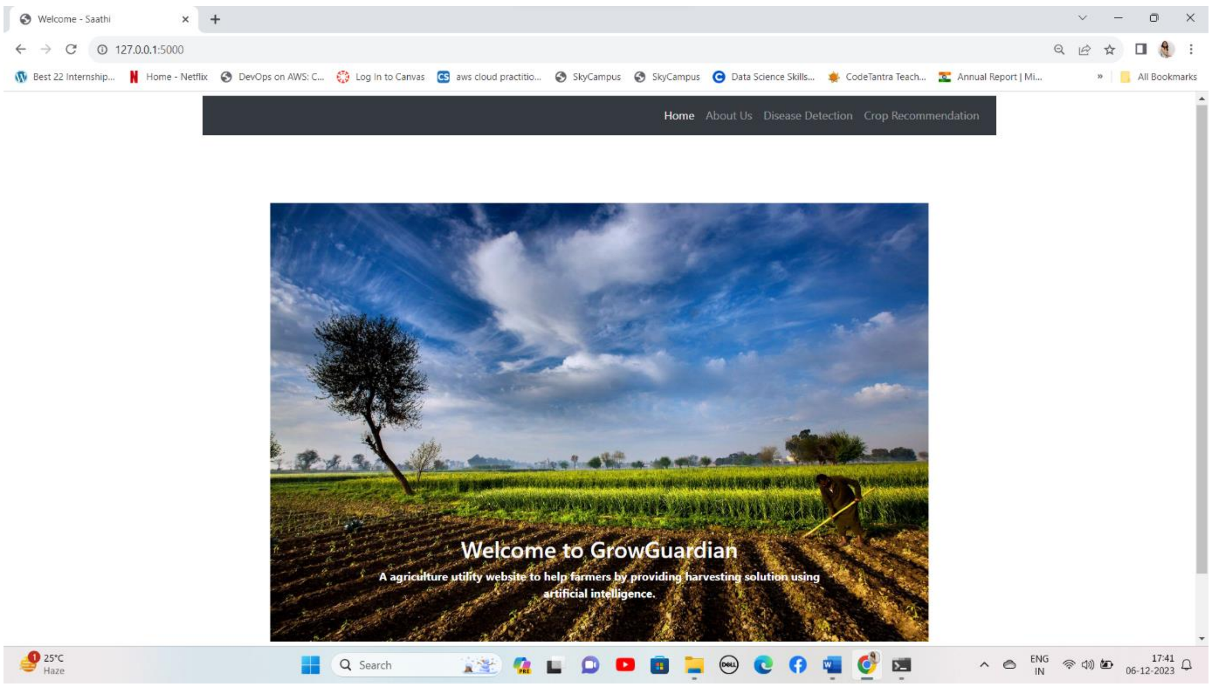
Table:4.2 Plant Disease Classification task accuracy over various architectures.

Architecture	Training Acc.	Validation Acc.	Test Accuracy.
VGG16	92.18	91.33	91.78
ResNet50	96.02	95.41	95.53
EfficientNetV2	96.06	95.53	95.83

This table concludes that EfficientNetV2 is efficient.

4.2 Deployed website

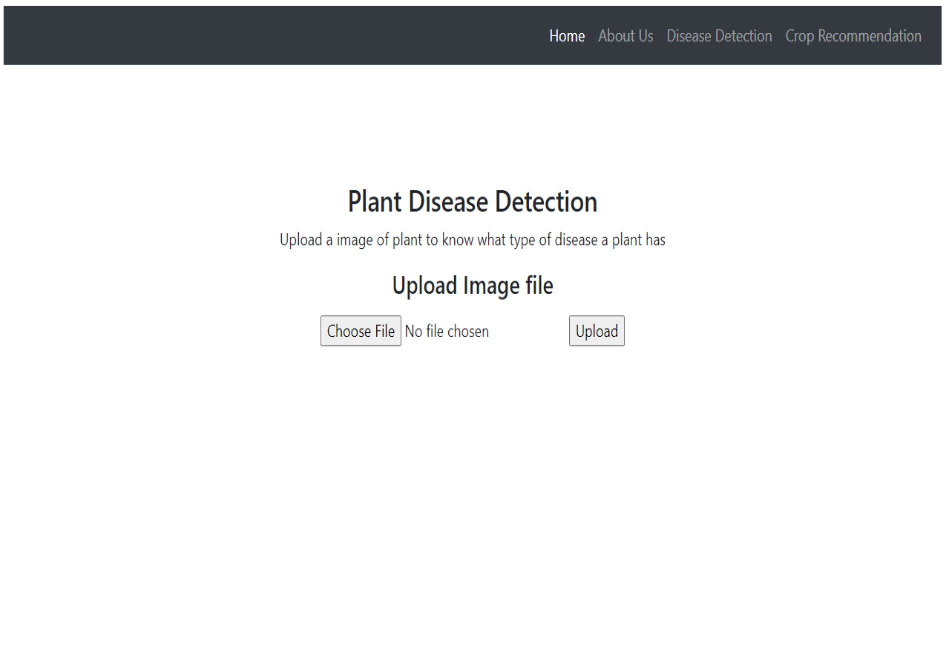
This section shows the various outputs of the deployed application. The web application is created in Flask and has a DL backend to it. It will later be deployed on cloud service platform.



Welcome to GrowGuardian

A agriculture utility website to help farmers by providing harvesting solution using artificial intelligence.

Fig:4.5 Landing page of the website

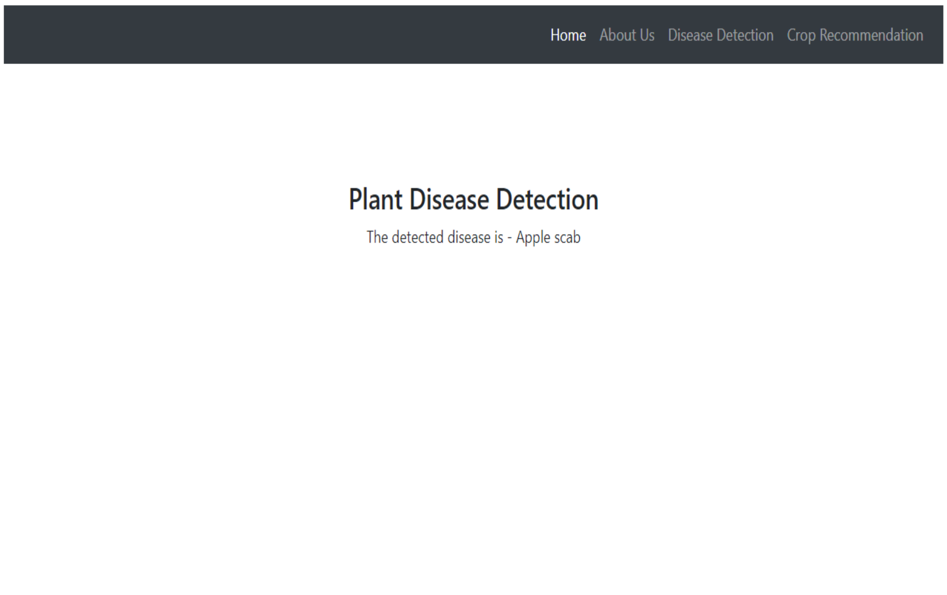


Plant Disease Detection

Upload a image of plant to know what type of disease a plant has

Upload Image file

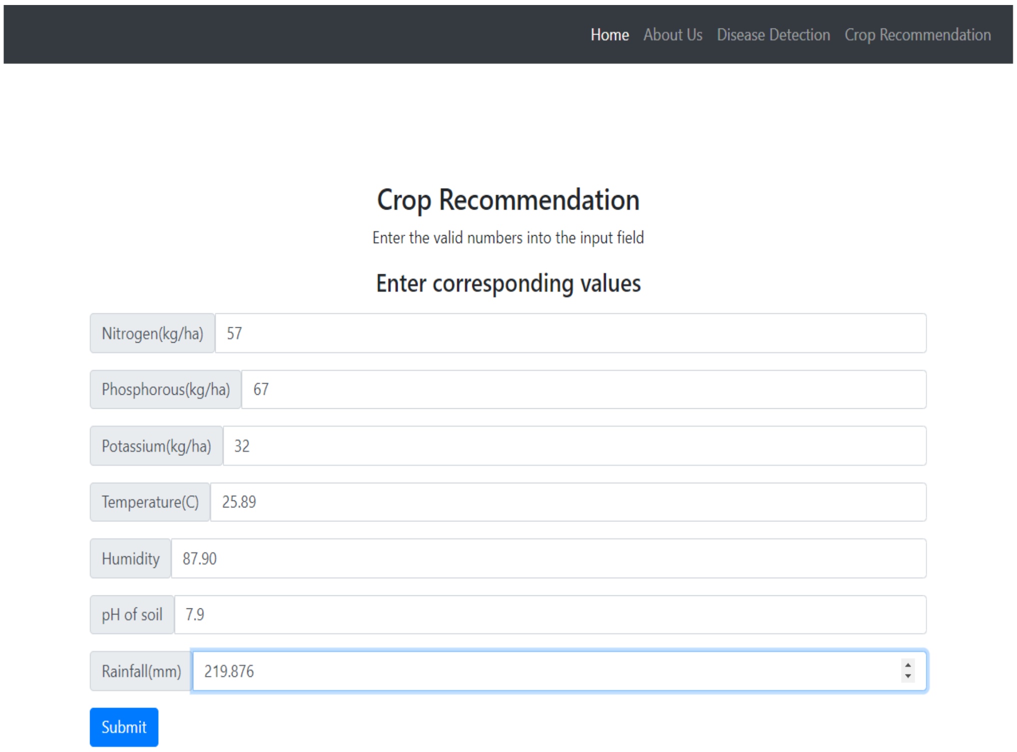
Fig:4.6 Plant disease detection landing page



Plant Disease Detection

The detected disease is - Apple scab

Fig:4.7 Result of the plant disease detection



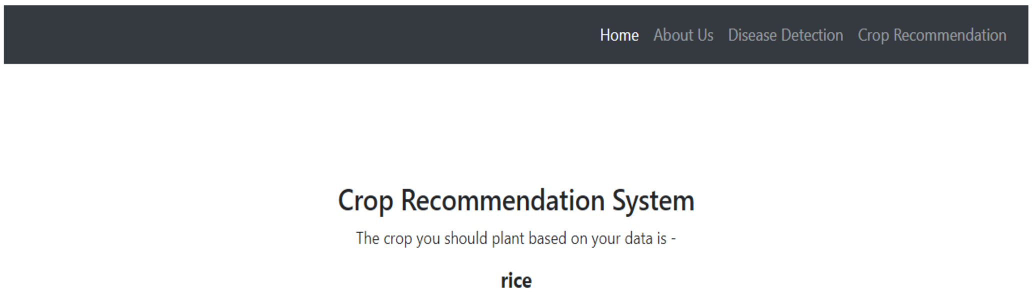
Crop Recommendation

Enter the valid numbers into the input field

Enter corresponding values

Nitrogen(kg/ha)	57
Phosphorous(kg/ha)	67
Potassium(kg/ha)	32
Temperature(C)	25.89
Humidity	87.90
pH of soil	7.9
Rainfall(mm)	219.876

Fig:4.8 Crop recommendation system landing page



Crop Recommendation System

The crop you should plant based on your data is -

rice

Fig:4.9 Result of crop recommendation

CHAPTER – 5

CONCLUSION AND FUTURE SCOPE OF STUDY

CONCLUSION

In conclusion, this project represents a pioneering effort in leveraging machine learning to address the profound challenges posed by climate change on agriculture. The escalating frequency of unexpected rainfalls, temperature fluctuations, and heatwaves has led to substantial ecosystem losses. The integration of advanced technologies into agricultural practices emerges as a beacon of hope in mitigating these adverse effects.

The first aspect of the project, the Crop Recommendation System, harnesses the predictive capabilities of five distinct machine learning algorithms. Notably, the Random Forest algorithm outshines others, achieving an exceptional accuracy of 99.31%. This accomplishment underscores the system's effectiveness in providing farmers with tailored crop recommendations, promoting resilient agricultural practices in the face of unpredictable climate patterns.

Simultaneously, the Plant Disease Identification System, employing three state-of-the-art CNN architectures, addresses the critical issue of declining plant health. EfficientNetV2 emerges as a standout performer with an accuracy of 96.06%. The comparative study conducted among VGG16, ResNet50, and EfficientNetV2 sheds light on their respective strengths, guiding future implementations in plant disease identification.

By seamlessly integrating these systems into a unified website, this project provides a comprehensive resource for farmers and stakeholders. The website serves as a dynamic tool for decision-making, offering real-time insights into optimal crop choices and early detection of plant diseases. As climate change continues to intensify, this initiative exemplifies the transformative potential of machine learning in fostering sustainable and adaptive agricultural practices, offering a glimpse into a future where technology becomes a vital ally in securing global food security amidst evolving environmental challenges.

FUTURE SCOPE

CNN architectures have been replaced by Transformer Networks. Models like ViT, CoAtNet which uses transformer neural networks which are the current state-of- the-art image classification neural nets. If these networks are used accuracy can be boosted by 2 to 3 %. Also in terms of datasets, for the crop recommendationtask, more data from different regions should be collected to improve the accuracy further. For plant disease classification more classes should be covered in terms of diseases as well as the plants.

For the deployment of the project, as currently there is no efficient load balancing solution given to the deployed website. Efficient load balancing algorithms can beused to properly distribute the traffic over available servers. The frontend can be improved for better user experience using modern web frameworks such as ReactJSor VueJS.

REFERENCES

[1] Kulkarni, P., Karwande, A., Kolhe, T., Kamble, S., Joshi, A. and Wyawahare, M., 2021. Plant Disease Detection Using Image Processing and Machine Learn-ing. arXiv preprint arXiv:2106.10698.

[2] G. Chauhan and A. Chaudhary, ”Crop Recommendation System using Machine Learning Algorithms,” 2021 10th International Conference on System Model- ing & Advancement in Research Trends (SMART), 2021, pp. 109-112, doi: 10.1109/SMART52563.2021.9676210.

[3] Sharada Prasanna Mohanty, David Hughes, Marcel Salathe, 2016, Using Deep Learning for Image-Based Plant Disease Detection. arXiv preprint arXiv:2106.10698.

[4] Hassan SM, Maji AK, Jasiński M, Leonowicz Z, Jasińska E. Identification of Plant-Leaf Diseases Using CNN and Transfer-Learning Approach. Electronics.2021; 10(12):1388

[5] Kumar S, Kaur R. Plant disease detection using image processing—a review. Int J Comput Appl. 2015;124(2):6–9.

APPENDIX A

Sample Source Code

recommender-sys.ipynb

```
import pandas as pd
import matplotlib.pyplot as plt
import pickle
import numpy as np
from sklearn.model_selection import train_test_split
import sklearn.metrics as metrics
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
import seaborn as sns
PATH = './datasets/Crop_recommendation.csv'
MODELS = './recommender-models/'
data = pd.read_csv(PATH)
data.head()
data.info()
data.columns
data['label'].unique()
data['label'].value_counts()
features = data[['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']]
labels = data['label']
# Splitting into the training and test dataset

# Train:Test = 4:1

X_train, X_test, Y_train, Y_test = train_test_split(features, labels,
test_size=0.2, random_state=42)
acc = []
models_list = []
RF = RandomForestClassifier(n_estimators=20, random_state=0)
RF.fit(X_train,Y_train)

predicted_values = RF.predict(X_test)

accuracy = metrics.accuracy_score(Y_test, predicted_values)

acc.append(accuracy)
models_list.append('Random Forest')
print("RF's Accuracy is: ", accuracy)
print(metrics.classification_report(Y_test,predicted_values))
filename = 'RF.pkl'
pickle.dump(LogReg, open(MODELS + filename, 'wb'))
plt.figure(figsize=[10,5],dpi = 100)
plt.title('Accuracy Comparison')
plt.xlabel('Accuracy')
plt.ylabel('Algorithm')
sns.barplot(x = acc,y = models_list,palette='twilight_shifted_r')
```

EfficientNetV2S.ipynb

```
from google.colab import drive
drive.mount("/content/drive", force_remount=True)

!unzip -q '/content/drive/MyDrive/mp_dataset/new-plant-diseases-dataset.zip' -
d '/content'
# drive.flush_and_unmount()
data_dir = '/content/New Plant Diseases Dataset(Augmented)/New Plant Diseases
Dataset(Augmented)'
train_dir = data_dir + '/train'
valid_dir = data_dir + '/valid'
test_dir = '/content/test_files'
import tensorflow
from tensorflow import keras
from tensorflow.keras.applications.efficientnet_v2 import EfficientNetV2S
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
from tensorflow.keras.applications.efficientnet_v2 import preprocess_input,
decode_predictions
from keras.utils.np_utils import to_categorical
import numpy as np
import os, shutil
from keras.models import Sequential,load_model,Model
from keras.layers import
Conv2D,MaxPool2D,AveragePooling2D,Dense,Flatten,ZeroPadding2D,BatchNormalizati
on,Activation,Add,Input,Dropout,GlobalAveragePooling2D
from keras.initializers import glorot_uniform
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint,EarlyStopping,ReduceLROnPlateau
newpath = '/content/test_files'
if not os.path.exists(newpath):
    os.makedirs(newpath)

x = os.listdir(train_dir)
for dir_names in x:
  # print(dir_names)
  p = os.listdir(os.path.join(train_dir, dir_names))
  if not os.path.exists(os.path.join(newpath, dir_names)):
    os.makedirs(os.path.join(newpath, dir_names))
  cnt = 0
  file_names = []
  for image_names in p:
    if cnt==250:
      break
    file_names.append(train_dir+'/'+dir_names+'/'+image_names)
    cnt += 1
  for f in file_names:
    shutil.move(f, os.path.join(newpath, dir_names))
    train_datagen = 
ImageDataGenerator(shear_range=0.2,zoom_range=0.2,horizontal_flip=False,vertic
al_flip=False
                                  ,fill_mode='nearest',width_shift_range=0.2,h
eight_shift_range=0.2)

val_datagen = ImageDataGenerator()

train = 
train_datagen.flow_from_directory(directory=train_dir,batch_size=32,target_siz
e=(224,224),
                                         color_mode='rgb',class_mode='categori
cal',seed=42)

valid = 
val_datagen.flow_from_directory(directory=valid_dir,batch_size=32,target_size=
(224,224),color_mode='rgb',class_mode='categorical')

test_datagen = ImageDataGenerator()

test = 
test_datagen.flow_from_directory(directory=test_dir,batch_size=32,target_size=
(224,224),color_mode='rgb',class_mode='categorical')
base_model_tf=EfficientNetV2S(include_top=False,weights='imagenet',input_shape
=(224,224,3),classes=38)
base_model_tf.trainable=False

pt=Input(shape=(224,224,3))
func=tensorflow.cast(pt,tensorflow.float32)
x=preprocess_input(func) #This function used to zero-center each color channel
wrt Imagenet dataset
model_resnet=base_model_tf(x,training=False)
model_resnet=GlobalAveragePooling2D()(model_resnet)
model_resnet=Dense(128,activation='relu')(model_resnet)
model_resnet=Dense(64,activation='relu')(model_resnet)
model_resnet=Dense(38,activation='softmax')(model_resnet)


model_main=Model(inputs=pt,outputs=model_resnet)
model_main.summary()
es=EarlyStopping(monitor='accuracy',verbose=1,patience=5,mode='auto')
mc=ModelCheckpoint(filepath='/content',monitor='val_accuracy',verbose=1,save_b
est_only=True)
```

```
lr=ReduceLROnPlateau(monitor='val_accuracy',verbose=1,patience=3,min_lr=0.001)
model_main.compile(optimizer='Adam',loss='categorical_crossentropy',metrics=['
accuracy'])
model_main.fit(train,validation_data=valid,epochs=30,steps_per_epoch=200,verbo
se=1,callbacks=[es, lr])
model_main.save("efficientnetv2s_PLANT_DISEASE.h5")
import matplotlib.pyplot as plt
%matplotlib inline
import cv2
from PIL import Image
plt.figure(figsize=(10,5))
plt.plot(model_main.history.history['loss'],color='b',label='Training loss')
plt.plot(model_main.history.history['val_loss'],color='r',label='Validation
loss')
plt.xlabel("epochs")
plt.ylabel("loss")
plt.legend(['train', 'validation'], loc='upper right')
plt.title("model loss")
plt.figure(figsize=(10,5))
plt.plot(model_main.history.history['accuracy'],color='b',label='Training
accuracy')
plt.plot(model_main.history.history['val_accuracy'],color='r',label='Validatio
n accsuracy')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.legend(['train', 'validation'], loc='lower right')
plt.title("model accuracy")
loss, accuracy = model_main.evaluate(train, verbose=1)
print("Train: accuracy = %f  ;  loss = %f" % (accuracy, loss))

loss, accuracy = model_main.evaluate(valid, verbose=1)
print("Validation: accuracy = %f  ;  loss = %f" % (accuracy, loss))

loss, accuracy = model_main.evaluate(test, verbose=1)
print("Test: accuracy = %f  ;  loss = %f" % (accuracy, loss))
model = load_model("/content/efficientnetv2s_PLANT_DISEASE.h5")
img = tensorflow.keras.utils.load_img(
    "/content/test/test/Potato___Early_blight/PotatoEarlyBlight1.JPG",
    target_size=(224, 224, 3)
)
input_arr = tensorflow.keras.preprocessing.image.img_to_array(img)
input_arr = np.array([input_arr])
result = model.predict(input_arr)
probability_model = tensorflow.keras.Sequential([model,
                                         tensorflow.keras.layers.Softmax()])
predict = probability_model.predict(input_arr)
predict[0]
```

```
classes = ['Apple___Apple_scab', 'Apple___Black_rot',
'Apple___Cedar_apple_rust', 'Apple___healthy', 'Blueberry___healthy',
'Cherry_(including_sour)___Powdery_mildew',
'Cherry_(including_sour)___healthy', 'Corn_(maize)___Cercospora_leaf_spot
Gray_leaf_spot', 'Corn_(maize)___Common_rust_',
'Corn_(maize)___Northern_Leaf_Blight', 'Corn_(maize)___healthy',
'Grape___Black_rot', 'Grape___Esca_(Black_Measles)',
'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)', 'Grape___healthy',
'Orange___Haunglongbing_(Citrus_greening)', 'Peach___Bacterial_spot',
'Peach___healthy', 'Pepper,_bell___Bacterial_spot', 'Pepper,_bell___healthy',
'Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy',
'Raspberry___healthy', 'Soybean___healthy', 'Squash___Powdery_mildew',
'Strawberry___Leaf_scorch', 'Strawberry___healthy', 'Tomato___Bacterial_spot',
'Tomato___Early_blight', 'Tomato___Late_blight', 'Tomato___Leaf_Mold',
'Tomato___Septoria_leaf_spot', 'Tomato___Spider_mites Two-
spotted_spider_mite', 'Tomato___Target_Spot',
'Tomato___Tomato_Yellow_Leaf_Curl_Virus', 'Tomato___Tomato_mosaic_virus',
'Tomato___healthy']

p = np.argmax(predict[0])
print(classes[p])
```

setup.py

```
from flask import Flask, render_template, redirect, request, url_for, send_from_directory,
flash
from werkzeug.utils import secure_filename
import os
import tensorflow
import numpy as np
import pickle

app = Flask(__name__)

BASE_DIR = os.path.dirname(os.path.realpath(__file__))
MODEL_DIR = os.path.join(BASE_DIR, 'model')

UPLOAD_FOLDER = os.path.join(BASE_DIR, 'bucket')
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'JPG'}

MODEL = tensorflow.keras.models.load_model(os.path.join(MODEL_DIR,
'efficientnetv2s.h5'))
REC_MODEL = pickle.load(open(os.path.join(MODEL_DIR, 'RF.pkl'), 'rb'))

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

```
CLASSES = ['Apple scab', 'Apple Black rot', 'Apple Cedar apple rust', 'Apple healthy',
'Blueberry healthy', 'Cherry (including sour) Powdery mildew', 'Cherry (including sour)
healthy', 'Corn (maize) Cercospora leaf spot Gray leaf spot', 'Corn(maize) Common rust',
'Corn(maize) Northern Leaf Blight', 'Corn(maize) healthy', 'Grape Black rot', 'Grape
Esca(Black Measles)', 'Grape Leaf blight(Isariopsis Leaf Spot)', 'Grape healthy', 'Orange
Haunglongbing(Citrus greening)', 'Peach Bacterial spot', 'Peach healthy', 'Bell
PepperBacterial_spot', 'Pepper bell healthy', 'Potato Early blight', 'Potato Late blight', 'Potato
healthy', 'Raspberry healthy', 'Soybean healthy', 'Squash Powdery mildew', 'Strawberry Leaf
scorch', 'Strawberry healthy', 'Tomato Bacterial spot', 'Tomato Early blight', 'Tomato Late
blight', 'Tomato Leaf Mold', 'Tomato Septoria leaf spot', 'Tomato Spider mites (Two-spotted
spider mite)', 'Tomato Target Spot', 'Tomato Yellow Leaf Curl Virus', 'Tomato mosaic virus',
'Tomato healthy']

@app.route('/')
def home():
        return render_template("index.html")

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/about')
def aboutme():
    return render_template('about.html')

@app.route('/plantdisease/<res>')
def plantresult(res):
    print(res)
    corrected_result = ""
    for i in res:
        if i!='_':
            corrected_result = corrected_result+i
    return render_template('plantdiseaseresult.html', corrected_result=corrected_result)

@app.route('/plantdisease', methods=['GET', 'POST'])
def plantdisease():
    if request.method == 'POST':
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # If the user does not select a file, the browser submits an
        # empty file without a filename.
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            model = MODEL
            imagefile =
tensorflow.keras.utils.load_img(os.path.join(app.config['UPLOAD_FOLDER'], filename),
target_size=(224, 224, 3))
            input_arr = tensorflow.keras.preprocessing.image.img_to_array(imagefile)
            input_arr = np.array([input_arr])
            result = model.predict(input_arr)
            probability_model = tensorflow.keras.Sequential([model,
                                     tensorflow.keras.layers.Softmax()])
            predict = probability_model.predict(input_arr)
            p = np.argmax(predict[0])
            res = CLASSES[p]
            print(res)
            return redirect(url_for('plantresult', res=res))
    return render_template("plantdisease.html")

@app.route('/croprecommendation/<res>')
def cropresult(res):
    print(res)
    corrected_result = res
    return render_template('croprecresult.html', corrected_result=corrected_result)

@app.route('/croprecommendation', methods=['GET', 'POST'])
def cr():
    if request.method == 'POST':
        X = []
        if request.form.get('nitrogen'):
            X.append(float(request.form.get('nitrogen')))
        if request.form.get('phosphorous'):
            X.append(float(request.form.get('phosphorous')))
        if request.form.get('potassium'):
            X.append(float(request.form.get('potassium')))
        if request.form.get('temperature'):
            X.append(float(request.form.get('temperature')))
        if request.form.get('humidity'):
            X.append(float(request.form.get('humidity')))
        if request.form.get('ph'):
            X.append(float(request.form.get('ph')))
        if request.form.get('rainfall'):
            X.append(float(request.form.get('rainfall')))
        X = np.array(X)
        X = X.reshape(1, -1)
        res = REC_MODEL.predict(X)[0]
        # print(res)
        return redirect(url_for('cropresult', res=res))
    return render_template('croprec.html')
```

```
if __name__== "__main__":
    app.run(debug=True)
```