PROJECT REPORT

Fertilizers Recommendation for Disease Prediction

Team ID: PNT2022TMID17425

CHANDRU .S

MANIKANDAN .K

INBARAJAN .B

MULLAYARASU .K

INTRODUCTION :

- Agriculture is the most important sector in today's life. Most plants are affected by a wide variety of bacterial and fungal diseases. Diseases on plants placed a major constraint on the production and a major threat to food security. Hence, early and accurate identification of plant diseases is essential to ensure high quantity and best quality. In recent years, the number of diseases on plants and the degree of harm caused has increased due to the variation in pathogen varieties, changes in cultivation methods, and inadequate plant protection techniques.

Project Overview

- An Automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant.Deep learning techniques are used to identify the diseases and suggest the precautions that

  can be taken for those diseases changes in cultivation method and inadequate plant protection techniques and suggest all the precautions that can be taken for those diseases.

Purpose

- To Detect and recognize the plant diseases and to recommend fertilizer, it is necessary to identify the diseases and to recommend to get different and useful features needed for the purpose of analyzing later.

- To provide symptoms in identifying the disease at its earliest. Hence the authors proposed and implemented new fertilizers Recommendation System for Crop Disease Prediction.

LITREATURE SURVEY

Literature Review

[1] The proposed method uses SVM to classify tree leaves, identify the disease and suggest the fertilizer. The proposed method is compared with the existing CNN based leaf disease prediction. The proposed SVM technique gives a better result when compared to existing CNN. For the same set of images, F-Measure for CNN is 0.7and 0.8 for SVM, the accuracy of identification of leaf disease of CNN is 0.6 and SVM is 0.8.

A**dvantages :** The prediction and diagnosing of leaf diseases are depending on the segmentation such as segmenting the healthy tissues from diseased tissues of leaves.

Disadvantages : This further research is implementing the proposed algorithm with the existing public datasets. Also, various segmentation algorithms can be implemented to improve accuracy. The proposed algorithm can be modified further to identify the disease that affects the various plant organs such as stems and fruits.

[2] Detection of Leaf Diseases and Classification using Digital Image Processing International Conference on Innovations in Information, Embedded and Communication Systems(ICIIECS), IEEE,
2017.

Advantages: The system detects the diseases on citrus leaves with 90% accuracy.

Disadvantages: System only able to detect the disease from citrus leave.

The main objective of this paper is image analysis & classification techniques for detection of leaf diseases and classification. The leaf image is firstly preprocessed and then does the fur- ther work. K-Means Clustering used for image segmentation and then system extract the GLCM features from disease detected images. The disease classification done through the SVM classifier.

Algorithm used: Gray-Level Co-Occurrence Matrix (GLCM) features, SVM, K-Means Clustering .

[3] Semi-automatic leaf disease detection and classification system for soybean culture IET Image Processing, 2018

Advantages: The system helps to compute the disease severity.

Disadvantages: The system uses leaf images taken from an online dataset, so cannot implement in real time.

This paper mainly focuses on the detecting and classifying the leaf disease of soybean plant. Using SVM the proposed system classifies the leaf disease in 3 classes like i.e. downy mil- dew, frog eye, and septoria leaf blight etc. The proposed system gives maximum average classification accuracyreported is ~90% using a big dataset of 4775 images.

Algorithm used: SVM.

[4] Cloud Based Automated Irrigation And Plant Leaf Disease Detection System Using An Android Application. International Conference on Electronics, Communication and Aero- space Technology, ICECA 2017.

Advantages: It is simple and cost effective system for plant leaf disease detection.

Disadvantages: Any H/w failures may affect the system performance.

The current paper proposesan android application for irrigation and plant leaf disease detec- tion with cloud and IoT. For monitoring irrigation system they use soil moisture and temper- ature sensor and sensor data send to the cloud. The user can also detect the plant leaf disease. K-means clustering used for feature extraction.
Algorithm used: K-means clustering,
Other than this there are some other levels which can be used for sentimental analysis these are- document level, sentence level, entity and aspect level to study positive and negative, interrogative, sarcastic, good and bad functionality, sentiment without sentiment, conditional sentence and author and reader understanding points.

[5] The author proposes a method which helps us predict crop yield by suggesting the best crops. It also focuses on soil types in order to identify which crop should be planted in the field to increase productivity. In terms of crop yield, soil types are vital. By incorporating the weather details of the previous year into the equation, soil information can be obtained. A**dvantages :**It allows us to predict which crops would be appropriate for a given climate. Using the weather and disease related data sets, the crop quality can also be improved. Pre- diction algorithms help us to classify the data based on the disease, and data extracted from the classifier is used to predict soil and crop.

Disadvantages : Due to the changing climatic conditions, accurate results cannot be predicted by this system.

[6] The current work examines and describes image processing strategies for identifying plant diseases in numerous plant species. BPNN, SVM, K-means clustering, and SGDM are the most common approaches used to identify plant diseases.
Disadvantages : Some of the issues in these approaches include the impact of background data on the final picture, optimization of the methodology for a specific plant leaf disease, and automation of the technique for continuous automated monitoring of plant leaf diseases in real-world field circumstances.

[7] The proposed method uses SVM to classify tree leaves, identify the disease and suggest the fertilizer. The proposed method is compared with the existing CNN based leaf disease prediction. The proposed SVM technique gives a better result when compared to existing CNN. For the same set of images, F-Measure for CNN is 0.7and 0.8 for SVM, the accuracy of identification of leaf disease of CNN is 0.6 and SVM is 0.8.
A**dvantages :** The prediction and diagnosing of leaf diseases are depending on the segmenta- tion such as segmenting the healthy tissues from diseased tissues of leaves.
Disadvantages : This further research is implementing the proposed algorithm with the ex- isting public datasets. Also, various segmentation algorithms can be implemented to improve accuracy. The proposed algorithm can be modified further to identify the disease that affects the various plant organs such as stems and fruits.

[8] In this paper, we propose a user-friendly web applicationsystem based on machine learn- ing and web-scraping calledthe 'Farmer's Assistant'. With our system, we are successfully able to provide several features - crop recommendation using Random Forest algorithm, ferti- lizer recommendation using arule based classification system, and crop disease detection using EfficientNet model on leaf images. The user can provide the input using forms on our user interface and quickly gettheir results. In addition, we also use the LIME interpretability method

to explain our predictions on the disease detectionimage, which can potentially help understand why our modelpredicts what it predicts, and improve the datasets and models us- ing this information.

A**dvantages :** For crop recommendation and fertilizer recommendation, we can provide the availability of the same on the popular shopping websites, and possibly allow users to buy the crops and fertilizers directly from our application.

Disadvantages : To provide fine-grained segmentations of the diseased portion of the dataset. this is not possible due to lack of such data. However, in our application,we can integrate a segmentation annotation tool where theusers might be able to help us with the lack. Also, we can usesome unsupervised algorithms to pin-point the diseased areas in the image. We intend to add these features and fix thesegaps in our upcoming work.

Existing Problem

- Adequate mineral nutrition is central to crop production. However, it can also exert considerable Influence on disease development. Fertilizer application can increase or decrease development of diseases caused by different pathogens,
  and the mechanisms responsible are complex, including effects of nutrients on plant growth, plant resistance mechanisms and direct effects on the pathogen. The effects of mineral nutrition on plant disease and the mechanisms responsible for those effects have been dealt with comprehensively elsewhere. In India, around 40% of land is kept and grown using reliable irrigation technologies, while the rest relies on the monsoon environment for water. Irrigation decreases reliance on the monsoon, increases food security, and boosts agricultural production.

- Most research articles use humidity, moisture, and temperature sensors near the plant's root, with an external device handling all of the data provided by the sensors and transmitting it directly to an Android application. It was created to
  measure the approximate values of temperature, humidityand moisture sensors that were programmed into a microcontroller to manage the amount of water.

References :

[1] Semi-automatic leaf disease detection and classification system for soybean culture IET Image Processing, 2018

[2] Cloud Based Automated Irrigation And Plant Leaf Disease Detection System Using An Android Application. International Conference on Electronics, Communication and Aerospace Technology, ICECA 2017.

[3] Ms. Kiran R. Gavhale, Ujwalla Gawande, Plant Leaves Disease detection using Image

Processing Techniques, January 2014.

https://www.researchgate.net/profile/UjwallaGawande/publication/314436486_An_Overview_of_the_Research_on_Plant_Leaves_Disease_detection_using_Image_Processing_Techniques/links/5d3710664585153e591a3d20/An-Overviewof-the-Research-on-Plant-Leaves-Diseae detection-using-Image-ProcessingTechniques.pdf

[4] Duan Yan-e, Design of Intelligent Agriculture Management Information System Based on

IOT‖, IEEE,4th, Fourth International reference on Intelligent Computation Technology and Automation, 2011

https://ieeexplore.ieee.org/document/5750779

[5] R. Neela, P. Fertilizers Recommendation System For Disease Prediction In Tree Leave International journal of scientific & technology research volume 8, issue 11, november 2019

http://www.ijstr.org/final-print/nov2019/Fertilizers-Recommendation-System-For-Disease-Prediction-In-Tree-Leave.pdf .

[6] Swapnil Jori1, Rutuja Bhalshankar2, Dipali Dhamale3, Sulochana Sonkamble , Healthy Farm: Leaf Disease Estimation and Fertilizer Recommendation System using Machine Learning,International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211

[7] Detection of Leaf Diseases and Classification using Digital Image Processing
International Conference on Innovations in Information, Embedded and Communication Systems(ICIIECS), IEEE,
2017.

[8] Shloka Gupta ,Nishit Jain ,Akshay Chopade, Farmer's Assistant: A Machine Learning BasedApplication for Agricultural Solutions.

Problem Statement Definition:

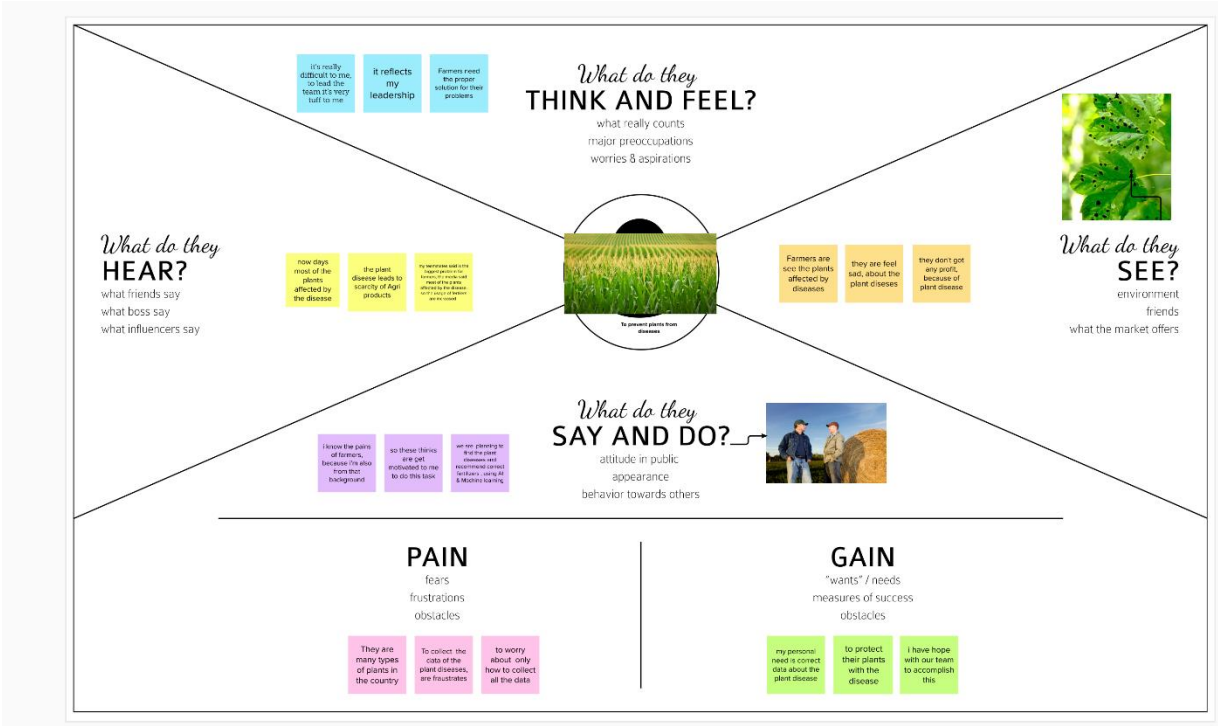
Mr.Subramany is a 80 years old man. He had a own farming land and do Agriculture for past 40 Years , In this 40 Years he Faced a problem in Choosing Fertilizers and Controlling of Plant Disease.

- Subramany wants to know the better recommendation for fertilizers for plants with the disease.
- He has faced huge losses for a long time.
- This problem is usually faced by most farmers.
- Mr. Subramany needs to know the result immediately.

Who does the problem affect?	Persons who do Agriculture
What are the boundaries of the problem?	People who Grow Crops and facing Issues of Plant Disease
What is the issue?	In agricultural aspects, if the plant is affected by leaf disease, then it reduces the growth and productiveness. Generally, the plant diseases are caused by the abnormal physiological functionalities of plants.
When does the issue occur?	During the development of the crops as they will be affected by various diseases.

Where does the issue occur?	The issue occurs in agriculture practicing areas, particularly in rural regions.
Why is it important that we fix the problem?	It is required for the growth of better quality food products. It is important to maximise the crop yield.
What solution to solve this issue?	An automated system is introduced to identify different diseases on plants by checking the symptoms shown on the leaves of the plant.
What methodology used to solve the issue?	Deep learning techniques are used to identify the diseases and suggest the precautions that can be taken for those diseases.

IDEATION &EMPATHY MAP CANVAS



Share your feedback

ProposedSolution:

- The idea of the proposed solution uses Deep learning and Machine algorithm to classify leaves and identify the diseases and siggest the fertilizers. The deep learning process includes the MobileNetV2 and VGG19 training Models.

- Based on the leaf disease detected , the model recommendation for fertilizers for the prevention. The farmers and researchers arethe endusersgetbenefied by the system.

- More accurate in others. The system is more robust corporating more image data sets with wider variations. This system also estimates the probability of infected plant.
- Plant growth can be enhanced. Ensure plants are getting supplied with every nutrient they need also and multiple cross in grow in every yields for every season. It also helps people's nutritional needs.

Problem Solution Fit

- This Learn and Build phase has proven to be the most important, parallel phase that successful startups follow. It contains the very first activity that startups should follow if they have an idea: Find prospective customers to talk to. Usually, this idea is already translated to a software product, which should always be a Minimum Viable Product (MVP) a version of the product that requires the least amount of development time with a minimum amount of effort.

- An MVP is based on requirements desired by potential customers, but to obtain these requirements, the startup should talk as early as possible with those customers. The startup then requires to prioritise the ‘must haves’, which are the minimum necessary requirements for the MVP. Once the MVP is ready for customer feedback, the second most important activity is performed by the startup foreveryone.

PNT2022TMID17425	Problem Solution-Fit	Fertilizers Recommendation System for Disease Prediction

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) — CS

Who is your customer?

- Farmers are our primary customers to solve their problem in choosing right fertilizers.
- Our secondary customers are the researchers to make their job easy with our AI Technology.
- People who couldn't afford for a Consultant for choosing crops and fertilizers.

6. CUSTOMER CONSTRAINTS — CC

What constraints prevent your customers from taking action or limit their choices of solutions?

- This is basically a web application , Which is Supported in almost all devices.
- The easy graphical representation make a clear understanding for all people.
- The Results for their problem will be in minute .

5. AVAILABLE SOLUTIONS — AS

Which solutions are available to the customers when they face the or need to get the job done?

- By using the AI will end up the existed problem , by provide results in low price.
- Its affordable by all people and the results are provided instantly
- Its Supports in Mobile ,Desktop, etc (Almost all device support)

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS — J&P

Which jobs-to-be-done (or problems) do you address for your customers?

- Its provides a good fertilizer recommendation for their crops.
- Its analyzes the disease which affects their plants .
- Its shows a set of crops which suitable for their soil and their climate .

9. PROBLEM ROOT CAUSE — RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?

- The traditional way are expensive.
- Farmers want to get results instantly .
- To improve Production in low cost and easy .
- Traditional way not contains a easily understandable graphical representation of results .

7. BEHAVIOUR — BE

What does your customer do to address the problem and get the job

- By using our product , they able to saves a lot of money spend for a expert.
- Its saves a time and makes their process faster .
- It improves their field growth with our product .
- It ensures the causes previously and provide solutions before the damage happens.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS — TR

- People will feel that our provides a bunch of valuable service affordable.

4. EMOTIONS: BEFORE / AFTER — EM

- Its reduces the farmers unwanted Work load ,stress , money , time , etc ...

10. YOUR SOLUTION — SL

- By Building a AI , ML based web application make their issues resolved in seconds .
- Make their expensive process affordable .
- Minimize the Time for analyze their problem and provide results in seconds .
- Easy Graphical representation makes a better understanding by everyone .

8. CHANNELS of BEHAVIOUR — CH

ONLINE

- Their Data analyzed early with help of cloud rendering

OFFLINE

- Its improves their crops production and reduces the losses .

Extract online & offline CH of BE

REQUIREMENT ANALYSIS :

Functional Requirements

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Specific characteristics	It identifies the diseases especially rice bran diseases
FR-4	Functions	The proposed methods uses the SVM to classify tree leaves, identify the diseases and suggest the fertilizer.
FR-5	Fault tolerance	This study enables a possible prediction of crop yield from the historic data collected and offers a suggestion to farmers.
FR-6	Analyze	It helps us to classify the data based on the diseases, and data extracted from the classifier is used to predict soil and crop.

Non Functional Requirements

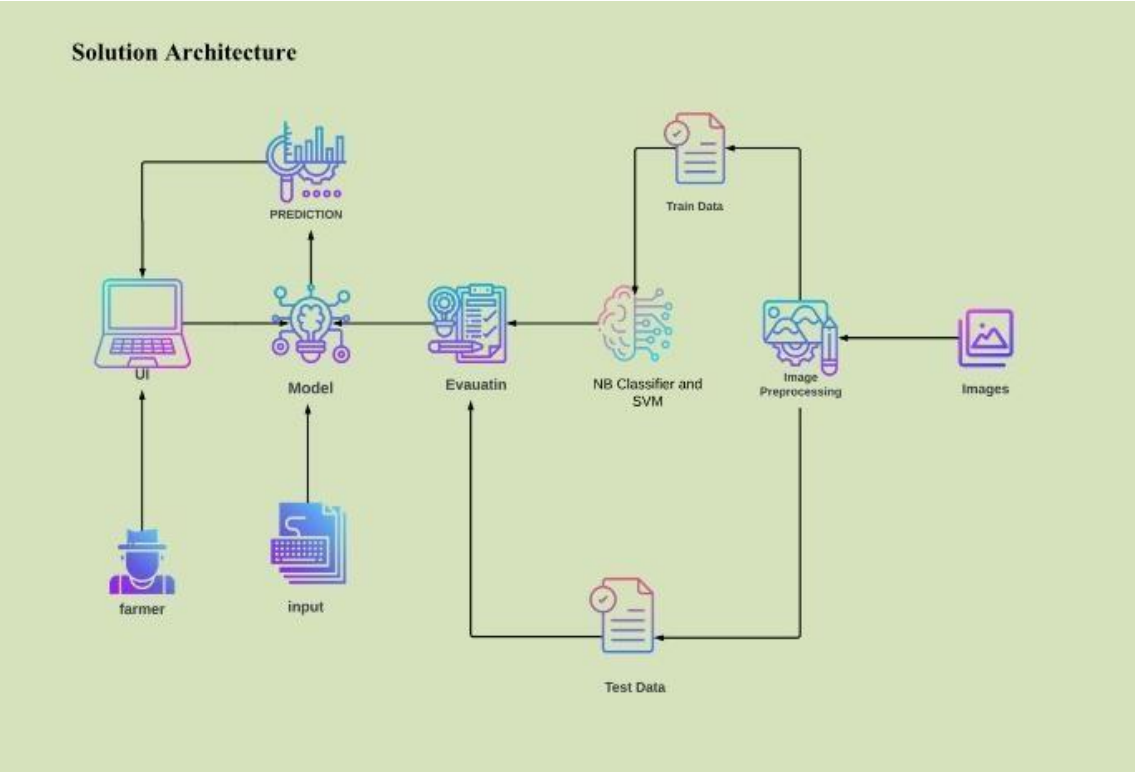
Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

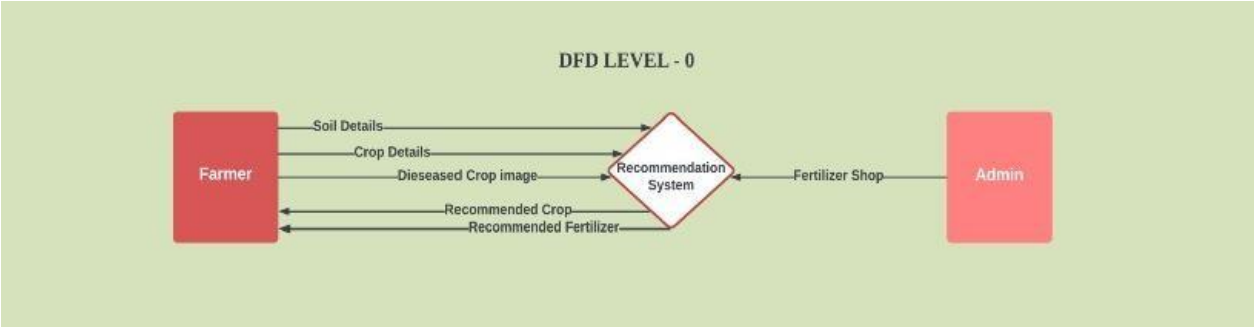
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Crop and fertilizer recommendation system help the farmer to identify the diseases.
NFR-2	**Security**	The proposed method combines two major aspects in farming , pest identification and insecticide recommendation.
NFR-3	**Reliability**	It is easy use so that health issues can be avoided.
NFR-4	**Performance**	Precision fertilizer and precision crops is mostly used. They used to predict the crop in artificial intelligence.
NFR-5	**Availability**	reduces the losses as ammonia , nitrate leaching, apply the right rate, apply accurately.
NFR-6	**Scalability**	If the soil is not replenised with nutrients through fertilizing ,crop yields will deteriorate over time.

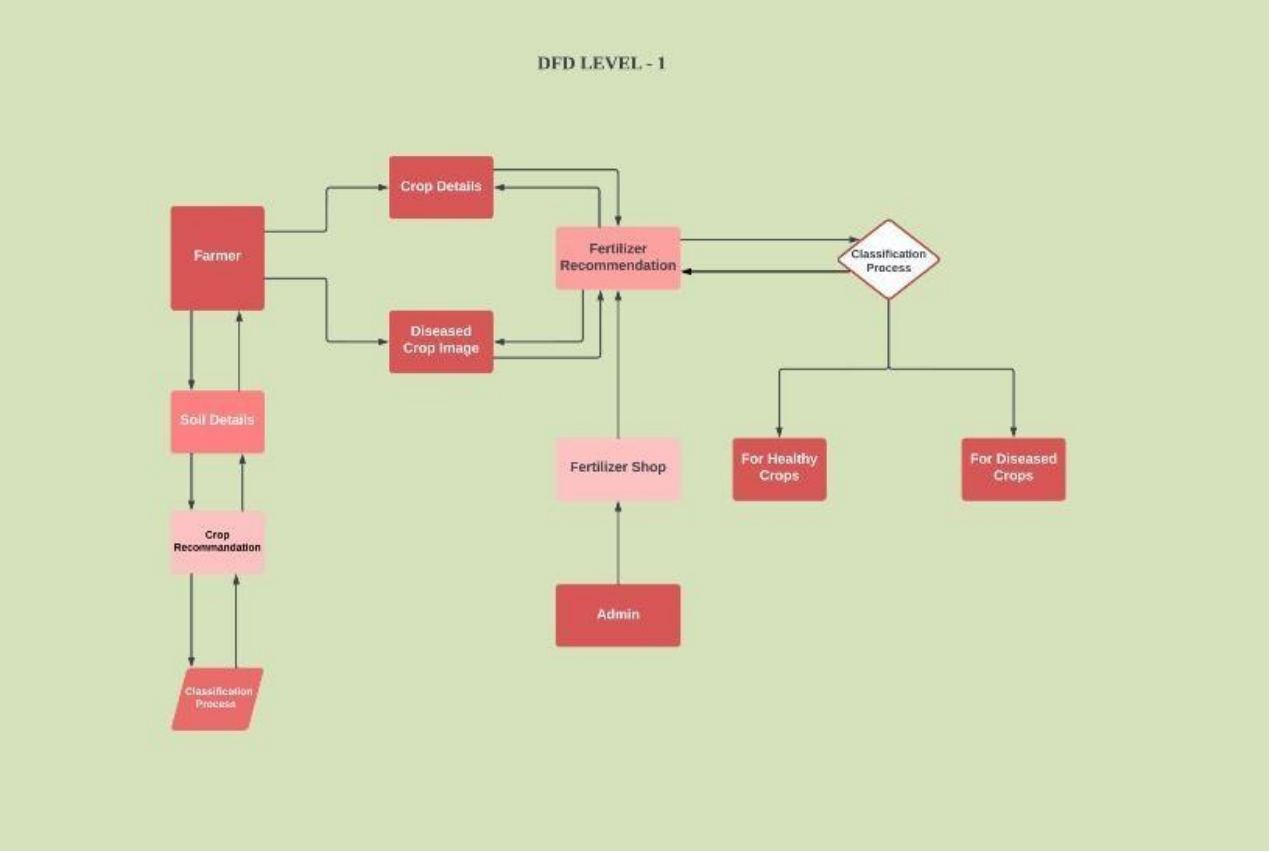
PROJECT DESIGN :

Solution & Technical Architecture



Data Flow Diagrams





User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile ssuser)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	**Sprint-1**
	Login	USN-2	As a user, I can log into the application by entering email & password	I can login using my E-mail ID accounts or user credentials	High	**Sprint-1**
	Dashboard	USN-3	As a user, I can view the page of the application where i can upload my images and the fertilizer should be recommended	I can access my account/ dashboard	High	**Sprint-2**
Customer (Webuser)	Registration	USN-4	As a user, I can login to web dashboard just Like website dashboard	I can register using my username and password	High	**Sprint-3**
	Login	USN-5	As a user, I can login to my web dashboard with the login credentials	I can login using my User credentials	High	**Sprint-3**
	Dashboard	USN-6	As a user, I can view the web application where i can upload my images and thefertilizer should be recommended	I can access my account/ dashboard	High	**Sprint-4**
		USN-7	As a user, the fertilizer recommended to me should be of higher accuracy	I can access my account/ dashboard	High	**Sprint-4**
Administrator	Login	USN-8	As a admin, I can login to the website using my login credentials	I can login to the website using my login credentials	High	**Sprint-5**
	Dashboard	**USN-9**	**As a admin, I can view the dashboard of the application**	**I can access my dashboard**	**High**	**Sprint-5**

PROJECT PLANNING & SCHEDULING:

Sprint Planning and Estimation

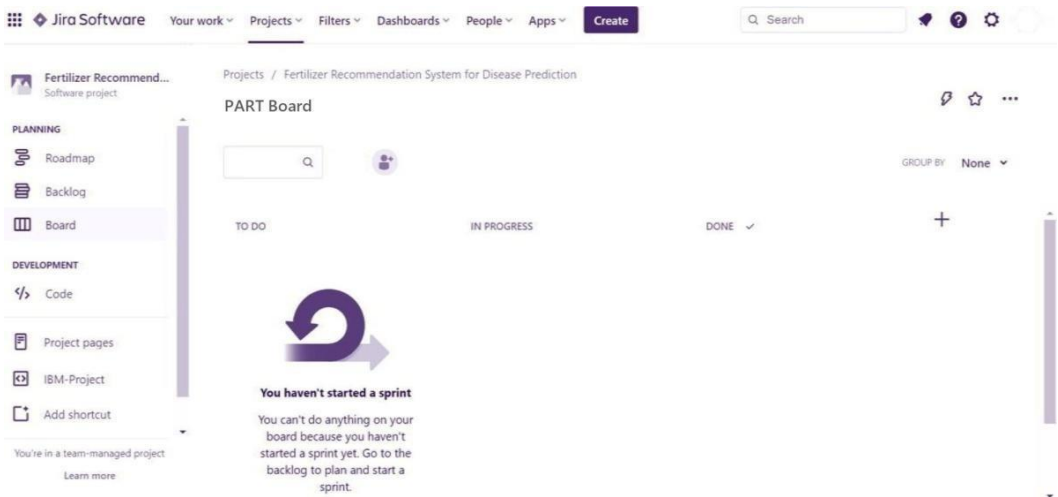
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points (Total)	Priority	Team Members
Sprint-1	Model Creation and Training (Fruits)		Create a model which can classify diseased fruit plants from given images. I also need to test the model and deploy it on IBM Cloud	8	High	Chandru,Mullaiyarasu,Inbarajan,Manikandan
	Model Creation and Training (Vegetables)		Create a model which can classify diseased vegetable plants from given images	2	High	Chandru,Mullaiyarasu,Inbarajan,Manikandan

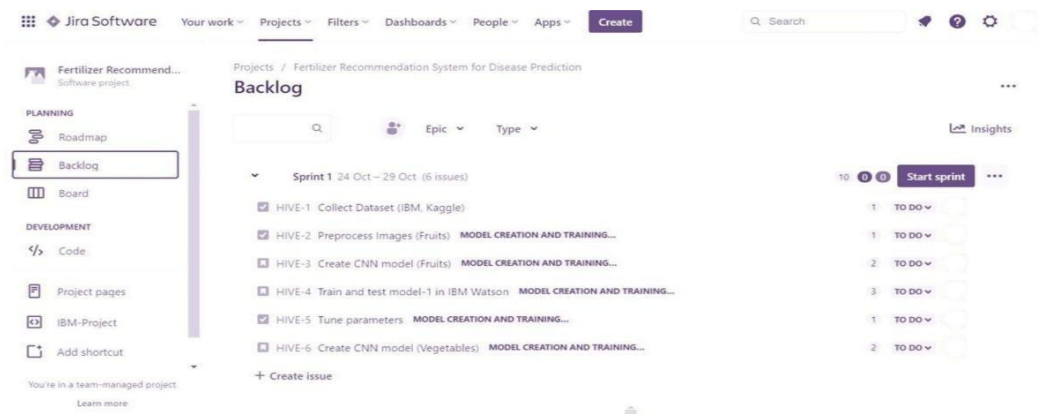
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points (Total)	Priority	Team Members
Sprint-2	Model Creation and Training (Vegetables)		Create a model which can classify diseased vegetable plants from given images and train on IBM Cloud	6	High	Chandru,Mullaiyarasu,Inbarajan,Manikandan
	Registration	USN-1	As a user, I can register by entering my email, password, and confirming my password or via OAuth API	3	Medium	Chandru,Mullaiyarasu,Inbarajan,Manikandan.
	Upload page	USN-2	As a user, I will be redirected to a page where I can upload my pictures of crops	4	High	Chandru,Mullaiyarasu,Inbarajan,Manikandan
	Suggestion results	USN-3	As a user, I can view the results and then obtain the suggestions provided by the ML model	4	High	Chandru,Mullaiyarasu,Inbarajan,Manikandan
	Base Flask App		A base Flask web app must be created as an interface for the ML model	2	High	Chandru,Mullaiyarasu,Inbarajan, Manikandan.
Sprint-3	Login	USN-4	As a user/admin/shopkeeper, I can log into the application by entering email & password	2	High	Chandru,Mullaiyarasu,Inbarajan,Manikandan
	User Dashboard	USN-5	As a user, I can view the previous results and history	3	Medium	Chandru,Mullaiyarasu,Inbarajan, Manikandan
	Integration		Integrate Flask, CNN model with Cloudant DB	5	Medium	Chandru,Mullaiyarasu,Inbarajan, Manikandan
	Containerization		Containerize Flask app using Docker	2	Low	Chandru,Mullaiyarasu,Inbarajan,Manikandan
Sprint-4	Dashboard (Admin)	USN-6	As an admin, I can view other user details and uploads for other purposes	2	Medium	Chandru,Mullaiyarasu,Inbarajan, Manikandan
	Dashboard (Shopkeeper)	USN-7	As a shopkeeper, I can enter fertilizer products and then update the details if any	2	Low	Chandru,Mullaiyarasu,Inbarajan,Manikandan
	Containerization		Create and deploy Helm charts using Docker Image made before	2	Low	Chandru,Mullaiyarasu,Inbarajan, Manikandan

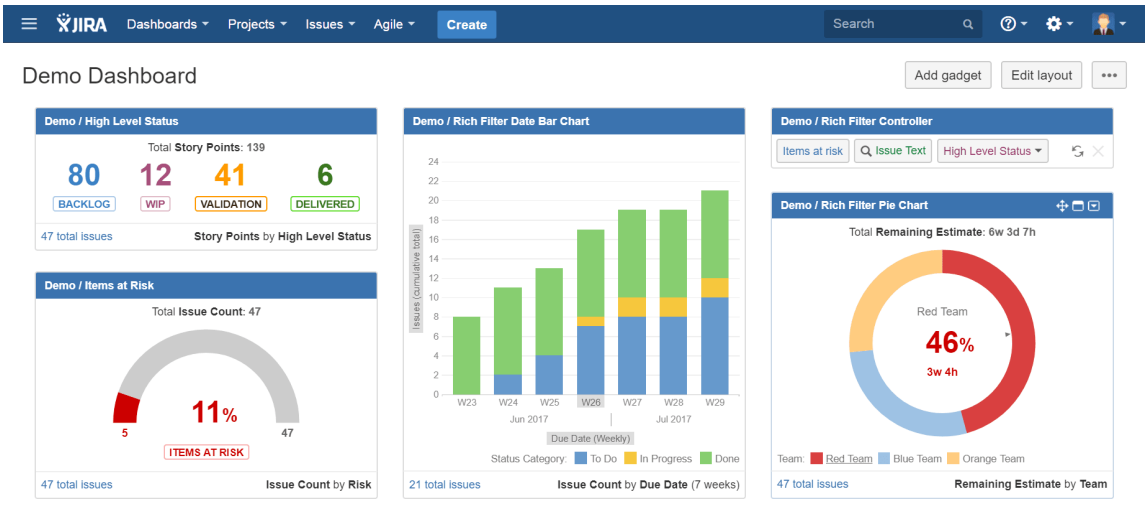
Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022	10	30 Oct 2022
Sprint-2	15	6 Days	31 Oct 2022	05 Nov 2022	15	06 Nov 2022
Sprint-3	15	6 Days	07 Nov 2022	12 Nov 2022	15	13 Nov 2022
Sprint-4	12	6 Days	14 Nov 2022	19 Nov 2022	10	20 Nov 2022

Reports from JIRA







Feature 1[Model Building]:

1. Import The Libraries

Import the libraries that are required to initialize the neural network layer, and create and add different layers to the neural network model.

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
```

2. Initializing The Model

Keras has 2 ways to define a neural network:

- Sequential
- Function API

The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add () method.
Now, will initialize our model.

Initialize the neural network layer by creating a reference/object to the Sequential class.

```
model=Sequential()
```

3. ADD CNNLayers

We will be adding three layers for CNN

- Convolution layer
- Pooling layer
- Flattening layer

Add Convolution Layer

The first layer of the neural network model, the convolution layer will be added. To create a convolution layer, Convolution2D class is used. It takes a number of feature detectors, feature detector size, expected input shape of the image, and activation function as arguments. This

layer applies feature detectors on the input image and returns a feature map (features from the image).

Activation Function: These are the functions that help us to decide if we need to activate the node or not. These functions introduce non-linearity in the networks.

```
model.add(Convolution2D(32,(3,3),input_shape = (128,128,3),activation = 'relu'))
```

Add the pooling layer

Max Pooling selects the maximum element from the region of the feature map covered by the filter. Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

After the convolution layer, a pooling layer is added. Max pooling layer can be added using MaxPooling2D class. It takes the pool size as a parameter. Efficient size of the pooling matrix is (2,2). It returns the pooled feature maps. (Note: Any number of convolution layers, pooling and dropout layers can beadded)

```
model.add(MaxPooling2D(pool_size = (2,2)))
```

Add the flatten layer

The flatten layer is used to convert n-dimensional arrays to 1-dimensional arrays. This 1D array will be given as input to ANN layers.

```
model.add(Flatten())
```

4. Add Dense Layers

Now, let's add Dense Layers to know more about dense layers click below

Dense layers

The name suggests that layers are fully connected (dense) by the neurons in a network layer. Each neuron in a layer receives input from all the neurons present in the previous layer. Dense is used to add thelayers.

Adding Hidden layers

This step is to add a dense layer (hidden layer). We flatten the feature map and convert it into a vector or single dimensional array in the Flatten layer. This vector array is fed it as an input to the neural network and applies an activation function, such as sigmoid or other, and returns the output.

- init is the weight initialization; function which sets all the weights and biases of a network to values suitable as a starting point for training.
- units/ output_dim, which denote is the number of neurons in the hidden layer.
- The activation function basically decides to deactivate neurons or activate them to get the desired output. It also performs a nonlinear transformation on the input to get better results on a complex neural network.
- You can add many hidden layers, in our project we are added two hidden layers. The 1st hidden layer with 40 neurons and 2nd hidden layer with 20neurons.

Adding the output layer

This step is to add a dense layer (output layer) where you will be specifying the number of classes your dependent variable has, activation function, and weight initializer as the arguments. We use the add () method to add dense layers. the output dimensions here is 6

```
model.add(Dense(output_dim = 40 ,init = 'uniform',activation = 'relu'))
model.add(Dense(output_dim = 20 ,init = 'random_uniform',activation = 'relu'))
model.add(Dense(output_dim = 6,activation = 'softmax',init ='random_uniform'))
```

5. Train And Save The Model

Compile the model

After adding all the required layers, the model is to be compiled. For this step, loss function, optimizer and metrics for evaluation can be passed as arguments.

```
model.compile(loss = 'categorical_crossentropy',optimizer = "adam",metrics = ["accuracy"])
```

Fit and save the model

Fit the neural network model with the train and test set, number of epochs and validation steps. Steps per epoch is determined by number of training images/ batch size, for validation steps number of validation images/ batch size.

```
model.fit_generator(x_train, steps_per_epoch = 168,epochs = 3,validation_data = x_test,validation_steps = 52)
```

Accuracy, Loss: Loss value implies how poorly or well a model behaves after each iteration of optimization. An accuracy metric is used to measure the algorithm's performance in an

interpretable way. The accuracy of a model is usually determined after the model parameters and is calculated in the form of a percentage.

The weights are to be saved for future use. The weights are saved in as .h5 file using save().

```
model.save("fruit.h5")
```

model.summary() can be used to see all parameters and shapes in each layer in our models.

6. Test The Model

The model is to be tested with different images to know if it is working correctly.

Import the packages and load the saved model

Import the required libraries

```
from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
```

Initially, we will be loading the fruit model. You can test it with the vegetable model in a similar way.

```
model = load_model("fruit.h5")
```

Load the test image, pre-process it and predict

Pre-processing the image includes converting the image to array and resizing according to the model. Give the pre-processed image to the model to know to which class your model belongs to.

```
img = image.load_img('apple_healthy.JPG',target_size = (128,128))
```

```
x  = image.img_to_array(img)
x = np.expand_dims(x,axis = 0)
```

```
pred = model.predict_classes(x)
```

```
pred
```

```
[1]
```

The predicted class is 1.

Feature 2[Python Code]:

Build Python Code:

After the model is built, we will be integrating it into a web application so that normal users can also use it. The user needs to browse the images to detect the disease.

Activity 1: Build a flask application

Step 1: Load the required packages

```
import requests
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
import numpy as np
import pandas as pd
import tensorflow as tf
from flask import Flask, request, render_template, redirect, url_for
import os
from werkzeug.utils import secure_filename
from tensorflow.python.keras.backend import set_session
```

Step 2: Initialize the flask app and load the model

An instance of Flask is created and the model is loaded using load_model from Keras.

```
app = Flask(__name__)

#load both the vegetable and fruit models
model = load_model("vegetable.h5")
model1=load_model("fruit.h5")
```

Step 3: Configure the home page

```
#home page
@app.route('/')
def home():
    return render_template('home.html')
```

Step 4: Pre-process the frame and run

Pre-process the captured frame and give it to the model for prediction. Based on the prediction the output text is generated and sent to the HTML to display. We will be loading the precautions for fruits and vegetables excel file to get the precautions based on the output and return it to the HTML Page.

```
#prediction page
@app.route('/prediction')
def prediction():
    return render_template('predict.html')

@app.route('/predict',methods=['POST'])
def predict():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['image']
        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        img = image.load_img(file_path, target_size=(128, 128))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        plant=request.form['plant']
        print(plant)
        if(plant=="vegetable"):
            preds = model.predict_classes(x)
            print(preds)
            df=pd.read_excel('precautions - veg.xlsx')
            print(df.iloc[preds[0]]['caution'])
        else:
            preds = model1.predict_classes(x)

            df=pd.read_excel('precautions - fruits.xlsx')
            print(df.iloc[preds[0]]['caution'])
        return df.iloc[preds[0]]['caution']
```

Run the flask application using the run method. By default, the flask runs on 5000 port. If the port is to be changed, an argument can be passed and the port can be modified.

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

TESTING

User Acceptance Testing:

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Fertilizers Recommendation System for Disease Prediction project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	0	0	1	0	1
Duplicate	1	3	2	2	8
External	2	3	0	0	5
Fixed	4	4	4	4	16
Not Reproduced	0	0	0	1	1
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	7	10	7	7	31

3. Test Case Analysis

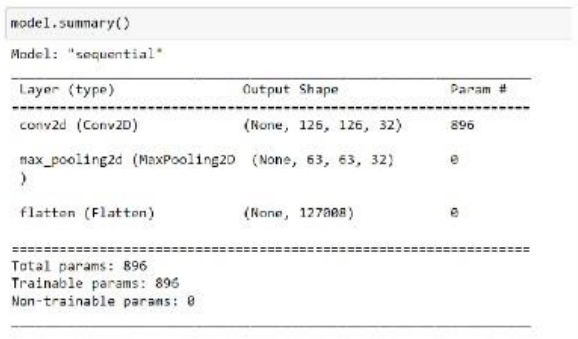
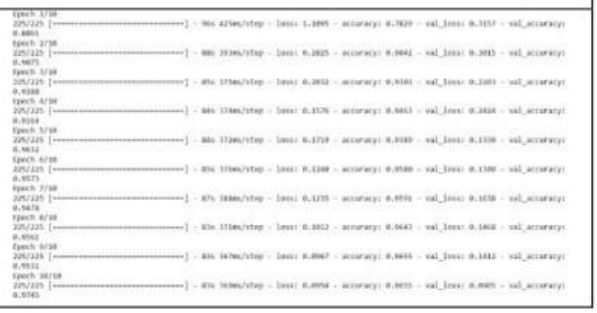
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	1	0	0	1
Client Application	1	0	0	1
Security	1	0	0	1
Outsource Shipping	1	0	0	1
Exception Reporting	1	0	0	1
Final Report Output	1	0	0	1
Version Control	1	0	0	1

RESULTS:

Performance Metrics:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Model Summary	Total params: 896 Trainable params: 896 Non-trainable params: 0	
2.	Accuracy	Training Accuracy – 96.55 Validation Accuracy – 97.45	

Model Summary:

```
model.summary()
```

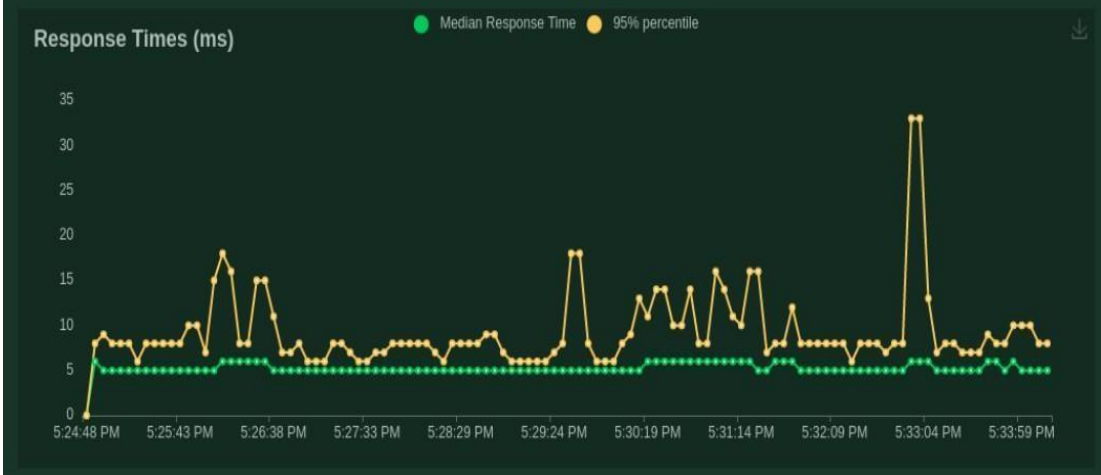
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 max_pooling2d (MaxPooling2D  (None, 63, 63, 32)       0
 )

 flatten (Flatten)           (None, 127008)            0

=================================================================
Total params: 896
Trainable params: 896
Non-trainable params: 0
_________________________________________________________________
```

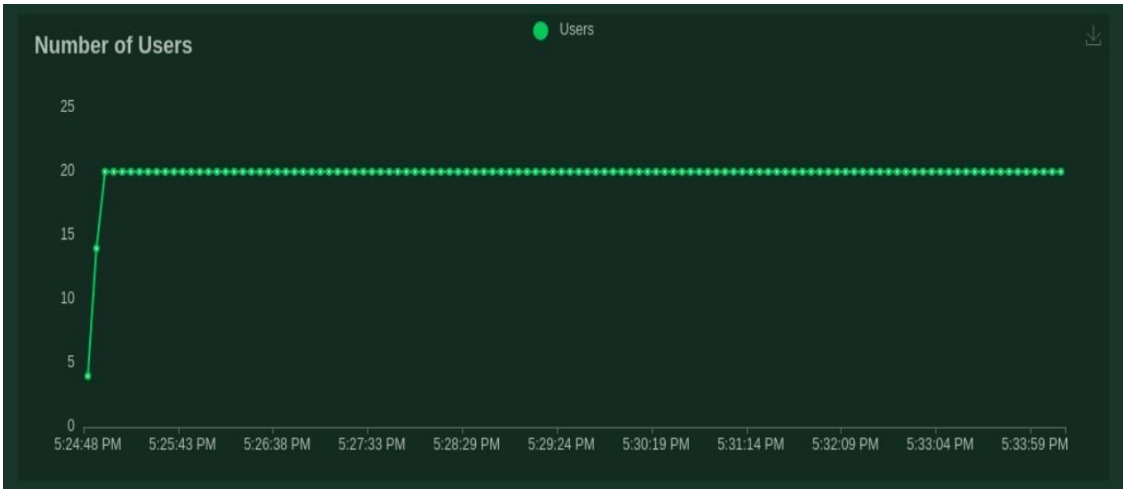
Accuracy:

```
Epoch 1/10
225/225 [==============================] - 96s 425ms/step - loss: 1.1095 - accuracy: 0.7829 - val_loss: 0.3157 - val_accuracy:
0.8861
Epoch 2/10
225/225 [==============================] - 88s 393ms/step - loss: 0.2825 - accuracy: 0.9042 - val_loss: 0.3015 - val_accuracy:
0.9075
Epoch 3/10
225/225 [==============================] - 85s 375ms/step - loss: 0.2032 - accuracy: 0.9303 - val_loss: 0.2203 - val_accuracy:
0.9288
Epoch 4/10
225/225 [==============================] - 84s 374ms/step - loss: 0.1576 - accuracy: 0.9463 - val_loss: 0.2424 - val_accuracy:
0.9164
Epoch 5/10
225/225 [==============================] - 84s 372ms/step - loss: 0.1719 - accuracy: 0.9389 - val_loss: 0.1330 - val_accuracy:
0.9632
Epoch 6/10
225/225 [==============================] - 85s 376ms/step - loss: 0.1240 - accuracy: 0.9580 - val_loss: 0.1340 - val_accuracy:
0.9573
Epoch 7/10
225/225 [==============================] - 87s 388ms/step - loss: 0.1235 - accuracy: 0.9591 - val_loss: 0.1638 - val_accuracy:
0.9478
Epoch 8/10
225/225 [==============================] - 83s 371ms/step - loss: 0.1012 - accuracy: 0.9643 - val_loss: 0.1468 - val_accuracy:
0.9561
Epoch 9/10
225/225 [==============================] - 83s 367ms/step - loss: 0.0967 - accuracy: 0.9655 - val_loss: 0.1412 - val_accuracy:
0.9531
Epoch 10/10
225/225 [==============================] - 83s 369ms/step - loss: 0.0954 - accuracy: 0.9655 - val_loss: 0.0905 - val_accuracy:
0.9745
```

Locust report:

Charts





Final ratio

Ratio per User class

- 100.0% AppUser
  - 50.0% home
  - 50.0% prediction

Total ratio

- 100.0% AppUser
  - 50.0% home
  - 50.0% prediction

ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

- The proposed model could predict the disease just from the image of a part icular plant.
- Easy to use UI.
- Model has some good accuracy in detecting the plant just by taking the **input(leaf).**
- These kind of web applications can be used in the agricultural sector as well as for small house hold plants as well.

Disadvantages:

- Prediction is limited to few plants as we havent trained all the plants.

Conclusion :

- The core strategy of this project is to predict the crop based on the soil nutrient content and the location where the crop is growing. This system will help he farmers to choose the right crop for their land and to give the suitable amount of fertilizer to produce the maximum yield. The Support Vector Machine algorithm helps to predict the crop the precisely based on the pre-processed crop data. This system will also help the new comers to choose the crop which will grow in their area and produce them a good profit. A decent amount of profit will attract morepeopletowardstheagriculture.

Future Scope :

- As of now we have just built the web application which apparently takes the input as an image and then predict the out in the near future we can develop an application which computer vision and AI techniques to predict the infection once you keep the camera near the plant or leaf this could make our project even more usable

- This further research is implementingthe proposed algorithm with the existing public datasets. Also, various segmentation algorithms can be implemented to improve accuracy. The proposed algorithm can be modified further to identify the disease that affects the various plant organs such as vegetables and fruits.

Appendix :

Requirement.txt

```
numpy
pandas
Flask
scikit-learn
https://download.pytorch.org/whl/cpu/torch-1.7.0%2Bcpu-cp36-cp36m-linux_x86_64.whl
https://download.pytorch.org/whl/cpu/torchvision-0.8.1%2Bcpu-cp36-cp36m-linux_x86_64.whl
requests
Pillow
gunicorn == 20.0.4
asgiref==3.5.0
bcrypt==3.2.0
cffi==1.15.0
click==8.1.2
dnspython==2.2.1
email-validator==1.1.3
Flask==2.1.1
Flask-Bcrypt==1.0.1
Flask-Login==0.6.0
Flask-SQLAlchemy==2.5.1
Flask-WTF==1.0.1
greenlet==1.1.2
idna==3.3
importlib-metadata==4.11.3
itsdangerous==2.1.2
Jinja2==3.1.1
MarkupSafe==2.1.1
pycparser==2.21
six==1.16.0
SQLAlchemy==1.4.35
sqlparse==0.4.2
Werkzeug==2.1.1
WTForms==3.0.1
zipp==3.8.0
flask_sqlalchemy
flask_login
flask_wtf
wtforms
wtforms.validators
flask_bcrypt
```

App.py

```
# Importing essential libraries and modules

from flask import Flask, render_template, request, Markup,url_for, redirect
import numpy as np
import pandas as pd
from utils.disease import disease_dic
from utils.fertilizer import fertilizer_dic
```

```python
import requests
import config
import pickle
import io
import torch
from torchvision import transforms
from PIL import Image
from utils.model import ResNet9
from flask_sqlalchemy import SQLAlchemy
from flask_login import UserMixin, login_user, LoginManager, login_required, logout_user, current_user
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import InputRequired, Length, ValidationError
from flask_bcrypt import Bcrypt

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

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

# Loading plant disease classification model

disease_classes = ['Apple___Apple_scab',
                   'Apple___Black_rot',
                   'Apple_Cedar_apple_rust',
                   'Apple___healthy',
                   'Blueberry__healthy',
                   'Cherry_(including_sour)_Powdery_mildew',
                   'Cherry_(including_sour)__healthy',
                   'Corn_(maize)_Cercospora_leaf_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']

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


# Loading crop recommendation model

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


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

# Custom functions for calculations


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

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

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

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


def predict_image(img, model=disease_model):
    """
    Transforms image to tensor and predicts disease label
    :params: image
    :return: prediction (string)
    """
    transform = transforms.Compose([
        transforms.Resize(256),
```

```
        transforms.ToTensor(),
    ])
    image = Image.open(io.BytesIO(img))
    img_t = transform(image)
    img_u = torch.unsqueeze(img_t, 0)

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

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


app = Flask(__name__)

db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
app.config['SECRET_KEY'] = 'thisisasecretkey'


login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'


@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))


class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), nullable=False, unique=True)
    password = db.Column(db.String(80), nullable=False)


class RegisterForm(FlaskForm):
    username = StringField(validators=[
                           InputRequired(), Length(min=4, max=20)], render_kw={"placeholder": "Username"})

    password = PasswordField(validators=[
                             InputRequired(), Length(min=8, max=20)], render_kw={"placeholder": "Password"})

    submit = SubmitField('Register')

    def validate_username(self, username):
        existing_user_username = User.query.filter_by(
            username=username.data).first()
        if existing_user_username:
            raise ValidationError(
```

```
                'That username already exists. Please choose a different one.')


class LoginForm(FlaskForm):
    username = StringField(validators=[
                           InputRequired(), Length(min=4, max=20)], render_kw={"placeholder": "Username"})

    password = PasswordField(validators=[
                             InputRequired(), Length(min=8, max=20)], render_kw={"placeholder": "Password"})

    submit = SubmitField('Login')


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


@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()
        if user:
            if bcrypt.check_password_hash(user.password, form.password.data):
                login_user(user)
                return redirect (url_for('dashboard'))
    return render_template('login.html', form=form)


@app.route('/dashboard', methods=['GET', 'POST'])
@login_required
def dashboard():
    return render_template('dashboard.html')


@app.route('/logout', methods=['GET', 'POST'])
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))


@ app.route('/register', methods=['GET', 'POST'])
def register():
    form = RegisterForm()

    if form.validate_on_submit():
        hashed_password = bcrypt.generate_password_hash(form.password.data)
        new_user = User(username=form.username.data, password=hashed_password)
        db.session.add(new_user)
        db.session.commit()
        return redirect(url_for('login'))

    return render_template('register.html', form=form)
```

```python
# render home page


# render crop recommendation form page


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

# render fertilizer recommendation form page


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

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

# render disease prediction input page




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

# RENDER PREDICTION PAGES

# render crop recommendation result page


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

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

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

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

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

        else:
```

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

# render fertilizer recommendation result page


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

    crop_name = str(request.form['cropname'])
    N = int(request.form['nitrogen'])
    P = int(request.form['phosphorous'])
    K = int(request.form['pottasium'])
    # ph = float(request.form['ph'])

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

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

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

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

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

# render disease prediction result page


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

    if request.method == 'POST':
        if 'file' not in request.files:
```

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

            prediction = predict_image(img)

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


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

Notebook Code(ipynb File) :

```
# Creating final data for crop and fertilizer recommendation system
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
fertilizer_data_path = '../Data-raw/FertilizerData.csv'
merge_fert = pd.read_csv(fertilizer_data_path)
merge_fert.head()
del merge_fert['Unnamed: 0']
merge_fert.describe()
merge_fert['Crop'].unique()
plt.plot(merge_fert["N"])
plt.plot(merge_fert["P"])
plt.plot(merge_fert["K"])
sns.heatmap(merge_fert.corr(),annot=True)
merge_crop = pd.read_csv('../Data-raw/MergeFileCrop.csv')
reco_fert = merge_fert
#Add +/-3 for every NPK value
import random
temp = pd.DataFrame(columns = ['N','P','K'])
for i in range(0,merge_crop.shape[0]):
    crop = merge_crop.label.iloc[i]
    #print(crop)
    N = reco_fert[reco_fert['Crop'] == crop]["N"].iloc[0] + random.randint(-20,20)
    P = reco_fert[reco_fert['Crop'] == crop]["P"].iloc[0] + random.randint(-5,20)
    K = reco_fert[reco_fert['Crop'] == crop]["K"].iloc[0] + random.randint(-5,5)
    d = {"N":N,"P":P,"K":K}
    #print(d)
    temp = temp.append(d,ignore_index = True)
temp
merge_crop['N'] = temp['N']
merge_crop['P'] = temp['P']
```

```
merge_crop['K'] = temp['K']
merge_crop
del merge_crop['Unnamed: 0']
merge_crop
merge_crop = merge_crop[[ 'N', 'P', 'K','temperature', 'humidity', 'ph', 'rainfall', 'label']]
merge_crop.to_csv("../Data-processed/crop_recommendation.csv",index=False)
# Checking if everything went fine
df = pd.read_csv('../Data-processed/crop_recommendation.csv')
df.head()
df.shape

# 🏵🏵 PLANT DISEASE CLASSIFICATION USING RESNET-9 🏵🏵
### Corresponding Kaggle notebook can be accessed [here](https://www.kaggle.com/atharvaingle/plant-disease-classification-resnet-99-2)
##### 🏵🏵🏵🏵🏵🏵DISCLAIMER: This notebook is beginner friendly, so don't worry if you don't know much about CNNs and Pytorch. Even if you have used TensorFlow in the past and are new to PyTorch, hang in there, everything is explained clearly and concisely. You will get a good overview of how to use PyTorch for image classification problems.
# Description of the dataset
This dataset is created using offline augmentation from the original dataset. The original PlantVillage Dataset can be found [here](https://github.com/spMohanty/PlantVillage-Dataset).This dataset consists of about 87K rgb images of healthy and diseased crop leaves which is categorized into 38 different classes. The total dataset is divided into 80/20 ratio of training and validation set preserving the directory structure. A new directory containing 33 test images is created later for prediction purpose.

Note: This description is given in the dataset itself
# Our goal
Goal is clear and simple. We need to build a model, which can classify between healthy and diseased crop leaves and also if the crop have any disease, predict which disease is it.
##### Let's get started....
## Importing necessary libraries
Let's import required modules
!pip install torchsummary
We would require torchsummary library to print the model's summary in keras style (nicely formatted and pretty to look) as Pytorch natively doesn't support that
import os                       # for working with files
import numpy as np              # for numerical computationss
import pandas as pd             # for working with dataframes
import torch                    # Pytorch module
import matplotlib.pyplot as plt # for plotting informations on graph and images using tensors
import torch.nn as nn           # for creating neural networks
from torch.utils.data import DataLoader # for dataloaders
from PIL import Image           # for checking images
import torch.nn.functional as F # for functions for calculating loss
import torchvision.transforms as transforms   # for transforming images into tensors
from torchvision.utils import make_grid       # for data checking
from torchvision.datasets import ImageFolder # for working with classes and images
from torchsummary import summary              # for getting the summary of our model

%matplotlib inline
#  Exploring the data
Loading the data
data_dir = "../input/new-plant-diseases-dataset/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)"
train_dir = data_dir + "/train"
valid_dir = data_dir + "/valid"
diseases = os.listdir(train_dir)
# printing the disease names
print(diseases)
print("Total disease classes are: {}".format(len(diseases)))
plants = []
NumberOfDiseases = 0
for plant in diseases:
```

```
    if plant.split('_')[0] not in plants:
        plants.append(plant.split('__')[0])
    if plant.split('_')[1] != 'healthy':
        NumberOfDiseases += 1
```

The above cell extract the number of unique plants and number of unique diseases

```
# unique plants in the dataset
print(f"Unique Plants are: \n{plants}")
# number of unique plants
print("Number of plants: {}".format(len(plants)))
# number of unique diseases
print("Number of diseases: {}".format(NumberOfDiseases))
```

So we have images of leaves of 14 plants and while excluding healthy leaves, we have 26 types of images that show a particular disease in a particular plant.

```
# Number of images for each disease
nums = {}
for disease in diseases:
    nums[disease] = len(os.listdir(train_dir + '/' + disease))

# converting the nums dictionary to pandas dataframe passing index as plant name and number of images as column

img_per_class = pd.DataFrame(nums.values(), index=nums.keys(), columns=["no. of images"])
img_per_class
```

Visualizing the above information on a graph

```
# plotting number of images available for each disease
index = [n for n in range(38)]
plt.figure(figsize=(20, 5))
plt.bar(index, [n for n in nums.values()], width=0.3)
plt.xlabel('Plants/Diseases', fontsize=10)
plt.ylabel('No of images available', fontsize=10)
plt.xticks(index, diseases, fontsize=5, rotation=90)
plt.title('Images per each class of plant disease')
```

We can see that the dataset is almost balanced for all classes, so we are good to go forward

Images available for training

```
n_train = 0
for value in nums.values():
    n_train += value
print(f"There are {n_train} images for training")
```

Data Preparation for training

```
# datasets for validation and training
train = ImageFolder(train_dir, transform=transforms.ToTensor())
valid = ImageFolder(valid_dir, transform=transforms.ToTensor())
```

`torchvision.datasets` is a class which helps in loading all common and famous datasets. It also helps in loading custom datasets. I have used subclass `torchvision.datasets.ImageFolder` which helps in loading the image data when the data is arranged in this way:

```
--------------------------
root/dog/xxx.png
root/dog/xxy.png
root/dog/xxz.png
<br>
root/cat/123.png
root/cat/nsdf3.png
root/cat/asd932_.png
--------------------------
```

Next, after loading the data, we need to transform the pixel values of each image (0-255) to 0-1 as neural networks works quite good with normalized data. The entire array of pixel values is converted to torch [tensor](https://pytorch.org/tutorials/beginner/examples_tensor/two_layer_net_tensor.html#:~:text=A%20PyTorch%20Tensor%20is%20basically,used%20for%20arbitrary%20numeric%20computation.) and then divided by 255.If you are not familiar why normalizing inputs help neural network, read [this](https://towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d) post.

Image shape

```
img, label = train[0]
print(img.shape, label)
```

We can see the shape (3, 256 256) of the image. 3 is the number of channels (RGB) and 256 x 256 is the width and height of the image

```
# total number of classes in train set
len(train.classes)
# for checking some images from training dataset
def show_image(image, label):
    print("Label :" + train.classes[label] + "(" + str(label) + ")")
    plt.imshow(image.permute(1, 2, 0))
## ♔ Some Images from training dataset ♔
show_image(*train[0])
show_image(*train[70000])
show_image(*train[30000])
# Setting the seed value
random_seed = 7
torch.manual_seed(random_seed)
# setting the batch size
batch_size = 32
```
`batch_size` is the total number of images given as input at once in forward propagation of the CNN. Basically, batch size defines the number of samples that will be propagated through the network.

For instance, let's say you have 1050 training samples and you want to set up a batch_size equal to 100. The algorithm takes the first 100 samples (from 1st to 100th) from the training dataset and trains the network. Next, it takes the second 100 samples (from 101st to 200th) and trains the network again. We can keep doing this procedure until we have propagated all samples through of the network.

```
# DataLoaders for training and validation
train_dl = DataLoader(train, batch_size, shuffle=True, num_workers=2, pin_memory=True)
valid_dl = DataLoader(valid, batch_size, num_workers=2, pin_memory=True)
```
- `DataLoader` is a subclass which comes from `torch.utils.data`. It helps in loading large and memory consuming datasets. It takes in `batch_size` which denotes the number of samples contained in each generated batch.

- Setting `shuffle=True` shuffles the dataset. It is heplful so that batches between epochs do not look alike. Doing so will eventually make our model more robust.

- `num_workers`, denotes the number of processes that generate batches in parallel. If you have more cores in your CPU, you can set it to number of cores in your CPU. Since, Kaggle provides a 2 core CPU, I have set it to 2

```
# helper function to show a batch of training instances
def show_batch(data):
    for images, labels in data:
        fig, ax = plt.subplots(figsize=(30, 30))
        ax.set_xticks([]); ax.set_yticks([])
        ax.imshow(make_grid(images, nrow=8).permute(1, 2, 0))
        break
# Images for first batch of training
show_batch(train_dl)
# ♔ Modelling ♔
```
It is advisable to use GPU instead of CPU when dealing with images dataset because CPUs are generalized for general purpose and GPUs are optimized for training deep learning models as they can process multiple computations simultaneously. They have a large number of cores, which allows for better computation of multiple parallel processes. Additionally, computations in deep learning need to handle huge amounts of data — this makes a GPU’s memory bandwidth most suitable.
To seamlessly use a GPU, if one is available, we define a couple of helper functions (`get_default_device` & `to_device`) and a helper class `DeviceDataLoader` to move our model & data to the GPU as required
Some helper functions
```
# for moving data into GPU (if available)
def get_default_device():
    """Pick GPU if available, else CPU"""
    if torch.cuda.is_available:
        return torch.device("cuda")
    else:
        return torch.device("cpu")

# for moving data to device (CPU or GPU)
def to_device(data, device):
    """Move tensor(s) to chosen device"""
```

```
    if isinstance(data, (list,tuple)):
        return [to_device(x, device) for x in data]
    return data.to(device, non_blocking=True)

# for loading in the device (GPU if available else CPU)
class DeviceDataLoader():
    """Wrap a dataloader to move data to a device"""
    def __init__(self, dl, device):
        self.dl = dl
        self.device = device

    def __iter__(self):
        """Yield a batch of data after moving it to device"""
        for b in self.dl:
            yield to_device(b, self.device)

    def __len__(self):
        """Number of batches"""
        return len(self.dl)
Checking the device we are working with
device = get_default_device()
device
Wrap up our training and validation data loaders using `DeviceDataLoader` for automatically transferring
batches of data to the GPU (if available)
# Moving data into GPU
train_dl = DeviceDataLoader(train_dl, device)
valid_dl = DeviceDataLoader(valid_dl, device)
##  Building the model architecture
*We are going to use **ResNet**, which have been one of the major breakthrough in computer vision since they
were introduced in 2015.*
If you want to learn more about ResNets read the following articles:
- [Understanding and Visualizing ResNets](https://towardsdatascience.com/understanding-and-visualizing-
resnets-442284831be8#:~:text=ResNet%20Layers,layers%20remains%20the%20same%20%E2%80%94%204.)
- [Overview of ResNet and its variants](https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-
5281e2f56035)
- [Paper with code implementation](https://paperswithcode.com/method/resnet)
In ResNets, unlike in traditional neural networks, each layer feeds into the next layer, we use a network with
residual blocks, each layer feeds into the next layer and directly into the layers about 2–3 hops away, to
avoid over-fitting (a situation when validation loss stop decreasing at a point and then keeps increasing
while training loss still decreases). This also helps in preventing [vanishing gradient
problem](https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484) and allow us to train
deep neural networks. Here is a simple residual block:
![image](https://www.mdpi.com/remotesensing/remotesensing-11-01896/article_deploy/html/images/remotesensing-
11-01896-g001.png)
#### Residual Block code implementation
class SimpleResidualBlock(nn.Module):
    def_init_(self):
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels=3, out_channels=3, kernel_size=3, stride=1, padding=1)
        self.relu1 = nn.ReLU()
        self.conv2 = nn.Conv2d(in_channels=3, out_channels=3, kernel_size=3, stride=1, padding=1)
        self.relu2 = nn.ReLU()

    def  forward(self,  x):
        out = self.conv1(x)
        out = self.relu1(out)
        out = self.conv2(out)
        return self.relu2(out) + x # ReLU can be applied before or after adding the input
**Then we define our `ImageClassificationBase` class whose functions are:**

- `training_step` - To figure out how “wrong” the model is going after training or validation step.We are
using this function other than just an accuracy metric that is likely not going to be differentiable (this
would mean that the gradient can’t be determined, which is necessary for the model to improve during training)
```

A quick look at the PyTorch docs that yields the cost function:
[cross_entropy](https://pytorch.org/docs/stable/nn.functional.html#cross-entropy).

- `validation_step` - Because an accuracy metric can't be used while training the model, doesn't mean it shouldn't be implemented! Accuracy in this case would be measured by a threshold, and counted if the difference between the model's prediction and the actual label is lower than that threshold.
- `validation_epoch_end` - We want to track the validation losses/accuracies and train losses after each epoch, and every time we do so we have to make sure the gradient is not being tracked.
- `epoch_end` - We also want to print validation losses/accuracies, train losses and learning rate too because we are using learning rate scheduler (which will change the learning rate after every batch of training) after each epoch.

We also define an `accuracy` function which calculates the overall accuracy of the model on an entire batch of outputs, so that we can use it as a metric in `fit_one_cycle`

```
# for calculating the accuracy
def accuracy(outputs, labels):
    _, preds = torch.max(outputs, dim=1)
    return torch.tensor(torch.sum(preds == labels).item() / len(preds))


# base class for the model
class ImageClassificationBase(nn.Module):

    def training_step(self, batch):
        images, labels = batch
        out = self(images)                  # Generate predictions
        loss = F.cross_entropy(out, labels) # Calculate loss
        return loss

    def validation_step(self, batch):
        images, labels = batch
        out = self(images)                    # Generate prediction
        loss = F.cross_entropy(out, labels)  # Calculate loss
        acc = accuracy(out, labels)           # Calculate accuracy
        return {"val_loss": loss.detach(), "val_accuracy": acc}

    def validation_epoch_end(self, outputs):
        batch_losses = [x["val_loss"] for x in outputs]
        batch_accuracy = [x["val_accuracy"] for x in outputs]
        epoch_loss = torch.stack(batch_losses).mean()       # Combine loss
        epoch_accuracy = torch.stack(batch_accuracy).mean()
        return {"val_loss": epoch_loss, "val_accuracy": epoch_accuracy} # Combine accuracies

    def epoch_end(self, epoch, result):
        print("Epoch [{}], last_lr: {:.5f}, train_loss: {:.4f}, val_loss: {:.4f}, val_acc: {:.4f}".format(
            epoch, result['lrs'][-1], result['train_loss'], result['val_loss'], result['val_accuracy']))

##  Defining the final architecture of our model #
Architecture for training

# convolution block with BatchNormalization
def ConvBlock(in_channels, out_channels, pool=False):
    layers = [nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
              nn.BatchNorm2d(out_channels),
              nn.ReLU(inplace=True)]
    if pool:
        layers.append(nn.MaxPool2d(4))
    return nn.Sequential(*layers)


# resnet architecture
class ResNet9(ImageClassificationBase):
    def_init_(self, in_channels, num_diseases):
        super()._init_()
```

```
        self.conv1 = ConvBlock(in_channels, 64)
        self.conv2 = ConvBlock(64, 128, pool=True) # out_dim : 128 x 64 x 64
        self.res1 = nn.Sequential(ConvBlock(128, 128), ConvBlock(128, 128))

        self.conv3 = ConvBlock(128, 256, pool=True) # out_dim : 256 x 16 x 16
        self.conv4 = ConvBlock(256, 512, pool=True) # out_dim : 512 x 4 x 44
        self.res2 = nn.Sequential(ConvBlock(512, 512), ConvBlock(512, 512))

        self.classifier = nn.Sequential(nn.MaxPool2d(4),
                                        nn.Flatten(),
                                        nn.Linear(512, num_diseases))

    def forward(self, xb): # xb is the loaded batch
        out = self.conv1(xb)
        out = self.conv2(out)
        out = self.res1(out) + out
        out = self.conv3(out)
        out = self.conv4(out)
        out = self.res2(out) + out
        out = self.classifier(out)
        return out
```

Now, we define a model object and transfer it into the device with which we are working...

```
# defining the model and moving it to the GPU
model = to_device(ResNet9(3, len(train.classes)), device)
model
```

Getting a nicely formatted summary of our model (like in Keras). Pytorch doesn't support it natively. So, we need to install the `torchsummary` library (discussed earlier)

```
# getting summary of the model
INPUT_SHAPE = (3, 256, 256)
print(summary(model.cuda(), (INPUT_SHAPE)))
```

♔ Training the model ♔

Before we train the model, Let’s define a utility functionan `evaluate` function, which will perform the validation phase, and a `fit_one_cycle` function which will perform the entire training process. In `fit_one_cycle`, we have use some techniques:

- **Learning Rate Scheduling**: Instead of using a fixed learning rate, we will use a learning rate scheduler, which will change the learning rate after every batch of training. There are many strategies for varying the learning rate during training, and the one we’ll use is called the *“One Cycle Learning Rate Policy”*, which involves starting with a low learning rate, gradually increasing it batch-by-batch to a high learning rate for about 30% of epochs, then gradually decreasing it to a very low value for the remaining epochs.

- **Weight Decay**: We also use weight decay, which is a regularization technique which prevents the weights from becoming too large by adding an additional term to the loss function.

- **Gradient Clipping**: Apart from the layer weights and outputs, it also helpful to limit the values of gradients to a small range to prevent undesirable changes in parameters due to large gradient values. This simple yet effective technique is called gradient clipping.

We'll also record the learning rate used for each batch.

```
# for training
@torch.no_grad()
def evaluate(model, val_loader):
    model.eval()
    outputs = [model.validation_step(batch) for batch in val_loader]
    return model.validation_epoch_end(outputs)


def get_lr(optimizer):
    for param_group in optimizer.param_groups:
        return param_group['lr']


def fit_OneCycle(epochs, max_lr, model, train_loader, val_loader, weight_decay=0,
                 grad_clip=None, opt_func=torch.optim.SGD):
    torch.cuda.empty_cache()
    history = []
```

```
    optimizer = opt_func(model.parameters(), max_lr, weight_decay=weight_decay)
    # scheduler for one cycle learniing rate
    sched = torch.optim.lr_scheduler.OneCycleLR(optimizer, max_lr, epochs=epochs,
steps_per_epoch=len(train_loader))


    for epoch in range(epochs):
        # Training
        model.train()
        train_losses = []
        lrs = []
        for batch in train_loader:
            loss = model.training_step(batch)
            train_losses.append(loss)
            loss.backward()

            # gradient clipping
            if grad_clip:
                nn.utils.clip_grad_value_(model.parameters(), grad_clip)

            optimizer.step()
            optimizer.zero_grad()

            # recording and updating learning rates
            lrs.append(get_lr(optimizer))
            sched.step()


        # validation
        result = evaluate(model, val_loader)
        result['train_loss'] = torch.stack(train_losses).mean().item()
        result['lrs'] = lrs
        model.epoch_end(epoch, result)
        history.append(result)

    return history
```

Let's check our validation loss and accuracy

```
%%time
history = [evaluate(model, valid_dl)]
history
```

Since there are randomly initialized weights, that is why accuracy come to near 0.019 (that is 1.9% chance of getting the right answer or you can say model randomly chooses a class).

Now, declare some hyper parameters for the training of the model. We can change it if result is not satisfactory.

```
epochs = 2
max_lr = 0.01
grad_clip = 0.1
weight_decay = 1e-4
opt_func = torch.optim.Adam
```

Let's start training our model ....

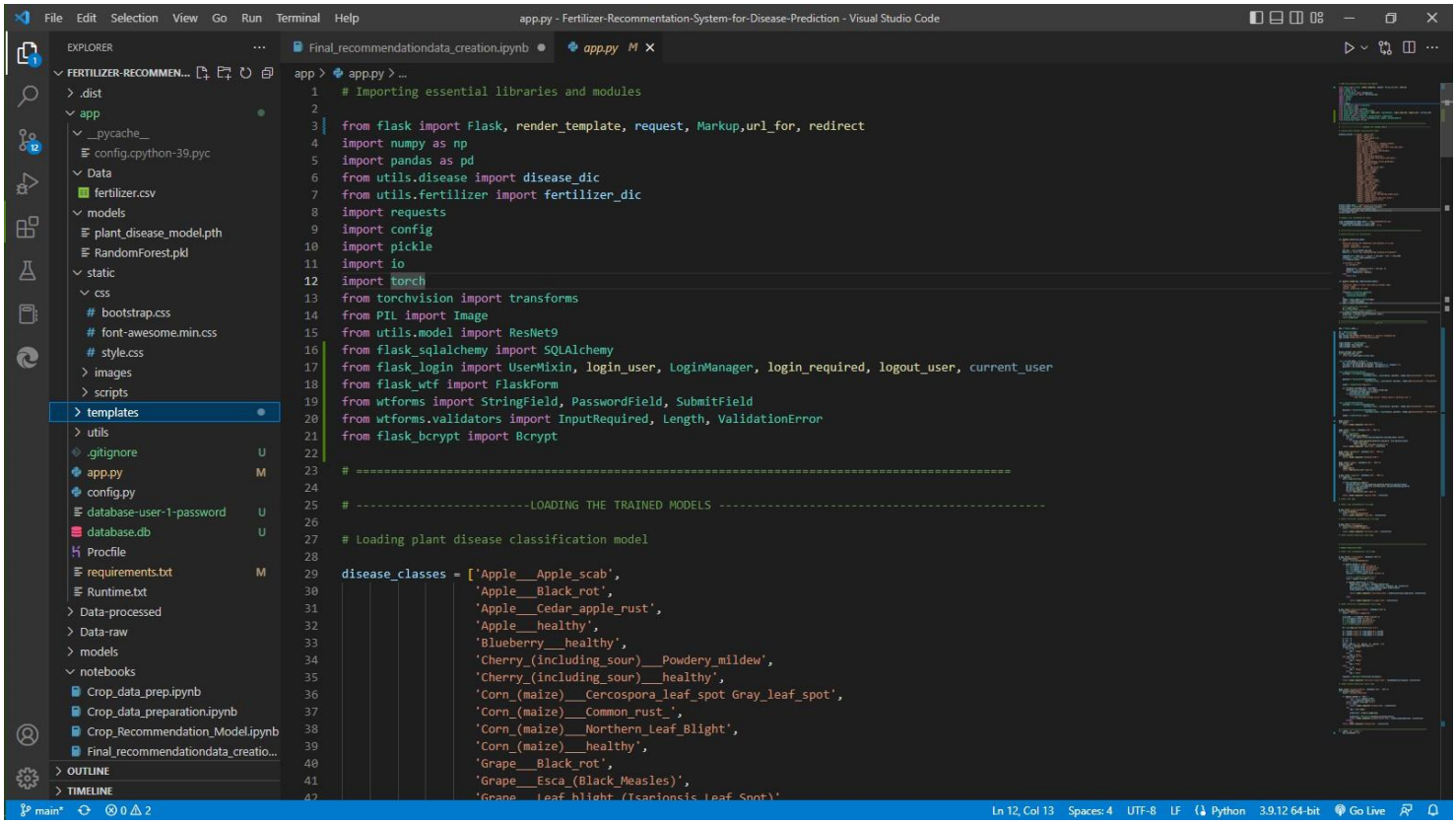
Note: The following cell may take 15 mins to 45 mins to run depending on your GPU. In kaggle (P100 GPU) it took around 20 mins of Wall Time.

```
%%time
history += fit_OneCycle(epochs, max_lr, model, train_dl, valid_dl,
                             grad_clip=grad_clip,
                             weight_decay=1e-4,
                             opt_func=opt_func)
### We got an accuracy of 99.2 % #
Plotting
#### Helper functions for plotting
def plot_accuracies(history):
    accuracies = [x['val_accuracy'] for x in history]
```

```
    plt.plot(accuracies, '-x')
    plt.xlabel('epoch')
    plt.ylabel('accuracy')
    plt.title('Accuracy vs. No. of epochs');

def plot_losses(history):
    train_losses = [x.get('train_loss') for x in history]
    val_losses = [x['val_loss'] for x in history]
    plt.plot(train_losses, '-bx')
    plt.plot(val_losses, '-rx')
    plt.xlabel('epoch')
    plt.ylabel('loss')
    plt.legend(['Training', 'Validation'])
    plt.title('Loss vs. No. of epochs');

def plot_lrs(history):
    lrs = np.concatenate([x.get('lrs', []) for x in history])
    plt.plot(lrs)
    plt.xlabel('Batch no.')
    plt.ylabel('Learning rate')
    plt.title('Learning Rate vs. Batch no.');
## Validation Accuracy
plot_accuracies(history)
## Validation loss
plot_losses(history)
## Learning Rate overtime
plot_lrs(history)
#  Testing model on test data
**We only have 33 images in test data, so let's check the model on all images**
test_dir = "../input/new-plant-diseases-dataset/test"
test = ImageFolder(test_dir, transform=transforms.ToTensor())
test_images = sorted(os.listdir(test_dir + '/test')) # since images in test folder are in alphabetical order
test_images
def predict_image(img, model):
    """Converts image to array and return the predicted class
        with highest probability"""
    # Convert to a batch of 1
    xb = to_device(img.unsqueeze(0), device)
    # Get predictions from model
    yb = model(xb)
    # Pick index with highest probability
    _, preds  = torch.max(yb, dim=1)
    # Retrieve the class label

    return train.classes[preds[0].item()]
# predicting first image
img, label = test[0]
plt.imshow(img.permute(1, 2, 0))
print('Label:', test_images[0], ', Predicted:', predict_image(img, model))
# getting all predictions (actual label vs predicted)
for i, (img, label) in enumerate(test):
    print('Label:', test_images[i], ', Predicted:', predict_image(img, model))
**We can see that the model predicted all the test images perfectly!!!!**
# Saving the model
**There are several ways to save the model in Pytorch, following are the two most common ways**
1. **Save/Load `state_dict` (Recommended)**

When saving a model for inference, it is only necessary to save the trained model’s learned parameters. Saving
the model’s `state_dict` with the `torch.save()` function will give you the most flexibility for restoring the
model later, which is why it is the recommended method for saving models.

A common PyTorch convention is to save models using either a `.pt` or `.pth` file extension.

Remember that you must call `model.eval()` to set dropout and batch normalization layers to evaluation mode
before running inference. Failing to do this will yield inconsistent inference results.
```

```
# saving to the kaggle working directory
PATH = './plant-disease-model.pth'
torch.save(model.state_dict(), PATH)
2. **Save/Load Entire Model**

This save/load process uses the most intuitive syntax and involves the least amount of code. Saving a model in this way will save the entire module using Python’s [pickle](https://docs.python.org/3/library/pickle.html) module. The disadvantage of this approach is that the serialized data is bound to the specific classes and the exact directory structure used when the model is saved. The reason for this is because pickle does not save the model class itself. Rather, it saves a path to the file containing the class, which is used during load time. Because of this, your code can break in various ways when used in other projects or after refactors.
# saving the entire model to working directory
PATH = './plant-disease-model-complete.pth'
torch.save(model, PATH)
# Conclusion
ResNets perform significantly well for image classification when some of the parameters are tweaked and techniques like scheduling learning rate, gradient clipping and weight decay are applied. The model is able to predict every image in test set perfectly without any errors !!!!
```

Project Structure :



Github Link : https://github.com/IBM-EPBL/IBM-Project-19227-1659694626

Project Demonstration Video Link : https://youtu.be/9Uiou95JVmU