

# Fertilizers Recommendation System for Disease Prediction

## 1 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.

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.

### 1.2 Purpose

As per Project objectives to achieve an automated system to identify different diseases in plants there are some pre-requisites in order to make the image dataset ready for training and testing using deep learning techniques. In order to meet the basic requirements:

- Different fruit and vegetable leaf images are pre-processed using Image data augmentation. it is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class. The first step is usually importing the libraries that will be needed in the program. Import Keras library from that library, import the ImageDataGenerator Library to your Python script.
- Applied the CNN algorithm to the dataset to train and test the model.
- Deep neural networks used to detect the disease.
- Accuracy of the model is calculated for both fruit and vegetable datasets.
- used the Flask framework to build web applications.

## 2 LITERATURE SURVEY

### 2.1 Existing problem

The major agricultural products in India are rice, wheat, pulses, and spices. As our population is increasing rapidly the demand for agriculture products also increasing alarmingly. A huge amount of data is incremented from various field of agriculture. Analysis of this data helps in predicting the crop yield, analysing soil quality, predicting disease in a plant, and how meteorological factor affects crop productivity. Crop protection plays a vital role in maintaining agriculture product. Pathogen, pest, weed, and animals are responsible for the productivity loss in agriculture product. Machine learning techniques like Random Forest, Bayesian Network, Decision Tree, Support Vector Machine etc. help in automatic detection of plant disease from visual symptoms in the plant.

## **2.2 Proposed solution**

Automatic detection of disease in plant helps in early diagnosis and prevention of disease which leads to an increase in agriculture productivity. Under this proposed solution:

A web Application is built where:

- Farmers interact with the portal build
- Interacts with the user interface to upload images of diseased leaf
- Our model-built analyses the Disease and suggests the farmer which fertilizers are to be used

To accomplish the above task below activities and tasks are performed the documents of which (notebooks and final model) are also attached in github

- Download the dataset.
- Classify the dataset into train and test sets.
- Add the neural network layers.
- Load the trained images and fit the model.
- Test the model.

- Save the model and its dependencies.
- Build a Web application using a flask that integrates with the model built.

### 3 THEORITICAL ANALYSIS

#### 3.1 Block diagram

The Diagrammatic overview of the project is as shown in figure-1.

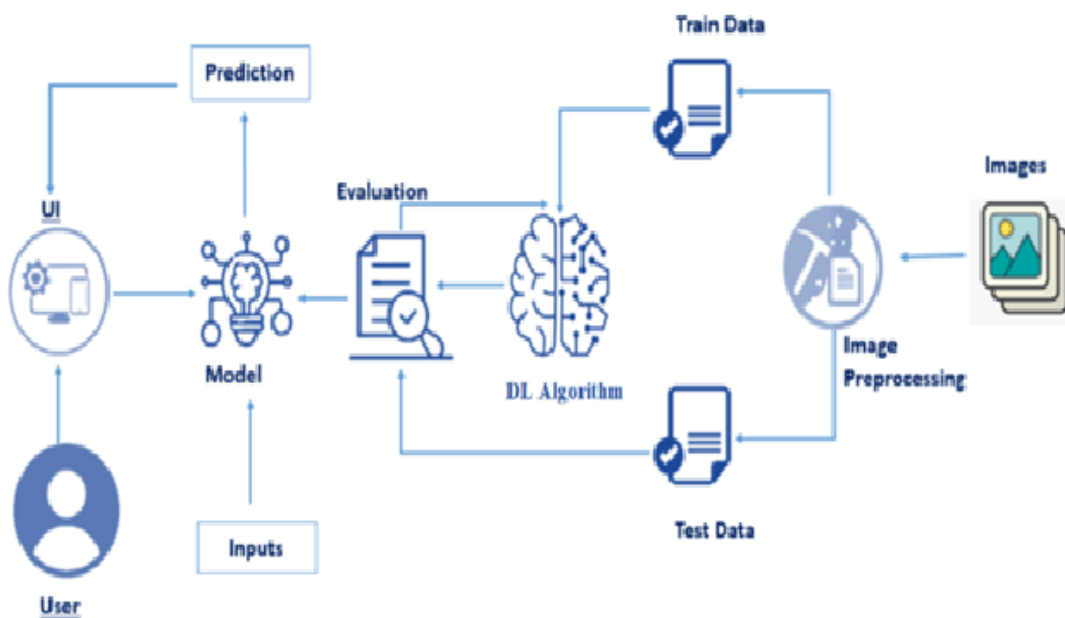


Figure-1 : Diagrammatic overview of the project

### 3.2 Hardware / Software designing

Anaconda Navigator is the prerequisites to complete this project:

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning-related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, I will be using Jupyter notebook and spyder.

To build Deep learning models we require the following packages

**Tensor flow:** TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML powered applications.

**Keras:** Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

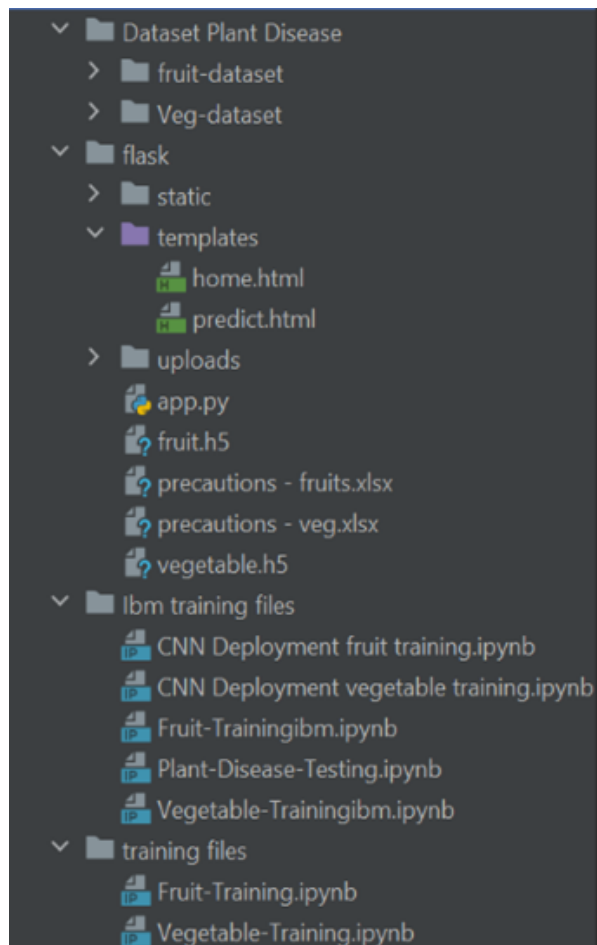
- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user-friendly framework that runs on both CPU and GPU.
- Highly scalability of computation.
- Flask: Web framework used for building Web applications

## 4 EXPERIMENTAL INVESTIGATIONS

In order to increase the model accuracy, we need to increase the number of epochs or reduce the batch size used. In our experiment for fruit dataset batch size used is 32 and for vegetable dataset batch size used is 16

## 5 CONTROL FLOW OF THE SOLUTION IS AS FOLLOWS

Plant disease dataset is downloaded then fruit and vegetable testing and training is performed and to improve accuracy increased the number of epochs or reduce the batch size approach is used. In our experiment for fruit dataset batch size used is 32 and for vegetable dataset batch size used is 16. Then spyder IDE is used for running the flask code for creating user interface of the model. In which saved state of model in the form of fruit.h5 and vegetable.h5 files is used.. Figure-2 shows the control flow of different folders and files created and used to achieve the outcomes.

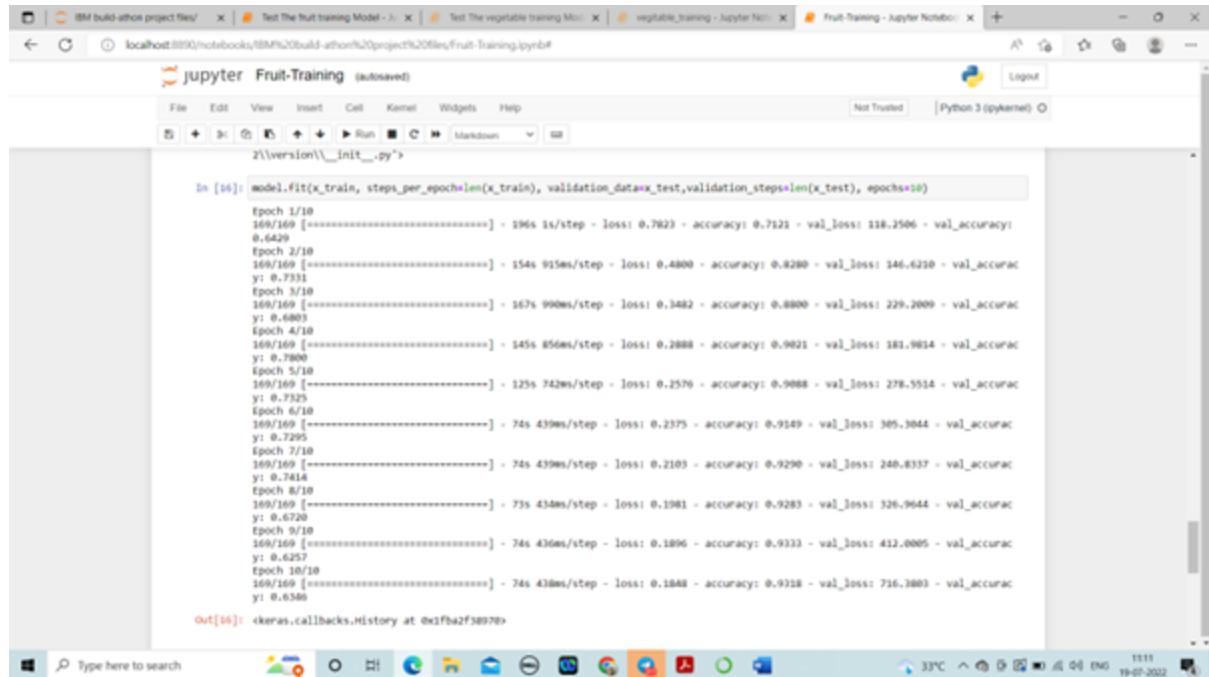


## 6 RESULT

Fruit and Vegetable data set training results

Fruit training Accuracy: 0.9318 = 93%

Vegetable training Accuracy : 0.8132 = 81%



```
2:\version\__init__.py">

In [16]: model.fit(x_train, steps_per_epoch=len(x_train), validation_data=x_test, validation_steps=len(x_test), epochs=10)

Epoch 1/10
169/169 [=====] - 196s 1s/step - loss: 0.7823 - accuracy: 0.7121 - val_loss: 118.2506 - val_accuracy: 0.6429
Epoch 2/10
169/169 [=====] - 154s 915ms/step - loss: 0.4800 - accuracy: 0.8280 - val_loss: 146.6210 - val_accuracy: 0.7331
Epoch 3/10
169/169 [=====] - 167s 990ms/step - loss: 0.3482 - accuracy: 0.8800 - val_loss: 229.2009 - val_accuracy: 0.6803
Epoch 4/10
169/169 [=====] - 145s 856ms/step - loss: 0.2888 - accuracy: 0.9021 - val_loss: 181.9814 - val_accuracy: 0.7800
Epoch 5/10
169/169 [=====] - 125s 742ms/step - loss: 0.2576 - accuracy: 0.9088 - val_loss: 278.5514 - val_accuracy: 0.7125
Epoch 6/10
169/169 [=====] - 74s 439ms/step - loss: 0.2375 - accuracy: 0.9149 - val_loss: 305.3044 - val_accuracy: 0.7295
Epoch 7/10
169/169 [=====] - 74s 439ms/step - loss: 0.2103 - accuracy: 0.9290 - val_loss: 240.8337 - val_accuracy: 0.7414
Epoch 8/10
169/169 [=====] - 73s 434ms/step - loss: 0.1981 - accuracy: 0.9283 - val_loss: 326.9644 - val_accuracy: 0.6720
Epoch 9/10
169/169 [=====] - 74s 436ms/step - loss: 0.1896 - accuracy: 0.9333 - val_loss: 412.0005 - val_accuracy: 0.6257
Epoch 10/10
169/169 [=====] - 74s 438ms/step - loss: 0.1848 - accuracy: 0.9318 - val_loss: 716.3803 - val_accuracy: 0.6346

Out[16]: <keras.callbacks.History at 0x1fb2f38979>
```

```
localhost:8890/notebooks/IBM%20build-athon%20project%20files/vegetable_training.ipynb
jupyter vegetable_training Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

In [31]: model=Sequential()
In [32]: model.add(Convolution2D(32,(4,4), input_shape=(128,128,3), activation='relu'))
In [33]: model.add(MaxPooling2D(pool_size=(2,2)))
In [34]: model.add(Flatten())
In [35]: model.add(Dense(300, kernel_initializer='uniform', activation='relu'))
model.add(Dense(150, kernel_initializer='uniform', activation='relu'))
model.add(Dense(75, kernel_initializer='uniform', activation='relu'))
In [36]: model.add(Dense(9, kernel_initializer='uniform', activation='softmax'))
In [37]: model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
In [38]: model.fit(x_train, steps_per_epoch=89, validation_data=x_test, validation_steps=27, epochs=20)

Epoch 1/20
89/89 [-----] - 41s 458ms/step - loss: 1.6310 - accuracy: 0.3806 - val_loss: 187.7353 - val_accuracy:
0.4560
Epoch 2/20
89/89 [-----] - 43s 483ms/step - loss: 1.4238 - accuracy: 0.4767 - val_loss: 164.9139 - val_accuracy:
0.3380
Epoch 3/20
89/89 [-----] - 41s 461ms/step - loss: 1.3073 - accuracy: 0.5098 - val_loss: 189.4257 - val_accuracy:
0.5278
Epoch 4/20
89/89 [-----] - 43s 472ms/step - loss: 1.1580 - accuracy: 0.5882 - val_loss: 250.1262 - val_accuracy:
0.3806
Epoch 5/20
89/89 [-----] - 40s 447ms/step - loss: 0.9880 - accuracy: 0.6362 - val_loss: 433.7624 - val_accuracy:
0.3472
```

```
localhost:8890/notebooks/IBM%20build-athon%20project%20files/vegetable_training.ipynb
jupyter vegetable_training Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Out[38]: <keras.callbacks.History at 0x2765b8bb8>
In [43]: model.save('vegetable.h5')
In [44]: model.summary()

Model: "sequential_2"
Layer (type) Output Shape Param #
-----
conv2d_2 (Conv2D) (None, 125, 125, 32) 1568
max_pooling2d_2 (MaxPooling (None, 62, 62, 32) 0
2D)
flatten_2 (Flatten) (None, 123008) 0
dense_8 (Dense) (None, 300) 36902700
dense_9 (Dense) (None, 150) 45150
dense_10 (Dense) (None, 75) 11325
dense_11 (Dense) (None, 9) 684
-----
Total params: 36,961,427
Trainable params: 36,961,427
Non-trainable params: 0

In [ ]:
```

Fruit dataset model testing results :

```
localhost:8890/notebooks/IBM%20build-athon%20project%20files/Test%20The%20fruit%20training%20Model.ipynb

jupyter Test The fruit training Model (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

[162., 161., 166.],
....
[173., 172., 178.],
[173., 172., 178.],
[178., 173., 179.]]], dtype=float32)

In [23]: y=np.argmax(model.predict(x), axis=1)

In [38]: y
Out[38]: array([3], dtype=int64)

In [27]: index=['Apple__Black_rot', 'Apple__healthy', 'Corn_(maize)__Northern_Leaf_Blight', 'Corn_(maize)__healthy',
              'Peach__Bacterial_spot', 'Peach__healthy']

In [28]: index[y[0]]
Out[28]: 'Apple__healthy'

In [33]: img=image.load_img('C:/Users/Dell/Desktop/Dataset Plant Disease/fruit-dataset/fruit-dataset/test/Peach__Bacterial_spot/ba05b466-
              ')

In [35]: x=image.img_to_array(img)
          x=np.expand_dims(x,axis=0)
          y=np.argmax(model.predict(x), axis=1)

In [40]: index[y[0]]
Out[40]: 'Corn_(maize)__healthy'

In [41]: y
Out[41]: array([3], dtype=int64)
```

Vegetable model testing results :

```
localhost:8890/notebooks/IBM%20build-athon%20project%20files/Test%20The%20vegetable%20training%20Model.ipynb

jupyter Test The vegetable training Model Last Checkpoint an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

import numpy as np

In [3]: model=load_model("vegetable.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.

In [12]: img=image.load_img('C:/Users/Dell/Desktop/Dataset Plant Disease/veg-dataset/Veg-dataset/test_set/Pepper,_bell___healthy/b61a5e27-
              ')

In [13]: x=image.img_to_array(img)

In [14]: x=np.expand_dims(x,axis=0)

In [15]: x
Out[15]: array([[[[151., 136., 141.],
                  [163., 148., 153.],
                  [148., 133., 138.],
                  ...,
                  [161., 149., 153.],
                  [160., 148., 152.],
                  [168., 152., 156.]],
                  [[150., 135., 140.],
                  [146., 131., 136.],
                  [147., 132., 137.],
                  ...,
                  [157., 145., 149.],
                  [162., 150., 154.],
                  [169., 157., 161.]],
                  [[152., 137., 142.],
                  [154., 139., 144.],
                  [140., 134., 139.]]]])])
```



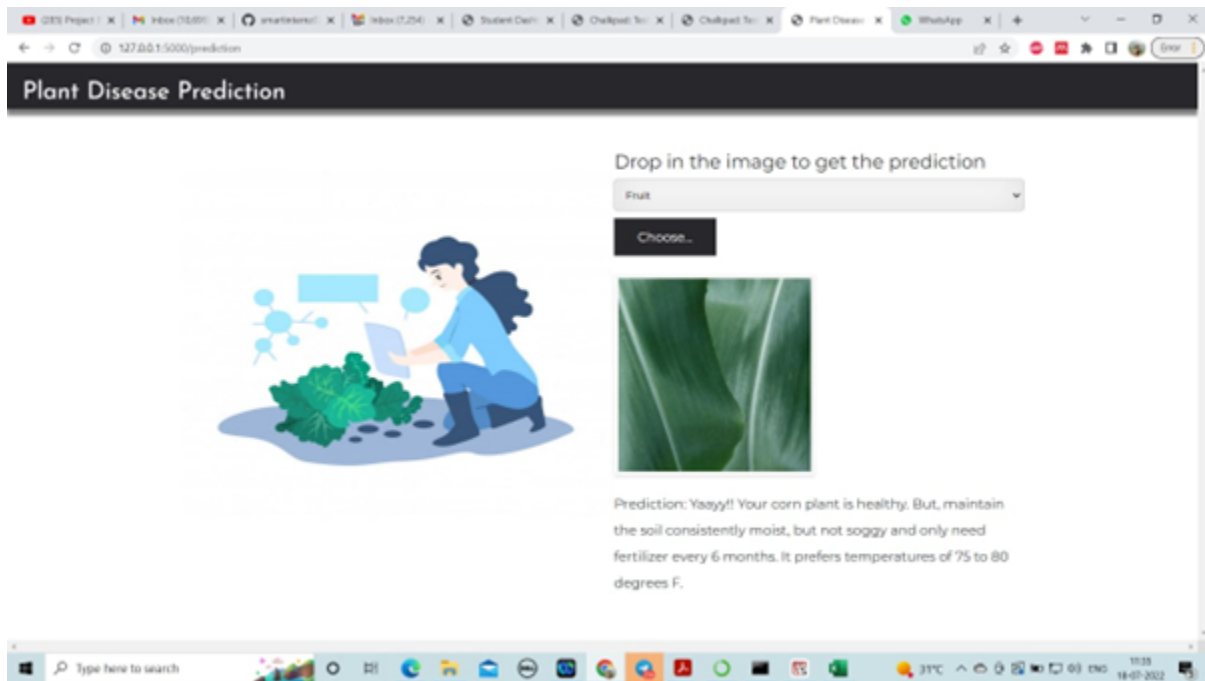
This screenshot shows a Jupyter Notebook titled "Test The vegetable training Model". The notebook is running on a local host. The code in the cells performs the following steps:

- Input data `x` is defined as a 2D array: `[[201., 192., 198.], [142., 131., 137.]]` with dtype `float32`.
- Cell 16: `ymp.argmax(model.predict(x), axis=1)`
- Cell 17: `y` is assigned the result of the previous cell, showing `array([1], dtype=int64)`.
- Cell 18: A list of disease names is defined: `index = ["Pepper__bell__bacterial_spot", "Pepper__bell__healthy", "Potato__early_blight", "Potato__late_blight", "Potato__healthy", "Tomato__bacterial_spot", "Tomato__late_blight", "Tomato__leaf_mold", "Tomato__septoria_leaf_spot"]`.
- Cell 19: `index[y[0]]` is used to retrieve the predicted class, showing `"Pepper__bell__healthy"`.
- Cell 20: An image is loaded from the file path `'C:/Users/Dell/Desktop/Dataset Plant Disease/Veg-dataset/Veg-dataset/test_set/Tomato__septoria_leaf_spot/c68M'`.
- Cell 21: The image is converted to an array and expanded to 3 channels: `ximage = img_to_array(img)`, `xmp.expand_dims(x, axis=0)`, and `ymp.argmax(model.predict(x), axis=1)`.
- Cell 22: The same list of disease names is repeated, and the predicted class is shown as `"Potato__healthy"`.

This screenshot shows the same Jupyter Notebook at a later stage. The code in the cells performs the following steps:

- Cell 23: An image is loaded from the file path `'C:/Users/Dell/Desktop/Dataset Plant Disease/Veg-dataset/Veg-dataset/test_set/Tomato__septoria_leaf_spot/c68M'`.
- Cell 24: The image is converted to an array and expanded to 3 channels: `ximage = img_to_array(img)`, `xmp.expand_dims(x, axis=0)`, and `ymp.argmax(model.predict(x), axis=1)`.
- Cell 25: The same list of disease names is repeated, and the predicted class is shown as `"Tomato__septoria_leaf_spot"`.
- Cell 26: `y` is assigned the result of the previous cell, showing `array([8], dtype=int64)`.
- Cell 27: `index[y[0]]` is used to retrieve the predicted class, showing `"Tomato__septoria_leaf_spot"`.

Outcomes achieved with local deployment using Flask Package:



## 7 ADVANTAGES & DISADVANTAGES

A web Application is built which enables:

- Farmers to interact with the portal build
- Interacts with the user interface to upload images of diseased leaf
- Our model-built analyses the Disease and suggests the farmer which fertilizers are to be used.

The disadvantage is that farmers must be familiar with the usage of this UI and must be educated enough otherwise they need some area expert to make them aware, how using this model could help them in choosing the best fertilizers to avoid crop leaf diseases.

## 8 APPLICATIONS

This solution can be applied in plant nurseries as well along with farms to yield better crop

production results and to make the farmers be aware timely to get rid of crop and plant leaf diseases.

## **9 CONCLUSION**

In this project we proposed a deep learning model for leaf disease prediction and to recommend appropriate fertilizers according to predicted disease to farmers. For this an image dataset consisting of fruit and vegetable images has been used for training and testing the model and then for final deployment using flask application. The fruit leaf disease detection accuracy achieved is 93% and for vegetable dataset accuracy achieved is 81%.

## **10 FUTURE SCOPE**

In future we can make this model using transfer learning in order to avoid image pre-processing steps and to avoid data labelling. Where transfer learning is an approach that makes use of pre-trained models like MobileNet to work on images and avoid the need of experts for data pre-processing. It will save a lot of time for model training when volumes of data will be required to work upon in today's data intensive applications. In addition to that dataset can be enhanced by adding more crop images from different seasonal crops and more generalized solution can be obtained in that way.

## **11 BIBILOGRAPHY**

[https://smartinternz.com/Student/guided\\_project\\_info/86118#](https://smartinternz.com/Student/guided_project_info/86118#)

## **APPENDIX**

A. **Source Code:** has been attached on github at: <https://github.com/smartinternz02/Sl-GuidedProject-86118-1657773138>