CROP DISEASE DETECTION

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

Yagnesh S RA2111026010354
Sashi Vardhan RA2111026010343
Sakthi Vel Ram RA2111026010346
Aroc Boris RA2111026010372

Under the guidance of

Dr. Karpagam M

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"CROP DISEASE DETECTION"** is the bona fide work of **Yagnesh (RA2111026010354), Sashi Vardhan (RA2111026010343), Sakthi vel Ram (RA2111026010346), Aroc Boris (RA2111026010372)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Dr Karpagam M
Faculty
CINTEL
SRM Institute of Science and Technology,
Kattankulathur

SIGNATURE
Dr Annie Uthra
Head of Department
CINTEL
SRM Institute of Science and Technology,
Kattankulathur

TABLE OF CONTENTS 3

ABSTRACT

Crop diseases pose significant threats to agricultural productivity and food security worldwide. Early and accurate detection of these diseases is crucial for timely intervention and effective management. In recent years, advancements in technology, especially in the fields of image processing, machine learning, and deep learning, have revolutionized the way crop diseases are detected and diagnosed.

This review paper provides an overview of the latest techniques and technologies used in crop disease detection. It covers traditional methods such as visual inspection and laboratory testing, as well as modern non-invasive techniques like hyperspectral imaging, drone-based monitoring, and smartphone applications. Moreover, the paper delves into the application of artificial intelligence (AI) algorithms, including convolutional neural networks (CNNs) and support vector machines (SVMs), in automating the detection and classification of crop diseases based on visual symptoms.

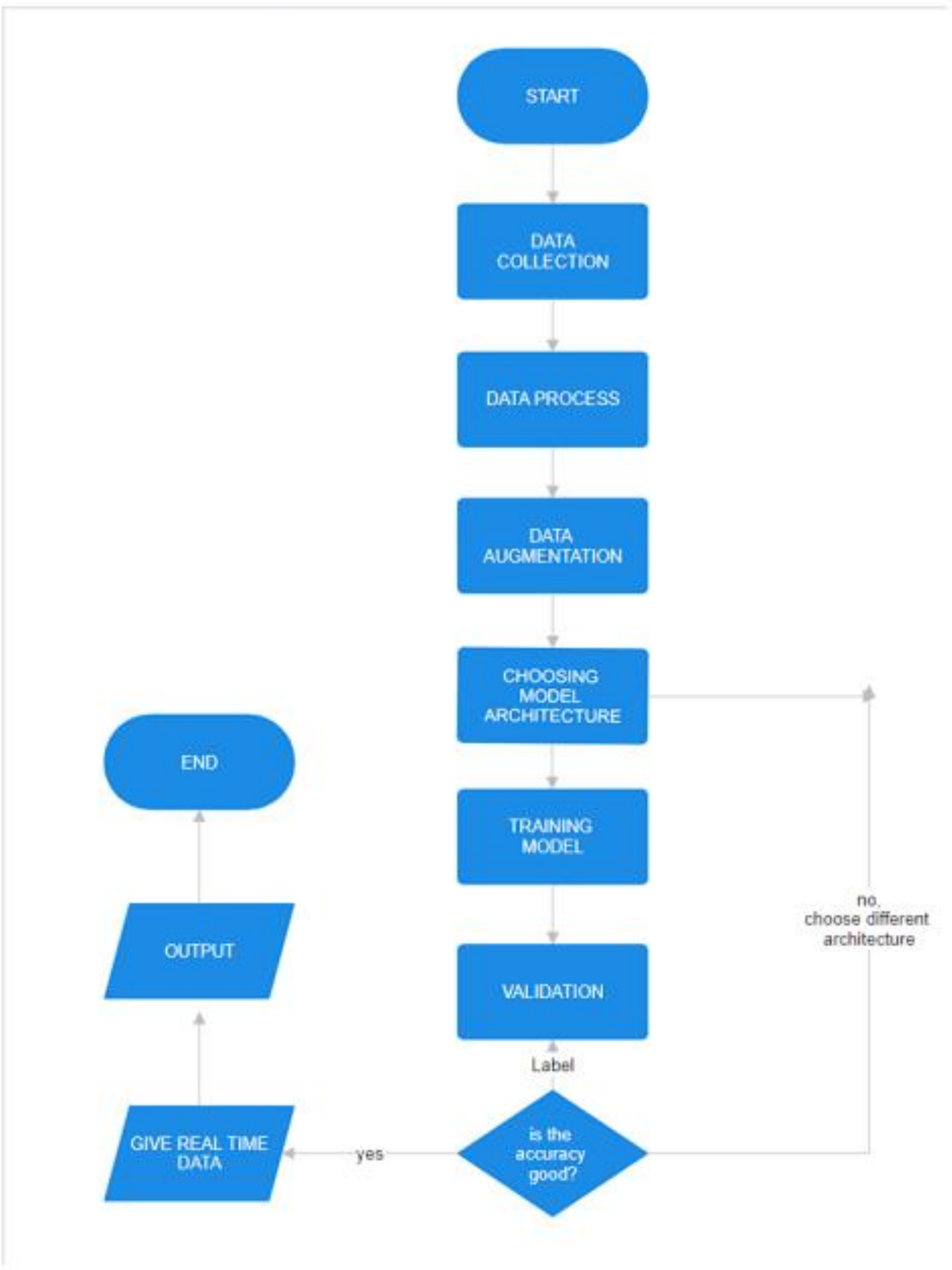
Furthermore, the review discusses the challenges and limitations associated with current methods, such as data variability, scalability, and model generalization. It also explores future research directions and potential solutions to improve the accuracy, efficiency, and accessibility of crop disease detection systems. Overall, this paper aims to provide insights into the state-of-the-art techniques and inspire further innovation in this critical domain of agricultural science and technology.

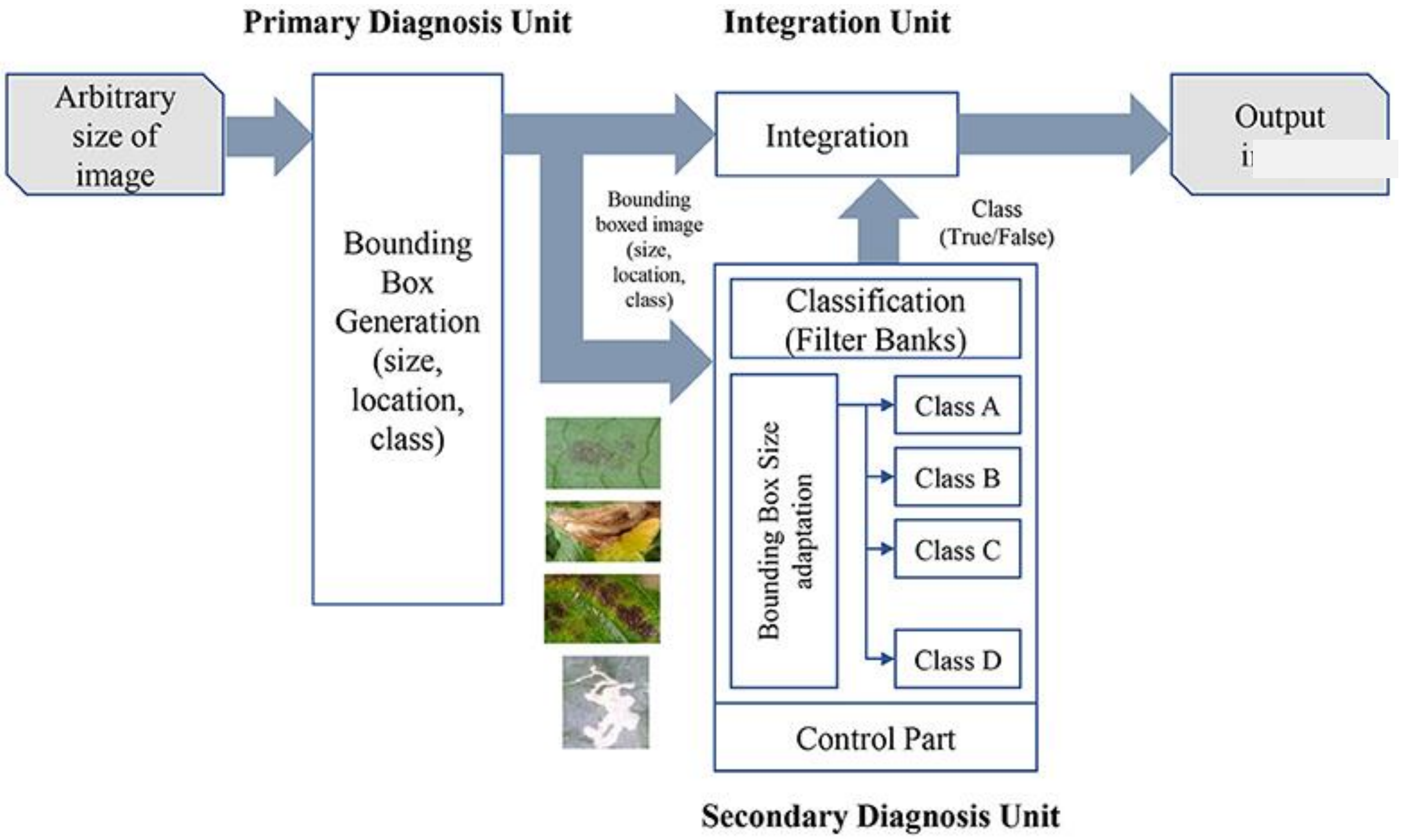
LITERATURE SURVEY

S.NO	Year	Paper title	Journal name	Algorithm/ technique	Dataset	Inference
1	2021	Crop leaf disease detection and classification using machine learning and deep learning algorithms by visual symptoms	International Journal of Electrical and Computer Engineering (IJECE)	SVM and modified CNN are used while multi-channels CNN gives more promising accuracy.	Plant Village	Dataset is the main drawback, using deep learning tiles enormous data and to fit in small systems are still lacking. Better than traditional ML models with increased .5% accuracy.
2	**2022**	**Effective multi-crop disease detection using pruned complete concatenated deep learning model.**	**ScienceDirect**	**CCDL Concatenated Convolutional DL id used and for feature map**	**Plant Village**	**The dataset used for training the deep learning model was relatively small, which could limit the performance of the model on new data. The model requires a large amount of labeled data to train.**
3	2018	Automated leaf disease detection in different crop species through image features analysis and One Class Classifiers	ScienceDirect	Image Segmentation of Multiple Medical Objects Using Deep Learning" is a convolutional neural network (CNN)	BraTS 2017 dataset	some ethical considerations that need to be taken into account when using the BraTS dataset. The model was not evaluated on images with different levels of noise, so it is not clear how well it would perform on noisy images.

SYSTEM ARCHITECTURE AND DESIGN

FLOW CHART:-




Primary Diagnosis Unit
Integration Unit
Arbitrary size of image
Bounding Box Generation (size, location, class)
Bounding boxed image (size, location, class)
Integration
Output i
Class (True/False)
Classification (Filter Banks)
Bounding Box Size adaptation
Class A
Class B
Class C
Class D
Control Part
Secondary Diagnosis Unit

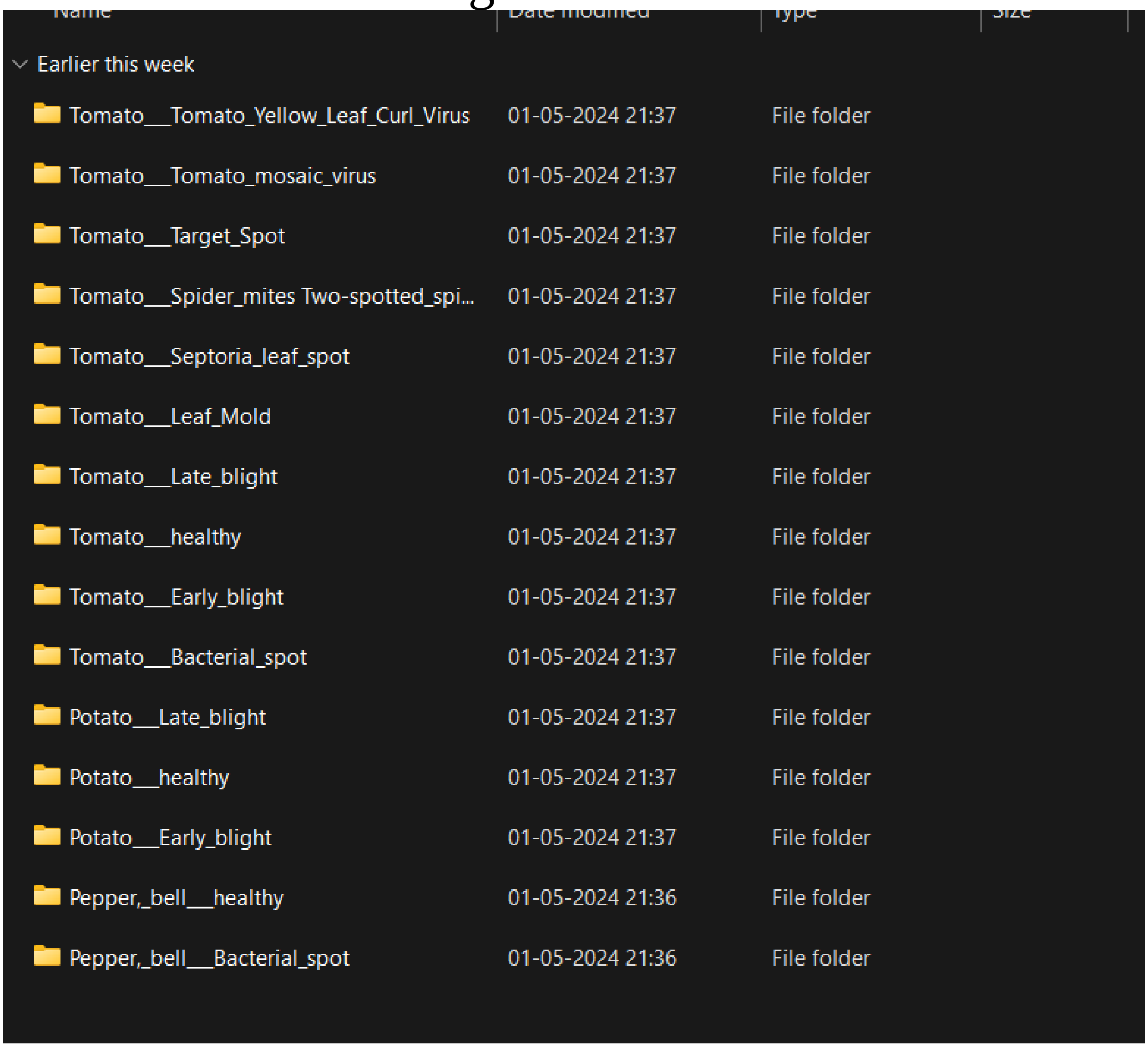
DATASET DESCRIPTION

The data we used for this project can easily be found on Kaggle, 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. But only 15 class was taken, and on those 15 classes 3 belong to healthy. Only potato, tomato and bell pepper leaves images were used to train the model to reduce the complexity and training time of the model.

Tomato___Tomato_Yellow_Leaf_Curl_Virus

Tomato healthy

The images were of different sizes so image resizing need to be done for all images in the taken dataset.



Name	Date modified	Type	Size
Earlier this week			
Tomato__Tomato_Yellow_Leaf_Curl_Virus	01-05-2024 21:37	File folder	
Tomato__Tomato_mosaic_virus	01-05-2024 21:37	File folder	
Tomato__Target_Spot	01-05-2024 21:37	File folder	
Tomato__Spider_mites Two-spotted_spi...	01-05-2024 21:37	File folder	
Tomato__Septoria_leaf_spot	01-05-2024 21:37	File folder	
Tomato__Leaf_Mold	01-05-2024 21:37	File folder	
Tomato__Late_blight	01-05-2024 21:37	File folder	
Tomato__healthy	01-05-2024 21:37	File folder	
Tomato__Early_blight	01-05-2024 21:37	File folder	
Tomato__Bacterial_spot	01-05-2024 21:37	File folder	
Potato__Late_blight	01-05-2024 21:37	File folder	
Potato__healthy	01-05-2024 21:37	File folder	
Potato__Early_blight	01-05-2024 21:37	File folder	
Pepper,_bell__healthy	01-05-2024 21:36	File folder	
Pepper,_bell__Bacterial_spot	01-05-2024 21:36	File folder	

Above is the dataset taken for training purpose, the validation data is saved separately in another directory. Most of the images were in the size of 256 x 256, later it was bought to 64 x 64, with the help of CV2 library. There was unnecessary images found in the dataset, though a cleaning function is created and iterated through the dataset

METHODOLOGY

1. **Data Collection and Preprocessing:**
    a. Collect images containing leafs with and without disease. We used Kaggle for data collection.
    b. Preprocess the images to standardize dimensions, normalize intensity levels, and remove noise.
    c. Split the dataset into training, validation, and testing sets.
2. **Data Augmentation:**
    a. Augment the training dataset by applying transformations such as rotation, scaling, flipping, and shifting.
    b. This step increases the diversity of the training data and helps prevent overfitting.

3. **Model Training:**
    a. Design a deep learning architecture, such as a Convolutional Neural Network (CNN), for crop disease detection.
    b. Train the model using the augmented training dataset.
    c. Utilize techniques like transfer learning if applicable, using pretrained models like VGG, ResNet, or Inception.

4. **Model Evaluation:**
    a. Evaluate the trained model using the validation dataset to monitor performance during training and adjust hyperparameters as needed.

    b. Fine-tune the model based on validation results to improve generalization and avoid overfitting.

5. **Testing and Performance Evaluation:**
    a. Evaluate the final trained model using the separate testing dataset to assess its real-world performance.
    b. Calculate metrics such as accuracy, sensitivity, specificity, and AUCROC to measure the model's effectiveness in disease detection.

6. **Deployment and Integration:**
    a. Once satisfied with the model's performance, deploy it in a clinical or research environment for real-world use.
    b. Integrate the model into existing croping systems or develop a user-friendly interface for farmers to use for disease detection.

7. Each of these steps represents a crucial part of the crop disease detection pipeline using deep learning. The flowchart helps visualize the sequential nature of the process and the interconnections between different stages.

CODING AND TESTING

1) Libraries and Data loading

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import cv2
import os
from tqdm import tqdm
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
from keras.models import Model,Sequential, Input, load_model
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D,
BatchNormalization, AveragePooling2D, GlobalAveragePooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
from keras.applications import DenseNet121

disease_types =
['Pepper__bell___Bacterial_spot','Pepper__bell___healthy','Potato___Early_blight','
Potato___Late_blight','Potato___healthy','Tomato_Bacterial_spot','Tomato_Early_bl
ight','Tomato_Late_blight','Tomato_Leaf_Mold','Tomato_Septoria_leaf_spot','Tomat
o_Spider_mites_Two_spotted_spider_mite','Tomato__Target_Spot','Tomato__Toma
to_YellowLeaf__Curl_Virus','Tomato__Tomato_mosaic_virus','Tomato_healthy']
data_dir = '../input/plantdisease/PlantVillage/'
train_dir = os.path.join(data_dir)

train_data = []
for defects_id, sp in enumerate(disease_types):
    for file in os.listdir(os.path.join(train_dir, sp)):
        train_data.append(['{}/{}'.format(sp, file), defects_id, sp])
```

```
train = pd.DataFrame(train_data, columns=['File', 'DiseaseID','Disease Type'])
train.tail()
```

2) Image Resizing and Imagedatagenerator

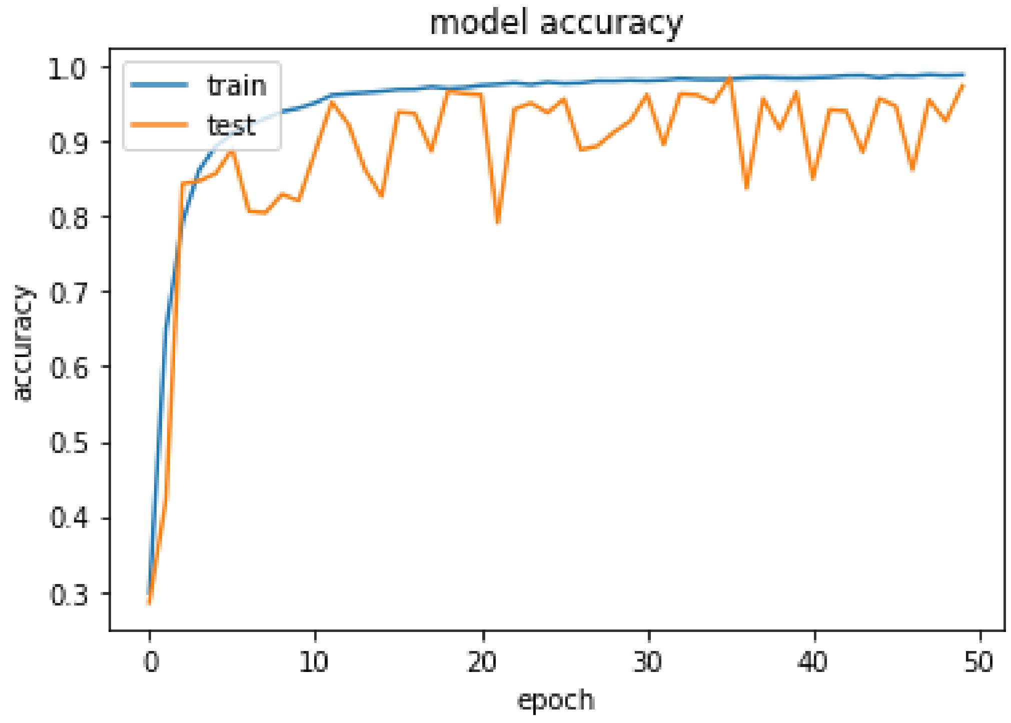
```
IMAGE_SIZE = 64

def read_image(filepath):
    return cv2.imread(os.path.join(data_dir, filepath)) # Loading a color image is the
default flag
# Resize image to target size
def resize_image(image, image_size):
    return cv2.resize(image.copy(), image_size, interpolation=cv2.INTER_AREA)
X_train = np.zeros((train.shape[0], IMAGE_SIZE, IMAGE_SIZE, 3))
for i, file in tqdm(enumerate(train['File'].values)):
    image = read_image(file)
    if image is not None:
        X_train[i] = resize_image(image, (IMAGE_SIZE, IMAGE_SIZE))
# Normalize the data
X_Train = X_train / 255.
print('Train Shape: {}'.format(X_Train.shape))
BATCH_SIZE = 64

# Split the train and validation sets
X_train, X_val, Y_train, Y_val = train_test_split(X_Train, Y_train, test_size=0.2,
random_state=SEED)
datagen = ImageDataGenerator(rotation_range=360, # Degree range for random
rotations
                        width_shift_range=0.2, # Range for random horizontal shifts
                        height_shift_range=0.2, # Range for random vertical shifts
                        zoom_range=0.2, # Range for random zoom
                        horizontal_flip=True, # Randomly flip inputs horizontally
                        vertical_flip=True) # Randomly flip inputs vertically

datagen.fit(X_train)
```

3) Modelling

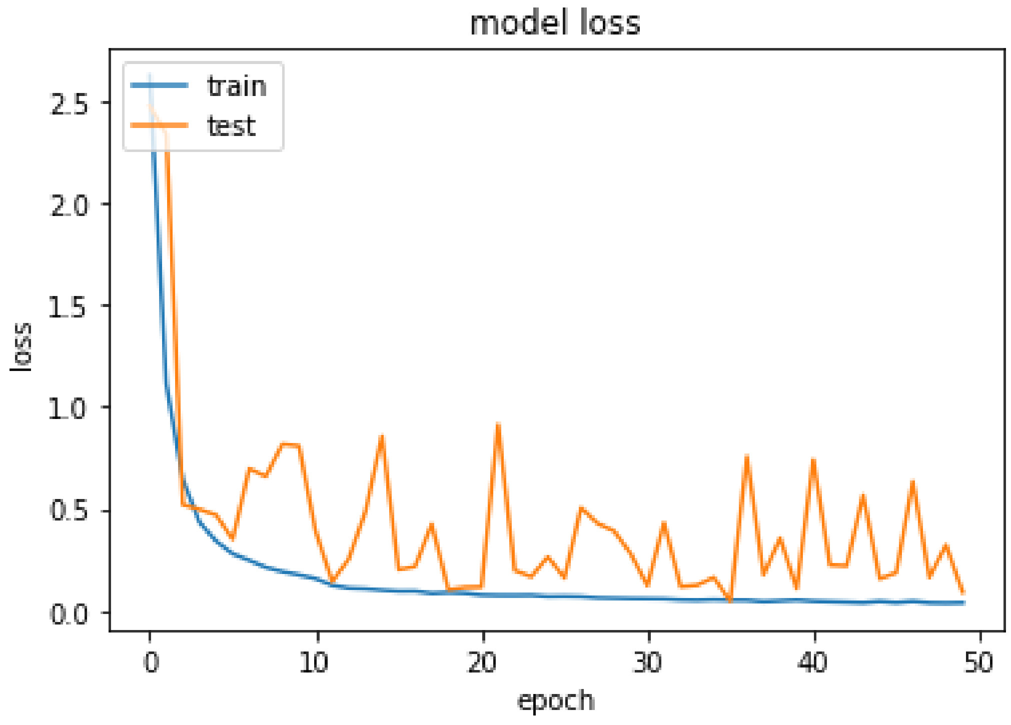
```
def build_densenet():
    densenet = DenseNet121(weights='imagenet', include_top=False)

    input = Input(shape=(SIZE, SIZE, N_ch))
    x = Conv2D(3, (3, 3), padding='same')(input)

    x = densenet(x)

    x = GlobalAveragePooling2D()(x)
    x = BatchNormalization()(x)
    x = Dropout(0.5)(x)
    x = Dense(256, activation='relu')(x)
    x = BatchNormalization()(x)
    x = Dropout(0.5)(x)

    # multi output
    output = Dense(15,activation = 'softmax', name='root')(x)



    # model
    model = Model(input,output)

    optimizer = Adam(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=0.1, decay=0.0)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer,
metrics=['accuracy'])
    model.summary()

    return model
model = build_densenet()
annealer = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5, patience=5,
verbose=1, min_lr=1e-3)
checkpoint = ModelCheckpoint('model.h5', verbose=1, save_best_only=True)
hist = model.fit_generator(datagen.flow(X_train, Y_train,
batch_size=BATCH_SIZE),
               steps_per_epoch=X_train.shape[0] // BATCH_SIZE,
               epochs=EPOCHS,
               verbose=2,
               callbacks=[annealer, checkpoint],
               validation_data=(X_val, Y_val))
```

4) Evaluation

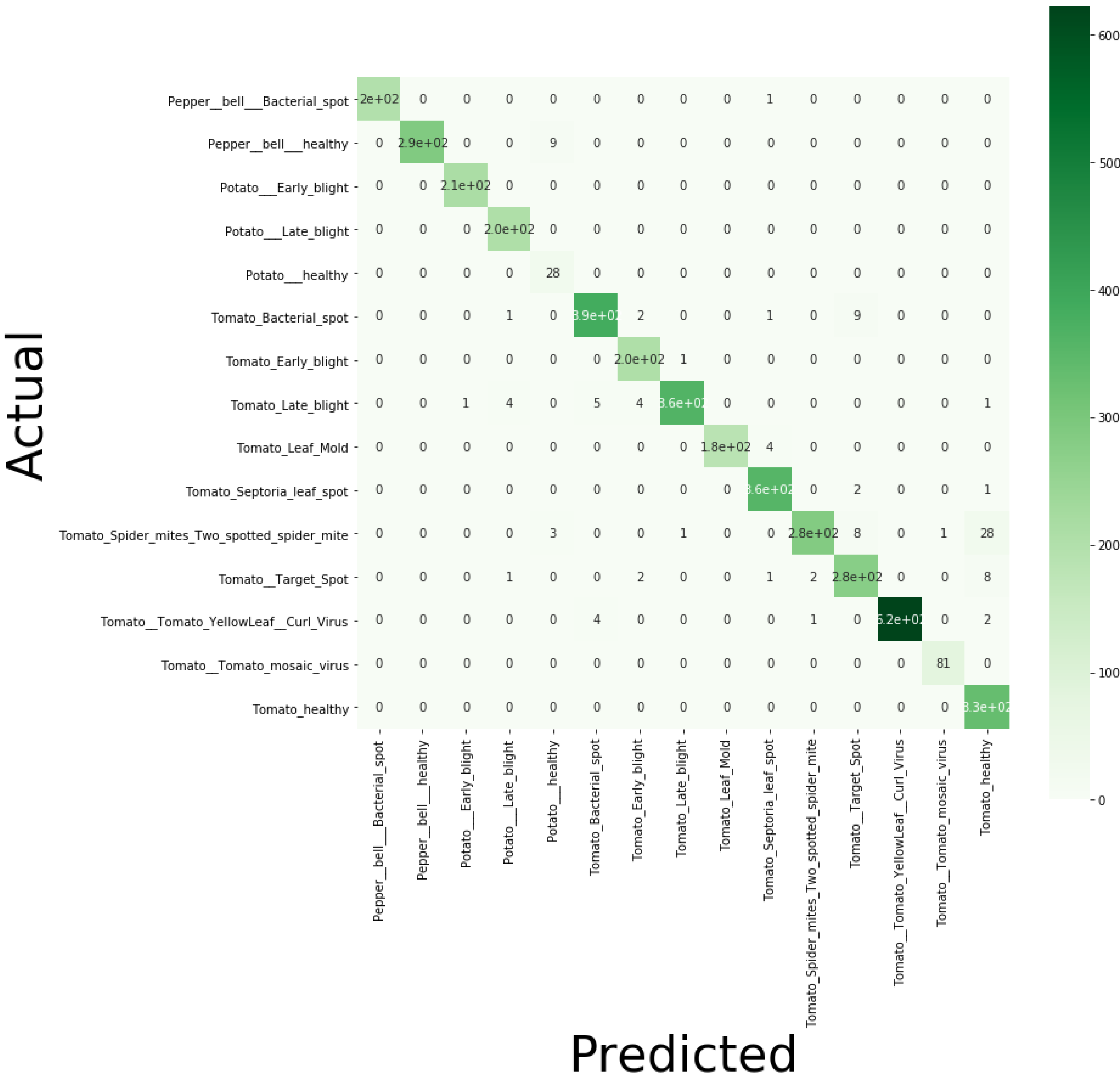
```
final_loss, final_accuracy = model.evaluate(X_val, Y_val)
print('Final Loss: {}, Final Accuracy: {}'.format(final_loss, final_accuracy))
Y_pred = model.predict(X_val)

Y_pred = np.argmax(Y_pred, axis=1)
Y_true = np.argmax(Y_val, axis=1)

cm = confusion_matrix(Y_true, Y_pred)
plt.figure(figsize=(12, 12))
ax = sns.heatmap(cm, cmap=plt.cm.Greens, annot=True, square=True,
xticklabels=disease_types, yticklabels=disease_types)
ax.set_ylabel('Actual', fontsize=40)
ax.set_xlabel('Predicted', fontsize=40)

# accuracy plot
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# loss plot
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



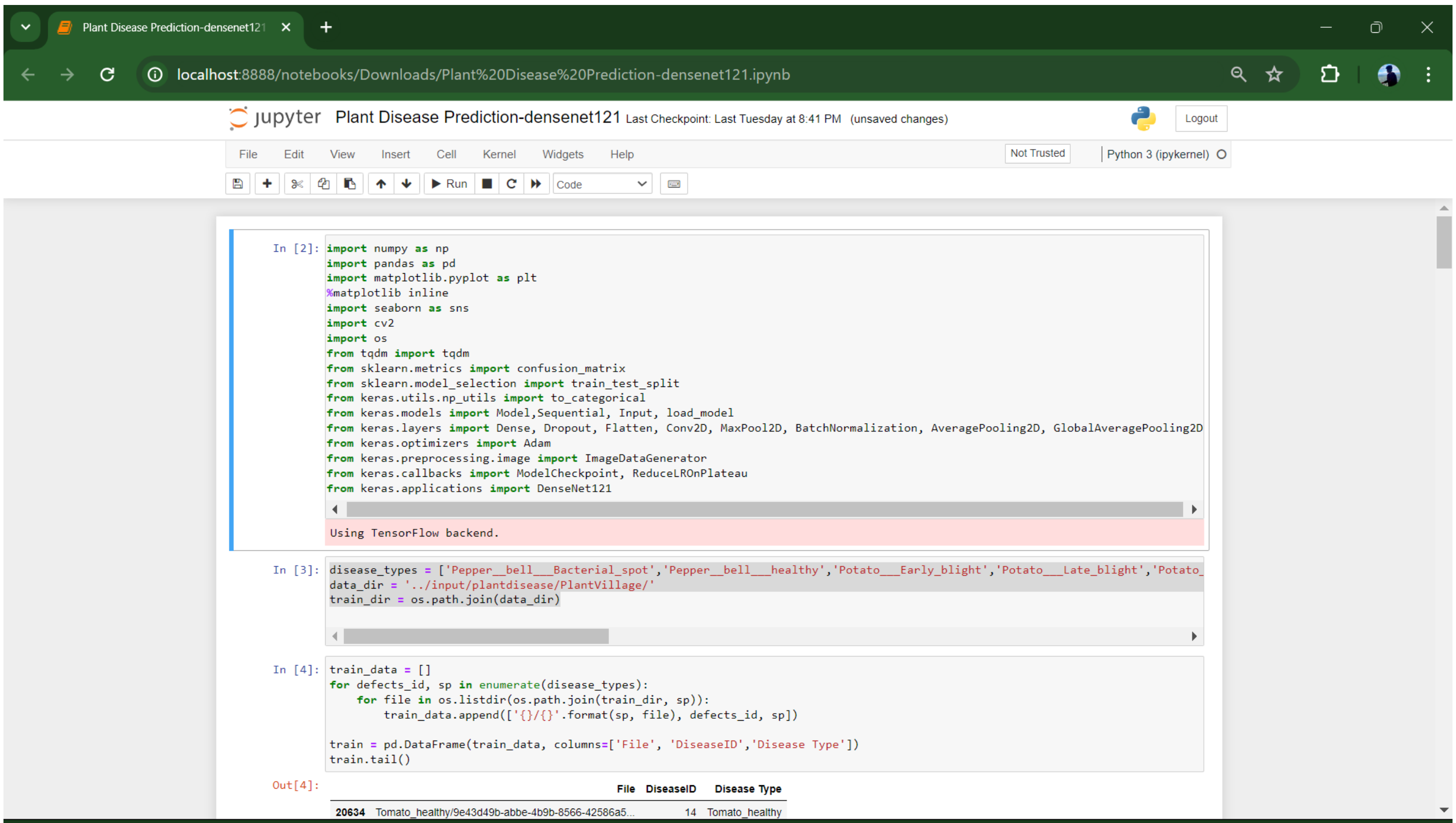
model accuracy
accuracy
epoch
train
test



model loss
loss
epoch
train
test

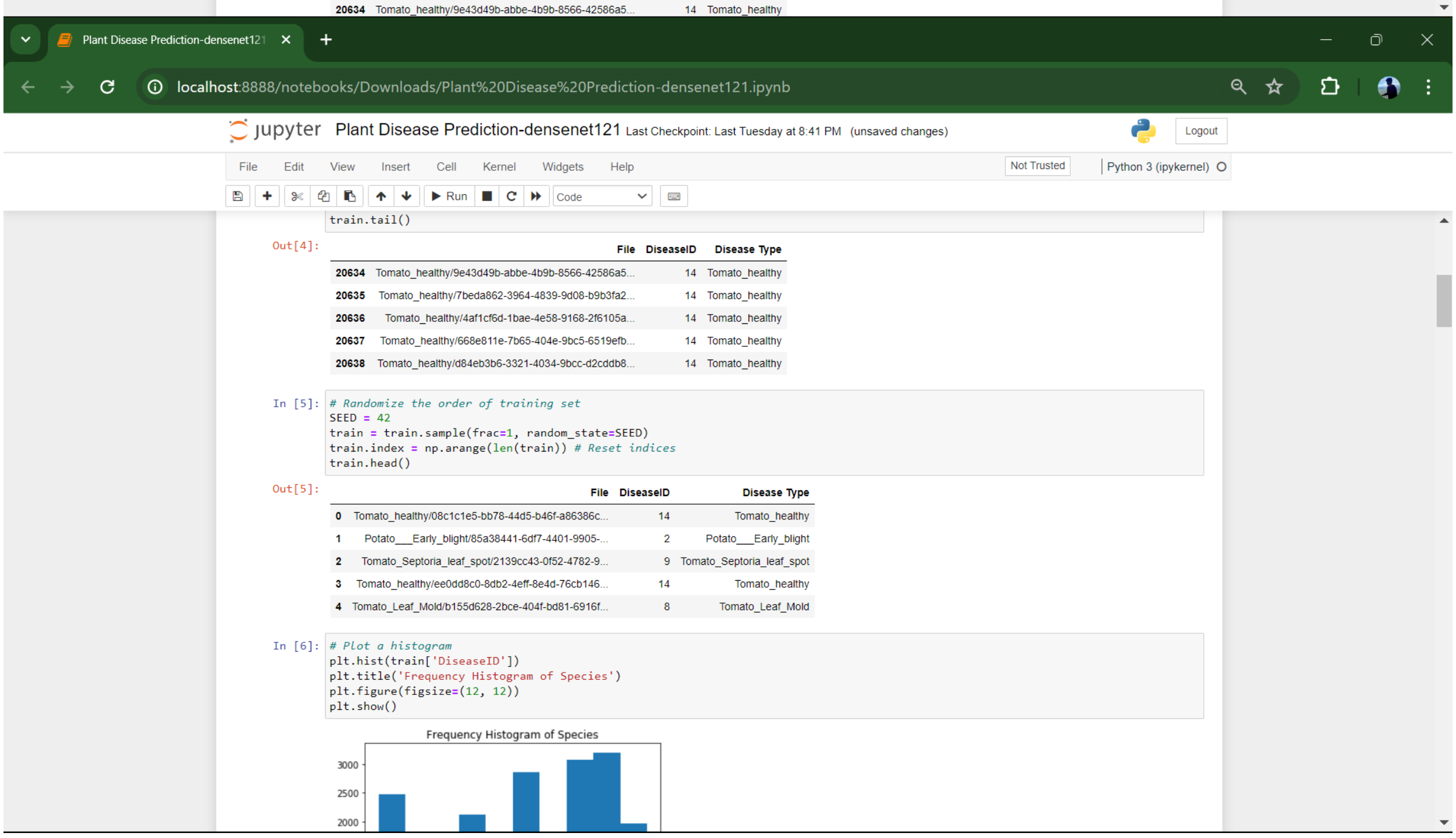

Actual
Predicted
Pepper__bell___Bacterial_spot
Pepper__bell___healthy
Potato___Early_blight
Potato___Late_blight
Potato___healthy
Tomato_Bacterial_spot
Tomato_Early_blight
Tomato_Late_blight
Tomato_Leaf_Mold
Tomato_Septoria_leaf_spot
Tomato_Spider_mites_Two_spotted_spider_mite
Tomato__Target_Spot
Tomato__Tomato_YellowLeaf__Curl_Virus
Tomato__Tomato_mosaic_virus
Tomato_healthy
2e+02 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 2.9e+02 0 0 9 0 0 0 0 0 0 0 0 0 0
0 0 2.1e+02 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 2.0e+02 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 28 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 3.9e+02 2 0 0 1 0 9 0 0 0
0 0 0 0 0 0 2.0e+02 1 0 0 0 0 0 0 0
0 0 1 4 0 5 4 3.6e+02 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 1.8e+02 4 0 0 0 0 0
0 0 0 0 0 0 0 0 0 3.6e+02 0 2 0 0 1
0 0 0 0 3 0 0 1 0 0 2.8e+02 8 0 1 28
0 0 0 1 0 0 2 0 0 1 2 2.8e+02 0 0 8
0 0 0 0 0 4 0 0 0 0 1 0 6.2e+02 0 2
0 0 0 0 0 0 0 0 0 0 0 0 0 81 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 3.3e+02
600
500
400
300
200
100
0

SCREENSHOTS


Plant Disease Prediction-densenet121
localhost:8888/notebooks/Downloads/Plant%20Disease%20Prediction-densenet121.ipynb
jupyter Plant Disease Prediction-densenet121 Last Checkpoint: Last Tuesday at 8:41 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        %matplotlib inline
        import seaborn as sns
        import cv2
        import os
        from tqdm import tqdm
        from sklearn.metrics import confusion_matrix
        from sklearn.model_selection import train_test_split
        from keras.utils.np_utils import to_categorical
        from keras.models import Model,Sequential, Input, load_model
        from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, BatchNormalization, AveragePooling2D, GlobalAveragePooling2D
        from keras.optimizers import Adam
        from keras.preprocessing.image import ImageDataGenerator
        from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
        from keras.applications import DenseNet121

        Using TensorFlow backend.

In [3]: disease_types = ['Pepper__bell___Bacterial_spot','Pepper__bell___healthy','Potato___Early_blight','Potato___Late_blight','Potato_
        data_dir = '../input/plantdisease/PlantVillage/'
        train_dir = os.path.join(data_dir)

In [4]: train_data = []
        for defects_id, sp in enumerate(disease_types):
            for file in os.listdir(os.path.join(train_dir, sp)):
                train_data.append(['{}/{}'.format(sp, file), defects_id, sp])

        train = pd.DataFrame(train_data, columns=['File', 'DiseaseID','Disease Type'])
        train.tail()

Out[4]:
```

	File	DiseaseID	Disease Type
20634	Tomato_healthy/9e43d49b-abbe-4b9b-8566-42586a5...	14	Tomato_healthy


Plant Disease Prediction-densenet121
localhost:8888/notebooks/Downloads/Plant%20Disease%20Prediction-densenet121.ipynb
jupyter Plant Disease Prediction-densenet121 Last Checkpoint: Last Tuesday at 8:41 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Frequency Histogram of Species
3000
2500
2000

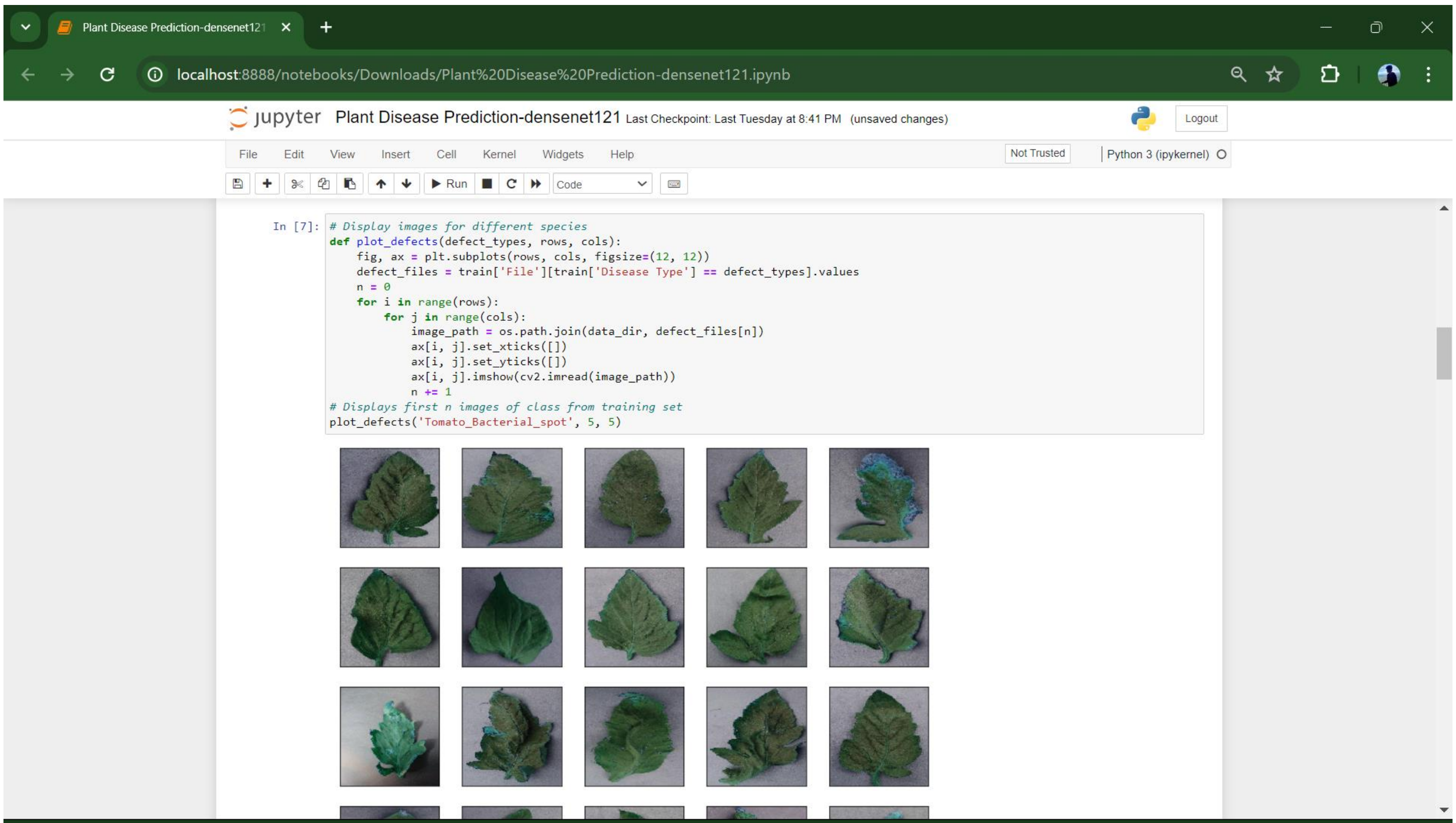
```
        train.tail()

Out[4]:
```

	File	DiseaseID	Disease Type
20634	Tomato_healthy/9e43d49b-abbe-4b9b-8566-42586a5...	14	Tomato_healthy
20635	Tomato_healthy/7beda862-3964-4839-9d08-b9b3fa2...	14	Tomato_healthy
20636	Tomato_healthy/4af1cf6d-1bae-4e58-9168-2f6105a...	14	Tomato_healthy
20637	Tomato_healthy/668e811e-7b65-404e-9bc5-6519efb...	14	Tomato_healthy
20638	Tomato_healthy/d84eb3b6-3321-4034-9bcc-d2cddb8...	14	Tomato_healthy

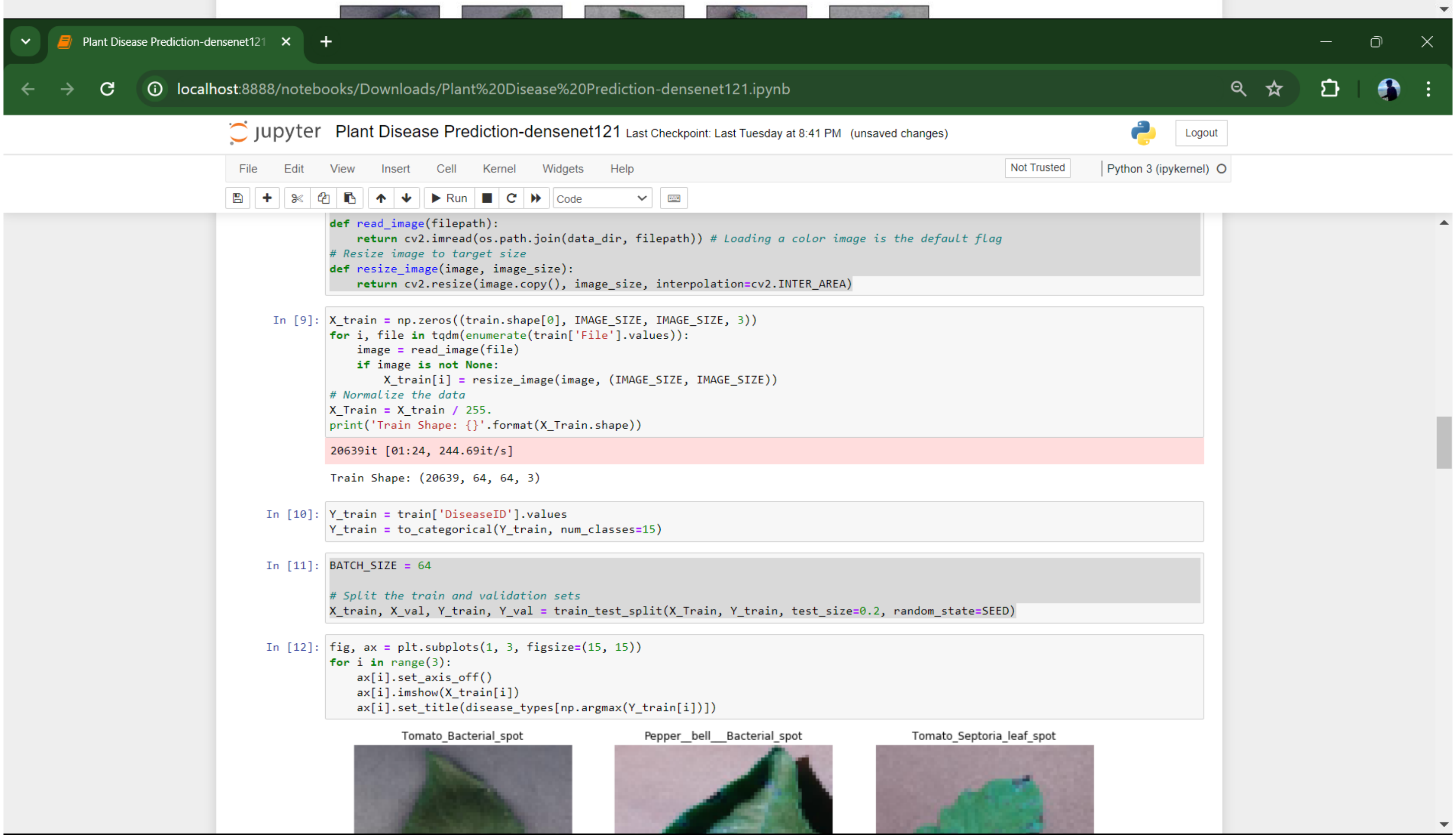
```
In [5]: # Randomize the order of training set
        SEED = 42
        train = train.sample(frac=1, random_state=SEED)
        train.index = np.arange(len(train)) # Reset indices
        train.head()

Out[5]:
```

	File	DiseaseID	Disease Type
0	Tomato_healthy/08c1c1e5-bb78-44d5-b46f-a86386c...	14	Tomato_healthy
1	Potato___Early_blight/85a38441-6df7-4401-9905-...	2	Potato___Early_blight
2	Tomato_Septoria_leaf_spot/2139cc43-0f52-4782-9...	9	Tomato_Septoria_leaf_spot
3	Tomato_healthy/ee0dd8c0-8db2-4eff-8e4d-76cb146...	14	Tomato_healthy
4	Tomato_Leaf_Mold/b155d628-2bce-404f-bd81-6916f...	8	Tomato_Leaf_Mold

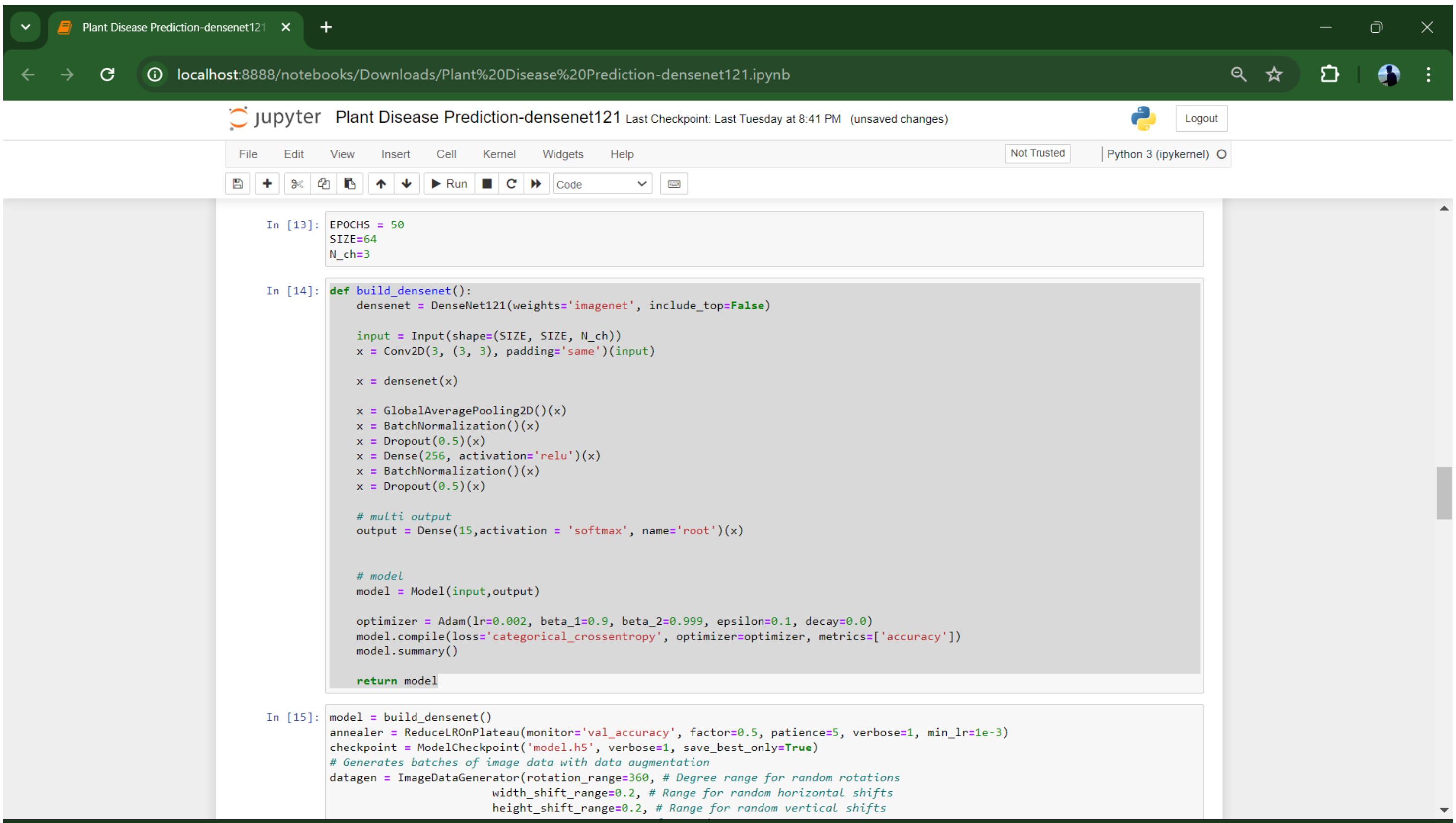
```
In [6]: # Plot a histogram
        plt.hist(train['DiseaseID'])
        plt.title('Frequency Histogram of Species')
        plt.figure(figsize=(12, 12))
        plt.show()
```


Plant Disease Prediction-densenet121
localhost:8888/notebooks/Downloads/Plant%20Disease%20Prediction-densenet121.ipynb
jupyter Plant Disease Prediction-densenet121 Last Checkpoint: Last Tuesday at 8:41 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [7]: # Display images for different species
        def plot_defects(defect_types, rows, cols):
            fig, ax = plt.subplots(rows, cols, figsize=(12, 12))
            defect_files = train['File'][train['Disease Type'] == defect_types].values
            n = 0
            for i in range(rows):
                for j in range(cols):
                    image_path = os.path.join(data_dir, defect_files[n])
                    ax[i, j].set_xticks([])
                    ax[i, j].set_yticks([])
                    ax[i, j].imshow(cv2.imread(image_path))
                    n += 1
        # Displays first n images of class from training set
        plot_defects('Tomato_Bacterial_spot', 5, 5)
```


Plant Disease Prediction-densenet121
localhost:8888/notebooks/Downloads/Plant%20Disease%20Prediction-densenet121.ipynb
jupyter Plant Disease Prediction-densenet121 Last Checkpoint: Last Tuesday at 8:41 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Tomato_Bacterial_spot
Pepper__bell___Bacterial_spot
Tomato_Septoria_leaf_spot

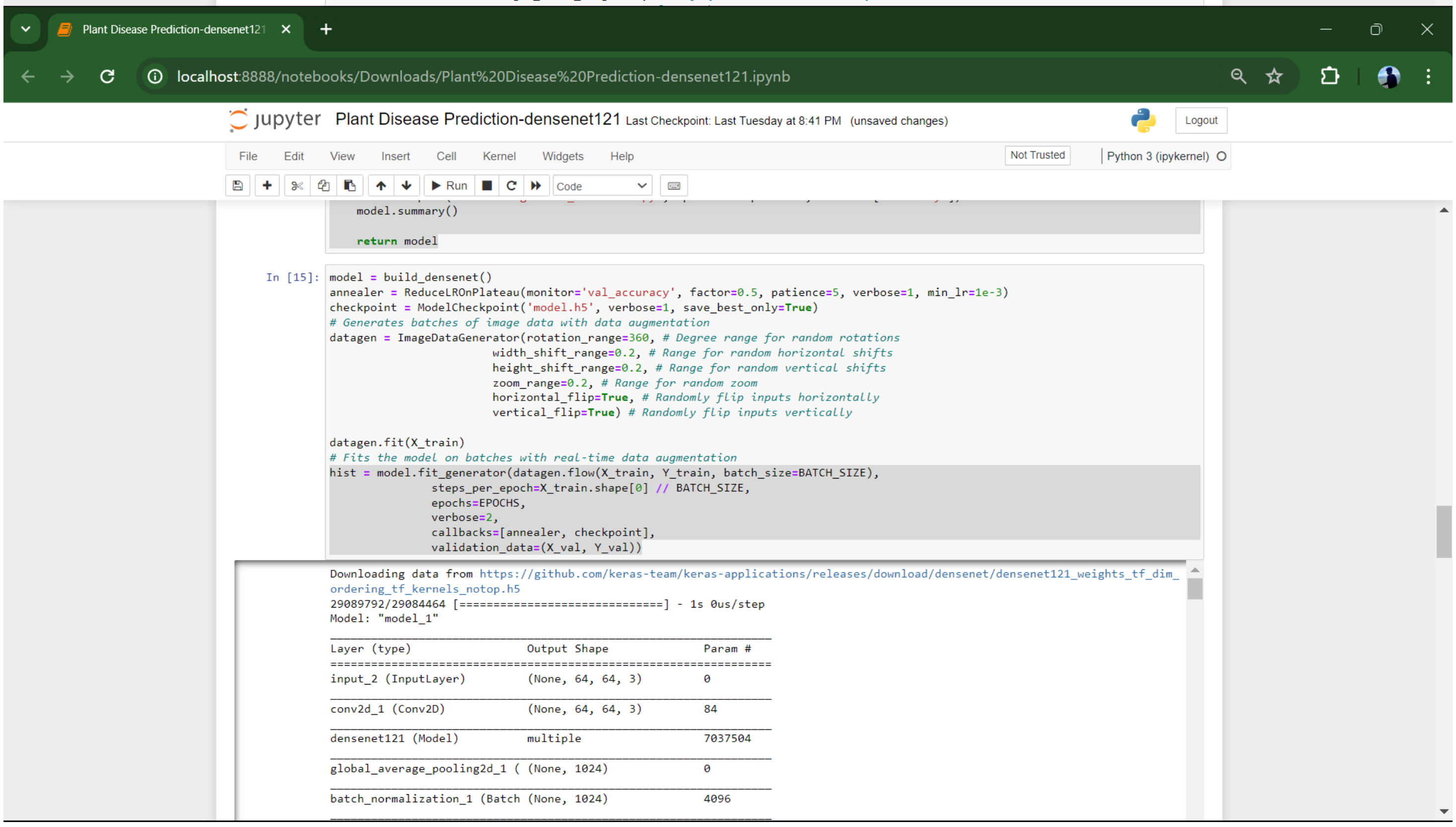
```
        def read_image(filepath):
            return cv2.imread(os.path.join(data_dir, filepath)) # Loading a color image is the default flag
        # Resize image to target size
        def resize_image(image, image_size):
            return cv2.resize(image.copy(), image_size, interpolation=cv2.INTER_AREA)

In [9]: X_train = np.zeros((train.shape[0], IMAGE_SIZE, IMAGE_SIZE, 3))
        for i, file in tqdm(enumerate(train['File'].values)):
            image = read_image(file)
            if image is not None:
                X_train[i] = resize_image(image, (IMAGE_SIZE, IMAGE_SIZE))
        # Normalize the data
        X_Train = X_train / 255.
        print('Train Shape: {}'.format(X_Train.shape))

        20639it [01:24, 244.69it/s]

        Train Shape: (20639, 64, 64, 3)

In [10]: Y_train = train['DiseaseID'].values
         Y_train = to_categorical(Y_train, num_classes=15)

In [11]: BATCH_SIZE = 64

         # Split the train and validation sets
         X_train, X_val, Y_train, Y_val = train_test_split(X_Train, Y_train, test_size=0.2, random_state=SEED)

In [12]: fig, ax = plt.subplots(1, 3, figsize=(15, 15))
         for i in range(3):
             ax[i].set_axis_off()
             ax[i].imshow(X_train[i])
             ax[i].set_title(disease_types[np.argmax(Y_train[i])])
```


Plant Disease Prediction-densenet121
localhost:8888/notebooks/Downloads/Plant%20Disease%20Prediction-densenet121.ipynb
jupyter Plant Disease Prediction-densenet121 Last Checkpoint: Last Tuesday at 8:41 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [13]: EPOCHS = 50
         SIZE=64
         N_ch=3
```

```
In [14]: def build_densenet():
             densenet = DenseNet121(weights='imagenet', include_top=False)

             input = Input(shape=(SIZE, SIZE, N_ch))
             x = Conv2D(3, (3, 3), padding='same')(input)

             x = densenet(x)

             x = GlobalAveragePooling2D()(x)
             x = BatchNormalization()(x)
             x = Dropout(0.5)(x)
             x = Dense(256, activation='relu')(x)
             x = BatchNormalization()(x)
             x = Dropout(0.5)(x)

             # multi output
             output = Dense(15,activation = 'softmax', name='root')(x)


             # model
             model = Model(input,output)

             optimizer = Adam(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=0.1, decay=0.0)
             model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
             model.summary()

             return model
```

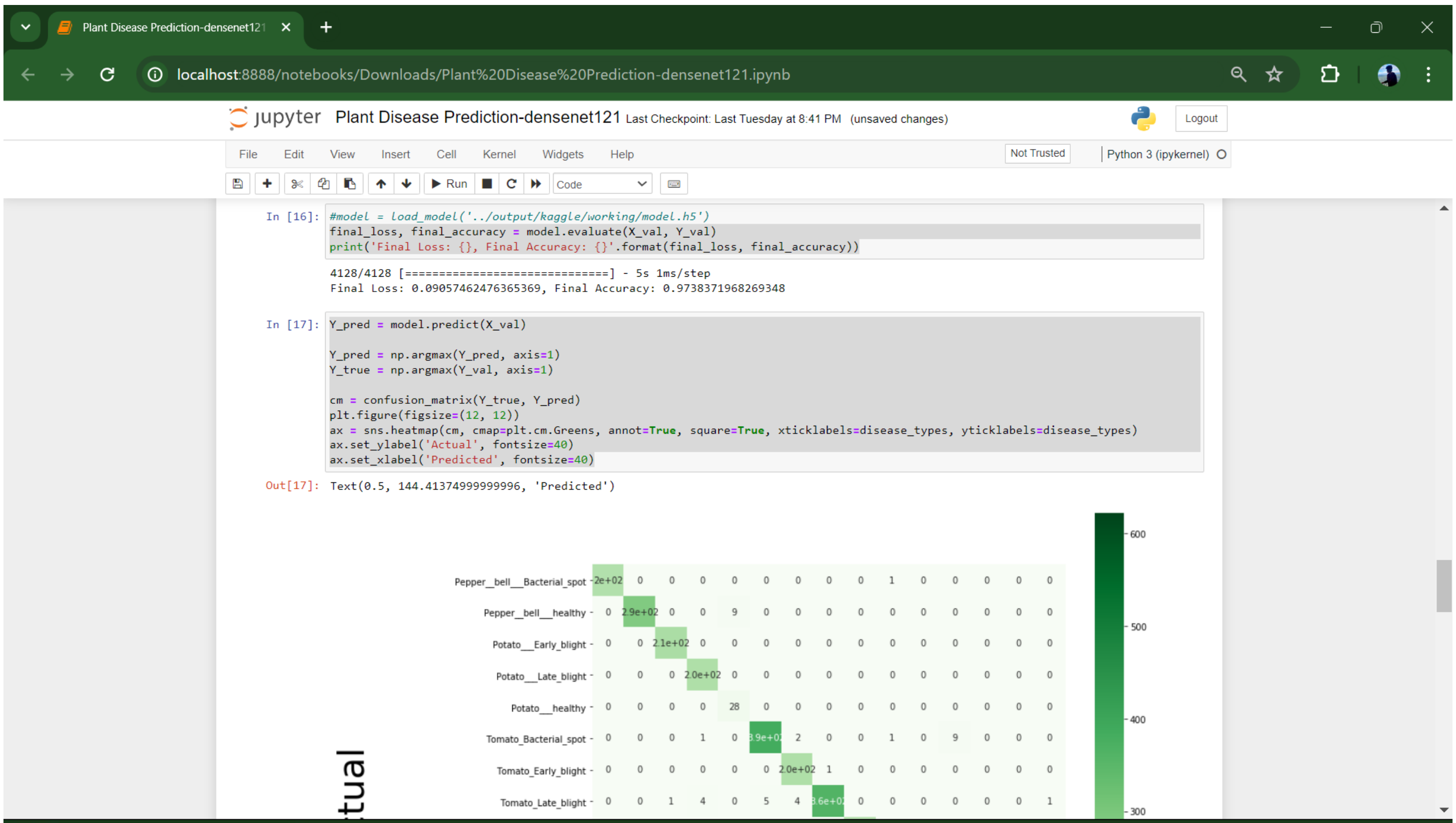
```
In [15]: model = build_densenet()
         annealer = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5, patience=5, verbose=1, min_lr=1e-3)
         checkpoint = ModelCheckpoint('model.h5', verbose=1, save_best_only=True)
         # Generates batches of image data with data augmentation
         datagen = ImageDataGenerator(rotation_range=360, # Degree range for random rotations
                                 width_shift_range=0.2, # Range for random horizontal shifts
                                 height_shift_range=0.2, # Range for random vertical shifts
```


Plant Disease Prediction-densenet121
localhost:8888/notebooks/Downloads/Plant%20Disease%20Prediction-densenet121.ipynb
jupyter Plant Disease Prediction-densenet121 Last Checkpoint: Last Tuesday at 8:41 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

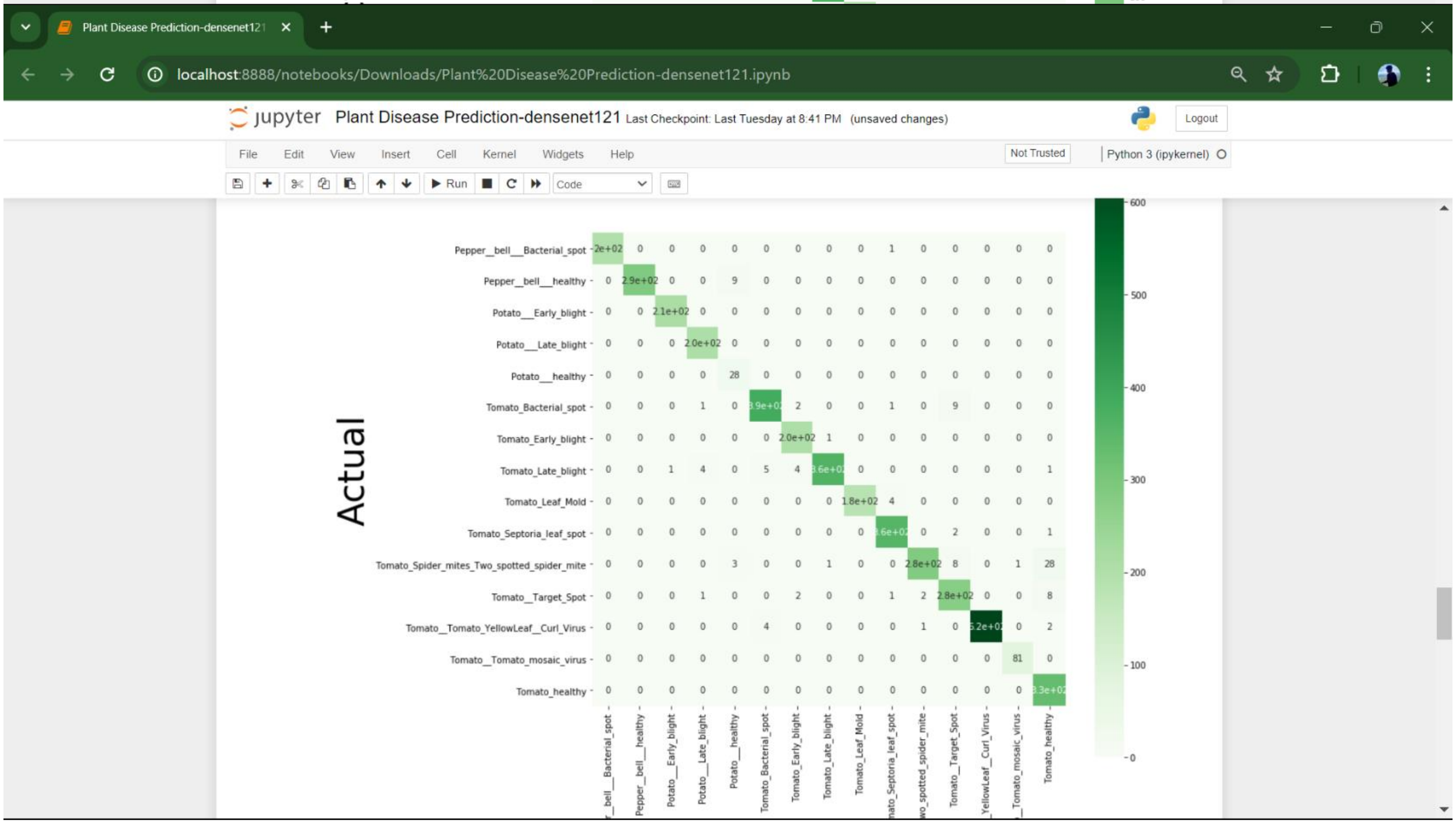
```
             model.summary()

             return model
```

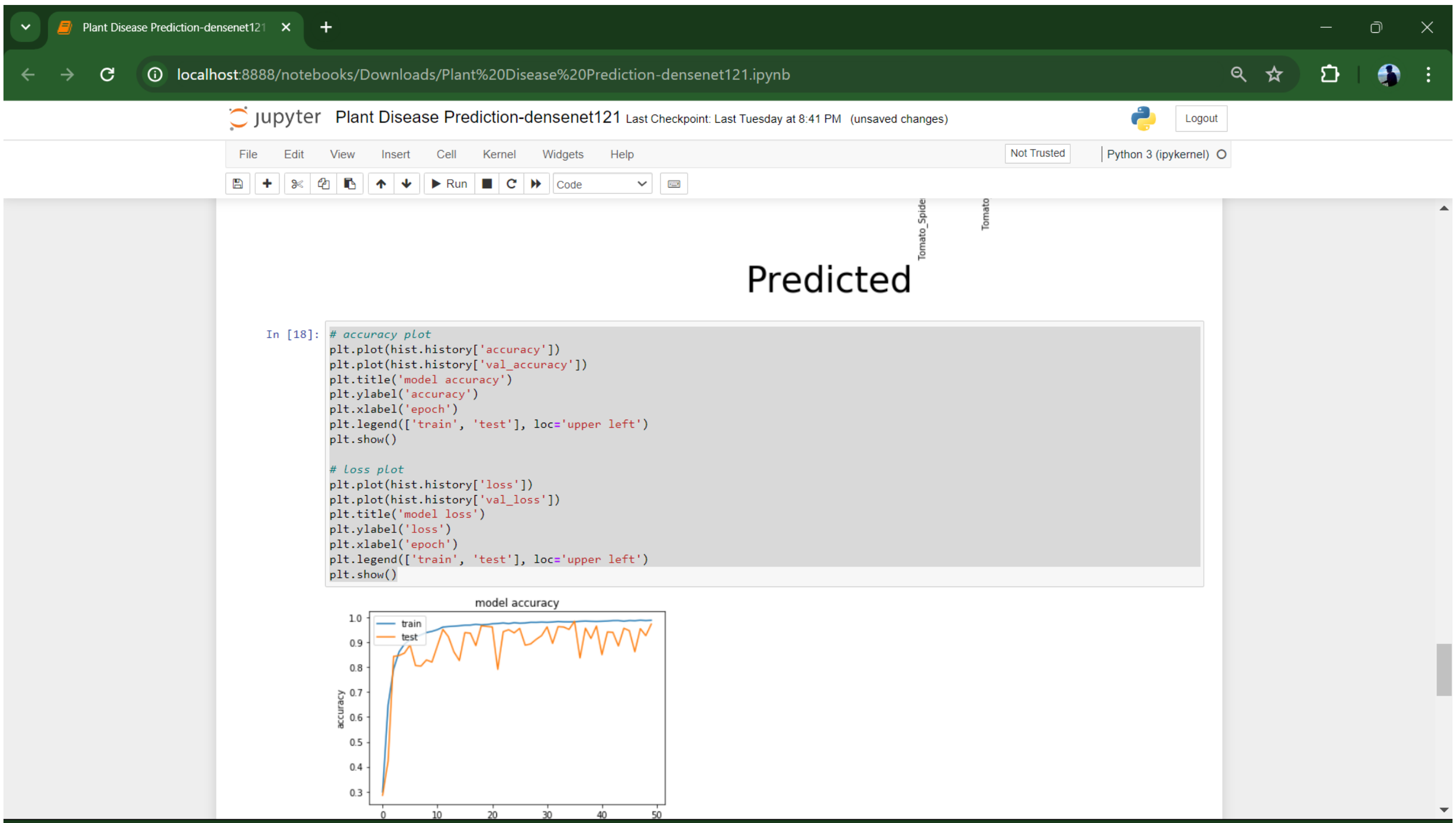
```
In [15]: model = build_densenet()
         annealer = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5, patience=5, verbose=1, min_lr=1e-3)
         checkpoint = ModelCheckpoint('model.h5', verbose=1, save_best_only=True)
         # Generates batches of image data with data augmentation
         datagen = ImageDataGenerator(rotation_range=360, # Degree range for random rotations
                                 width_shift_range=0.2, # Range for random horizontal shifts
                                 height_shift_range=0.2, # Range for random vertical shifts
                                 zoom_range=0.2, # Range for random zoom
                                 horizontal_flip=True, # Randomly flip inputs horizontally
                                 vertical_flip=True) # Randomly flip inputs vertically

         datagen.fit(X_train)
         # Fits the model on batches with real-time data augmentation
         hist = model.fit_generator(datagen.flow(X_train, Y_train, batch_size=BATCH_SIZE),
                        steps_per_epoch=X_train.shape[0] // BATCH_SIZE,
                        epochs=EPOCHS,
                        verbose=2,
                        callbacks=[annealer, checkpoint],
                        validation_data=(X_val, Y_val))
```

```
Downloading data from https://github.com/keras-team/keras-applications/releases/download/densenet/densenet121_weights_tf_dim_
ordering_tf_kernels_notop.h5
29089792/29084464 [==============================] - 1s 0us/step
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 64, 64, 3)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 3)         84
_________________________________________________________________
densenet121 (Model)          multiple                  7037504
_________________________________________________________________
global_average_pooling2d_1 ( (None, 1024)              0
_________________________________________________________________
batch_normalization_1 (Batch (None, 1024)              4096
_________________________________________________________________
```


Plant Disease Prediction-densenet121
localhost:8888/notebooks/Downloads/Plant%20Disease%20Prediction-densenet121.ipynb
jupyter Plant Disease Prediction-densenet121 Last Checkpoint: Last Tuesday at 8:41 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [16]: #model = load_model('../output/kaggle/working/model.h5')
         final_loss, final_accuracy = model.evaluate(X_val, Y_val)
         print('Final Loss: {}, Final Accuracy: {}'.format(final_loss, final_accuracy))

         4128/4128 [==============================] - 5s 1ms/step
         Final Loss: 0.09057462476365369, Final Accuracy: 0.9738371968269348

In [17]: Y_pred = model.predict(X_val)

         Y_pred = np.argmax(Y_pred, axis=1)
         Y_true = np.argmax(Y_val, axis=1)

         cm = confusion_matrix(Y_true, Y_pred)
         plt.figure(figsize=(12, 12))
         ax = sns.heatmap(cm, cmap=plt.cm.Greens, annot=True, square=True, xticklabels=disease_types, yticklabels=disease_types)
         ax.set_ylabel('Actual', fontsize=40)
         ax.set_xlabel('Predicted', fontsize=40)

Out[17]: Text(0.5, 144.41374999999996, 'Predicted')
```

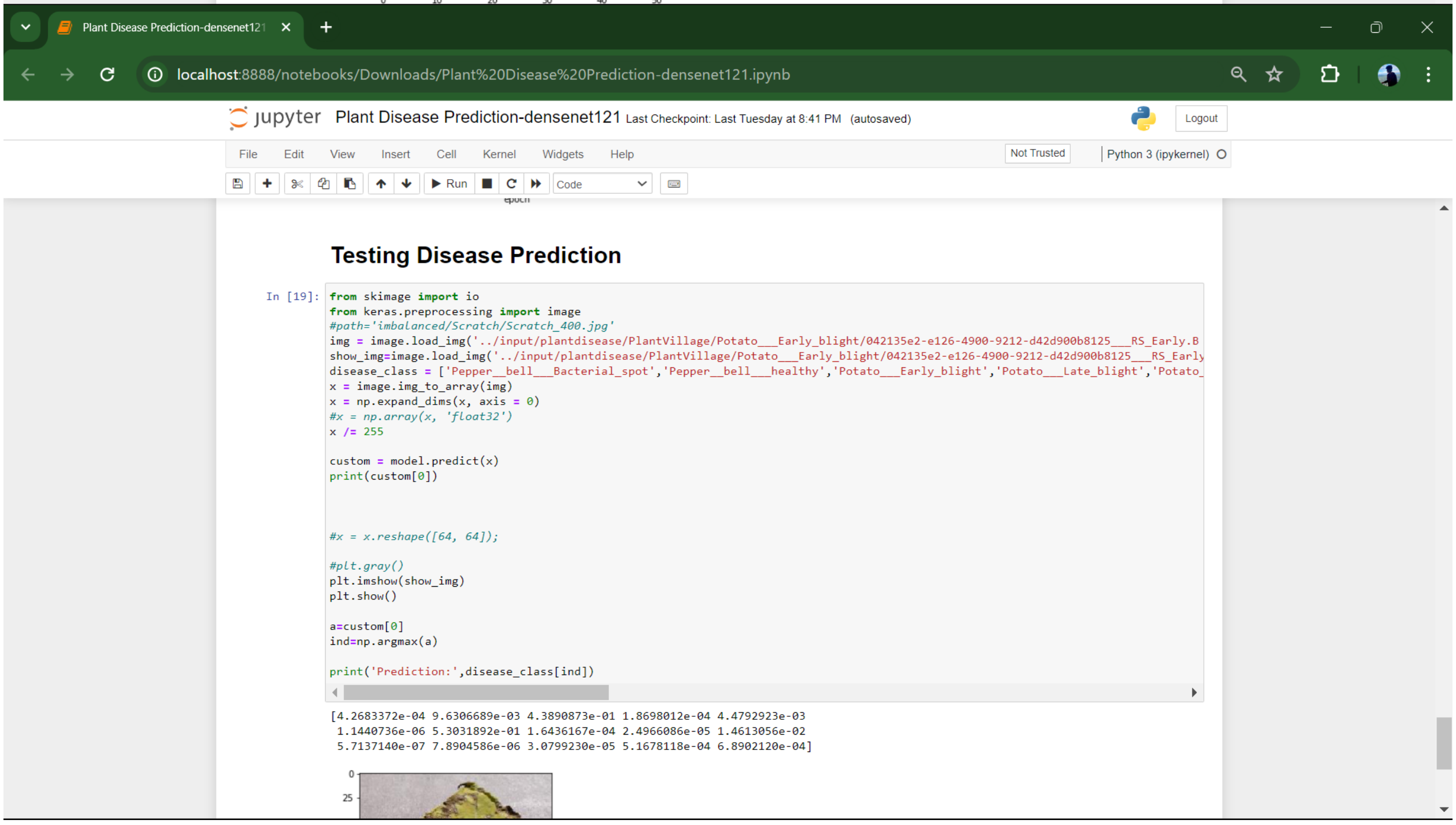

Actual
Pepper__bell__Bacterial_spot 2e+02 0 0 0 0 0 0 0 0 1 0 0 0 0 0
Pepper__bell__healthy 0 2.9e+02 0 0 9 0 0 0 0 0 0 0 0 0 0
Potato__Early_blight 0 0 2.1e+02 0 0 0 0 0 0 0 0 0 0 0 0
Potato__Late_blight 0 0 0 2.0e+02 0 0 0 0 0 0 0 0 0 0 0
Potato__healthy 0 0 0 0 28 0 0 0 0 0 0 0 0 0 0
Tomato_Bacterial_spot 0 0 0 1 0 3.9e+02 2 0 0 1 0 9 0 0 0
Tomato_Early_blight 0 0 0 0 0 0 2.0e+02 1 0 0 0 0 0 0 0
Tomato_Late_blight 0 0 1 4 0 5 4 3.6e+02 0 0 0 0 0 0 1
Tomato_Leaf_Mold 0 0 0 0 0 0 0 0 1.8e+02 4 0 0 0 0 0
Tomato_Septoria_leaf_spot 0 0 0 0 0 0 0 0 0 3.6e+02 0 2 0 0 1
Tomato_Spider_mites_Two_spotted_spider_mite 0 0 0 0 3 0 0 1 0 0 2.8e+02 8 0 1 28
Tomato__Target_Spot 0 0 0 1 0 0 2 0 0 1 2 2.8e+02 0 0 8
Tomato__Tomato_YellowLeaf__Curl_Virus 0 0 0 0 0 4 0 0 0 0 1 0 6.2e+02 0 2
Tomato__Tomato_mosaic_virus 0 0 0 0 0 0 0 0 0 0 0 0 0 81 0
Tomato_healthy 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3.3e+02
Pepper__bell__healthy
Potato__Early_blight
Potato__Late_blight
Potato__healthy
Tomato_Bacterial_spot
Tomato_Early_blight
Tomato_Late_blight
Tomato_Leaf_Mold
Tomato__Target_Spot
Tomato_Tomato_mosaic_virus
Tomato_healthy
600
500
400
300
200
100
0


Plant Disease Prediction-densenet121
localhost:8888/notebooks/Downloads/Plant%20Disease%20Prediction-densenet121.ipynb
jupyter Plant Disease Prediction-densenet121 Last Checkpoint: Last Tuesday at 8:41 PM (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Tomato_Spide
Tomato
Predicted

In [18]:

```
# accuracy plot
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# loss plot
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

model accuracy
train
test
accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
10
20
30
40
50

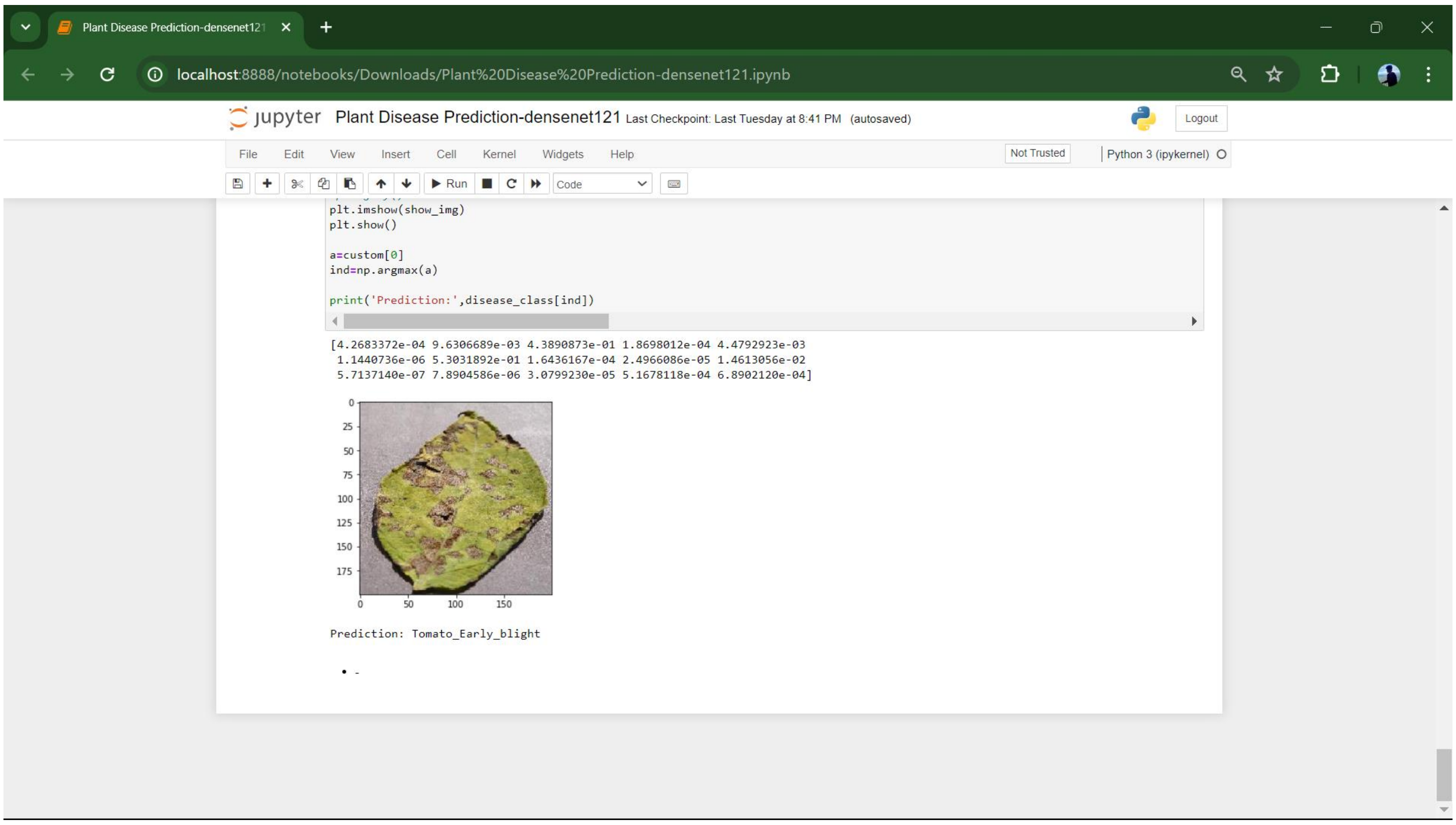

Plant Disease Prediction-densenet121
localhost:8888/notebooks/Downloads/Plant%20Disease%20Prediction-densenet121.ipynb
jupyter Plant Disease Prediction-densenet121 Last Checkpoint: Last Tuesday at 8:41 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
epoch

Testing Disease Prediction

In [19]:

```
from skimage import io
from keras.preprocessing import image
#path='imbalanced/Scratch/Scratch_400.jpg'
img = image.load_img('../input/plantdisease/PlantVillage/Potato___Early_blight/042135e2-e126-4900-9212-d42d900b8125___RS_Early.B
show_img=image.load_img('../input/plantdisease/PlantVillage/Potato___Early_blight/042135e2-e126-4900-9212-d42d900b8125___RS_Early
disease_class = ['Pepper__bell___Bacterial_spot','Pepper__bell___healthy','Potato___Early_blight','Potato___Late_blight','Potato_
x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0)
#x = np.array(x, 'float32')
x /= 255

custom = model.predict(x)
print(custom[0])



#x = x.reshape([64, 64]);

#plt.gray()
plt.imshow(show_img)
plt.show()

a=custom[0]
ind=np.argmax(a)

print('Prediction:',disease_class[ind])
```

```
[4.2683372e-04 9.6306689e-03 4.3890873e-01 1.8698012e-04 4.4792923e-03
 1.1440736e-06 5.3031892e-01 1.6436167e-04 2.4966086e-05 1.4613056e-02
 5.7137140e-07 7.8904586e-06 3.0799230e-05 5.1678118e-04 6.8902120e-04]
```


Plant Disease Prediction-densenet121
localhost:8888/notebooks/Downloads/Plant%20Disease%20Prediction-densenet121.ipynb
jupyter Plant Disease Prediction-densenet121 Last Checkpoint: Last Tuesday at 8:41 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
plt.imshow(show_img)
plt.show()

a=custom[0]
ind=np.argmax(a)

print('Prediction:',disease_class[ind])
```

```
[4.2683372e-04 9.6306689e-03 4.3890873e-01 1.8698012e-04 4.4792923e-03
 1.1440736e-06 5.3031892e-01 1.6436167e-04 2.4966086e-05 1.4613056e-02
 5.7137140e-07 7.8904586e-06 3.0799230e-05 5.1678118e-04 6.8902120e-04]
```

0
25
50
75
100
125
150
175
0
50
100
150

```
Prediction: Tomato_Early_blight
```

CONCLUSION FUTURE ENHANCEMENTS

Conclusion:
In conclusion, the development of a plant disease prediction system based on data-driven techniques and machine learning algorithms represents a significant advancement in agricultural technology. This system serves as a valuable tool for farmers and agricultural experts seeking accurate and timely guidance in managing crop health and yield. By analyzing key indicators and leveraging advanced algorithms, the system provides personalized disease predictions tailored to specific crops and environmental conditions. Through this project, we have demonstrated the feasibility and effectiveness of using data-driven approaches to enhance crop management strategies, ultimately contributing to sustainable agriculture practices and food security.

Future Enhancements:-

1. High-resolution imaging: Advances in imaging technologies such as hyperspectral imaging and multispectral imaging can provide detailed and accurate images of plants. This can help in detecting subtle changes in plant health due to diseases at an early stage.

2. Machine learning and AI: Continued progress in machine learning algorithms can improve disease detection accuracy. AI can be trained on vast datasets of plant diseases, symptoms, and environmental factors to develop more robust and accurate predictive models.

3. Internet of Things (IoT) sensors: IoT devices can monitor various parameters such as temperature, humidity, soil moisture, and nutrient levels in real-time. Integrating this data with disease detection algorithms can provide a comprehensive view of plant health and early warning systems for disease outbreaks.

4. Drone technology: Drones equipped with cameras and sensors can cover large agricultural areas quickly and efficiently. They can capture aerial images for analysis, helping identify disease patterns across fields and enabling targeted interventions.

5. Blockchain for data validation: Blockchain technology can be used to securely store and validate data related to plant diseases, treatments, and outcomes. This can enhance transparency, traceability, and trust in the information shared among stakeholders in the agricultural sector.

6. Mobile apps for farmers: User-friendly mobile applications can empower farmers with tools for real-time disease diagnosis and management recommendations. These apps can leverage image recognition, AI algorithms, and expert knowledge to assist farmers in making informed decisions about crop health.

7. Bioinformatics and genomics: Advances in bioinformatics and genomics can lead to the development of disease-resistant plant varieties through genetic engineering or selective breeding. Understanding the genetic basis of plant diseases can also improve diagnostic techniques and treatment strategies.

8. Collaborative platforms: Online platforms that facilitate collaboration among researchers, agronomists, farmers, and industry experts can accelerate knowledge sharing and innovation in plant disease detection and management. Crowdsourcing data and expertise can lead to more comprehensive solutions.

9. Robotics and automation: Robotic systems can be designed for autonomous scouting, monitoring, and treatment of plant diseases. These robots can navigate fields, collect data, and perform targeted interventions such as applying pesticides or beneficial microorganisms.

10. Climate and environmental modeling: Integrating climate and environmental data into disease prediction models can improve the accuracy of forecasts regarding disease prevalence and severity. Understanding how climate change impacts disease dynamics is crucial for developing adaptive strategies.

These enhancements collectively can revolutionize plant disease detection by making it more precise, timely, and scalable, thereby contributing to sustainable agriculture and food security.

Result:

Hence the **Crop Disease Detection** project was carried out successfully using Deep Learning techniques.