Department of Information Science and Engineering

U18ISE0001 – IMAGE AND VIDEO ANALYTICS

Project Report on Plant Disease Classification into healthy, powdery or rust using Tensorflow and Keras

SUBMITTED BY

Jahfiyathul Firthous A 21BIS011

Nithyashree A 21BIS020

Nivethika R 21BIS022

CODE:

```
import os


def total_files(folder_path):
    num_files = len([f for f in os.listdir(folder_path) if
os.path.isfile(os.path.join(folder_path, f))])
    return num_files
```

```
train_files_healthy = "Dataset/Train/Train/Healthy"
train_files_powdery = "Dataset/Train/Train/Powdery"
train_files_rust = "Dataset/Train/Train/Rust"

test_files_healthy = "Dataset/Test/Test/Healthy"
test_files_powdery = "Dataset/Test/Test/Powdery"
test_files_rust = "Dataset/Test/Test/Rust"

valid_files_healthy = "Dataset/Validation/Validation/Healthy"
valid_files_powdery = "Dataset/Validation/Validation/Powdery"
valid_files_rust = "Dataset/Validation/Validation/Rust"

print("Number of healthy leaf images in training set",
total_files(train_files_healthy))
print("Number of powder leaf images in training set",
total_files(train_files_powdery))
print("Number of rusty leaf images in training set", total_files(train_files_rust))

print("======================================================
==")

print("Number of healthy leaf images in test set", total_files(test_files_healthy))
print("Number of powder leaf images in test set", total_files(test_files_powdery))
print("Number of rusty leaf images in test set", total_files(test_files_rust))

print("======================================================
==")
```

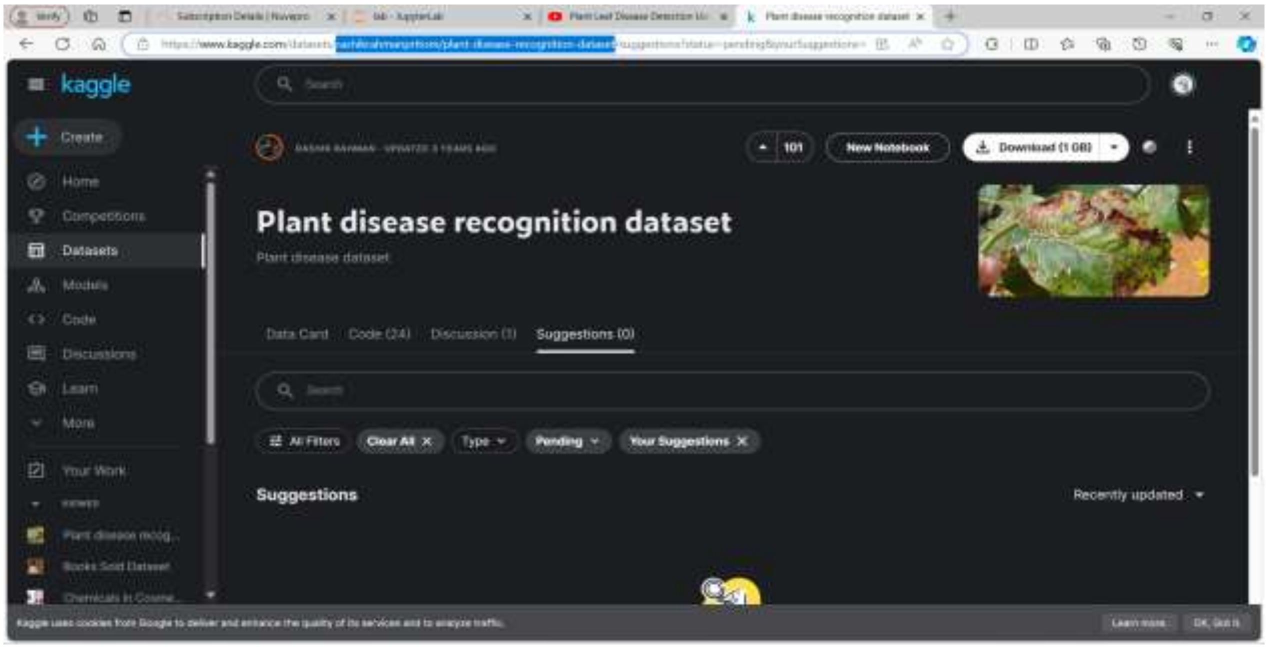
```
print("Number of healthy leaf images in validation set",
total_files(valid_files_healthy))

print("Number of powder leaf images in validation set",
total_files(valid_files_powdery))

print("Number of rusty leaf images in validation set", total_files(valid_files_rust))


from PIL import Image

import IPython.display as display


image_path = 'Dataset/Train/Train/Healthy/8ce77048e12f3dd4.jpg'


with open(image_path, 'rb') as f:

    display.display(display.Image(data=f.read(), width=500))


image_path = 'Dataset/Train/Train/Rust/80f09587dfc7988e.jpg'


with open(image_path, 'rb') as f:

    display.display(display.Image(data=f.read(), width=500))


from keras.preprocessing.image import ImageDataGenerator


train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2,
zoom_range=0.2, horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)
```

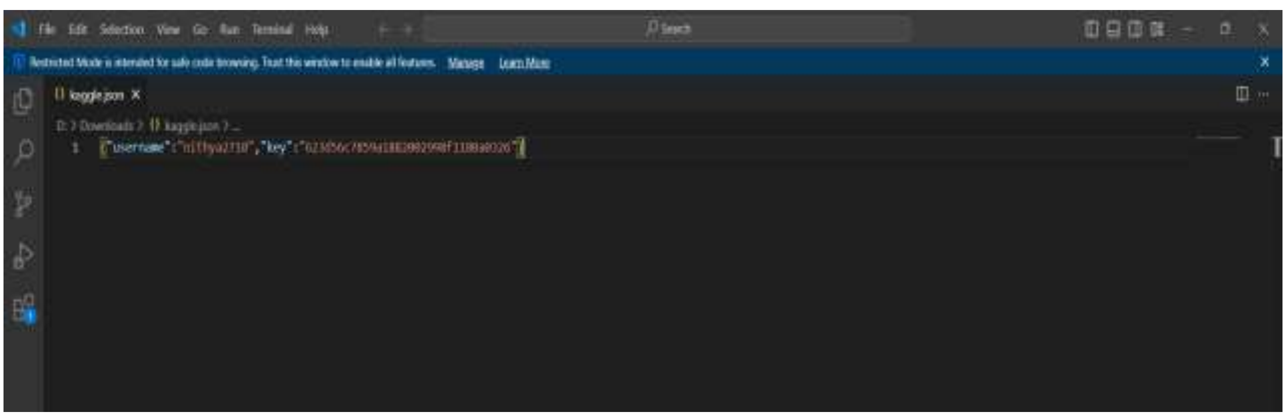
```
train_generator = train_datagen.flow_from_directory('Dataset/Train/Train',
                                                    target_size=(225, 225),
                                                    batch_size=32,
                                                    class_mode='categorical')

validation_generator =
test_datagen.flow_from_directory('Dataset/Validation/Validation',
                                                        target_size=(225, 225),
                                                        batch_size=32,
                                                        class_mode='categorical')

from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense



model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(225, 225, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(3, activation='softmax'))

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

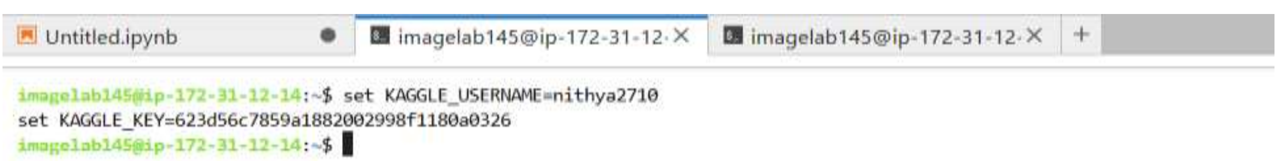
```
history = model.fit(train_generator,
                    batch_size=16,
                    epochs=5,
                    validation_data=validation_generator,
                    validation_batch_size=16
                    )

from matplotlib import pyplot as plt
from matplotlib.pyplot import figure

import seaborn as sns
sns.set_theme()
sns.set_context("poster")

figure(figsize=(25, 25), dpi=100)

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

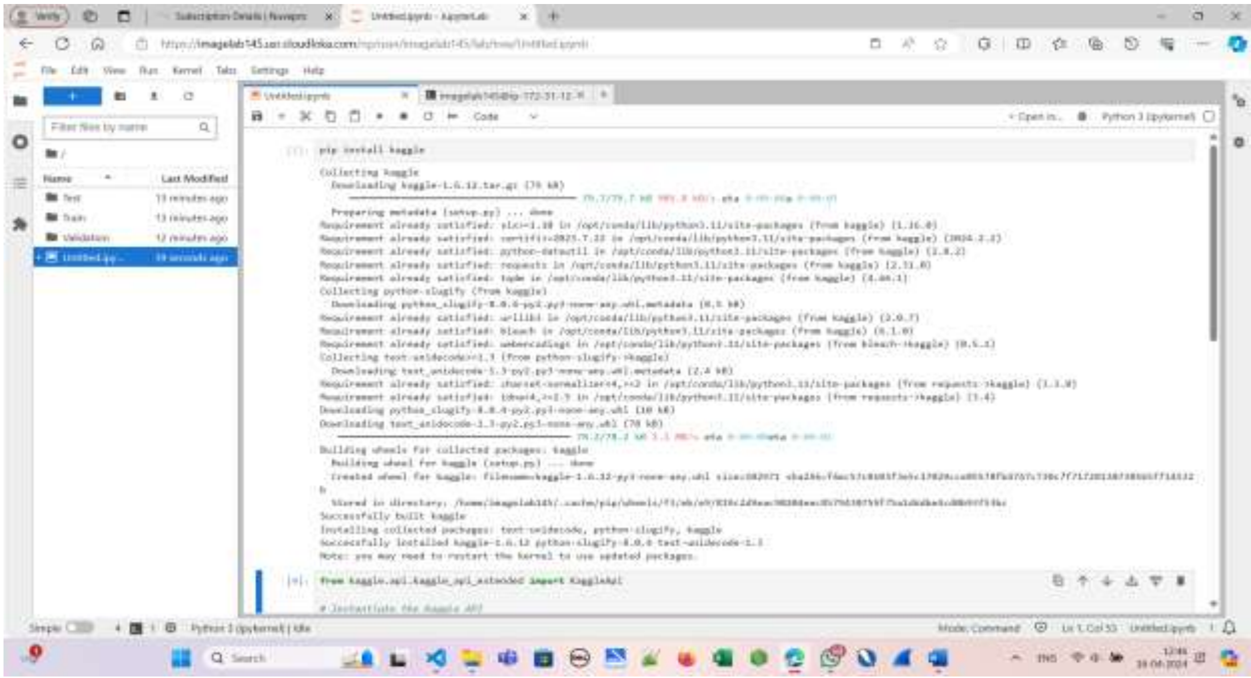
```
model.save("model.h5")


from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np

def preprocess_image(image_path, target_size=(225, 225)):
    img = load_img(image_path, target_size=target_size)
    x = img_to_array(img)
    x = x.astype('float32') / 255.
    x = np.expand_dims(x, axis=0)
    return x

x = preprocess_image('Dataset/Test/Test/Rust/82f49a4a7b9585f1.jpg')

predictions = model.predict(x)
predictions[0]

labels = train_generator.class_indices
labels = {v: k for k, v in labels.items()}
labels

predicted_label = labels[np.argmax(predictions)]
print(predicted_label)
```

SCREENSHOTS :

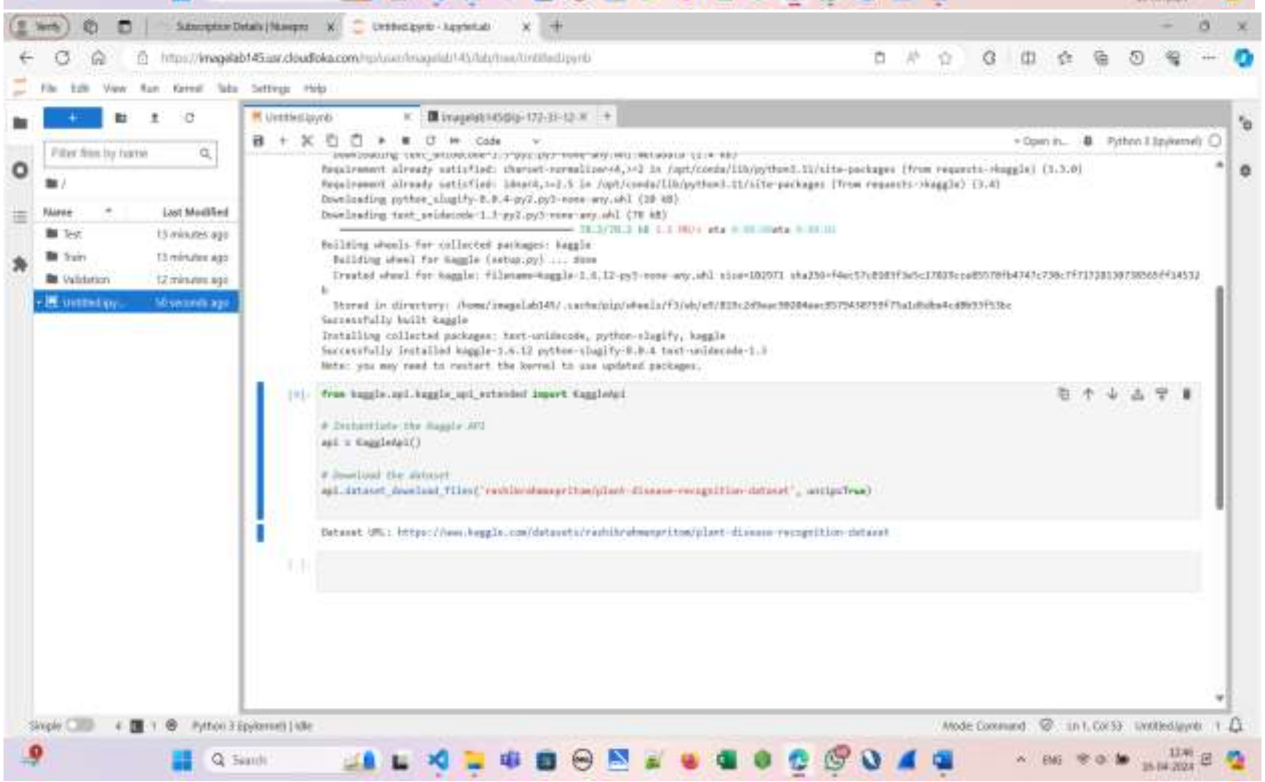




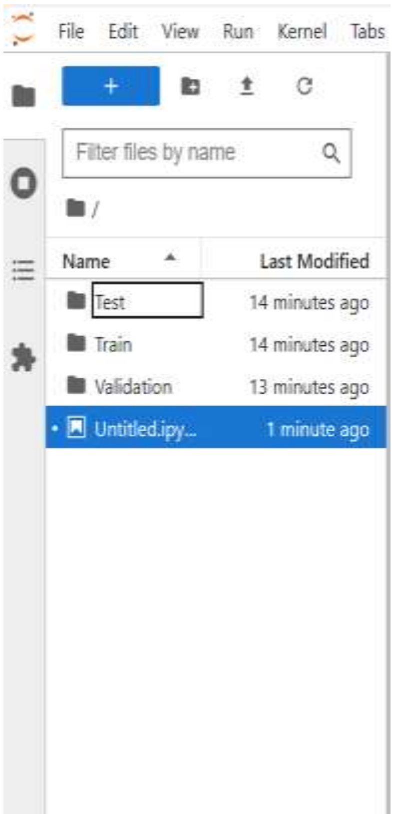


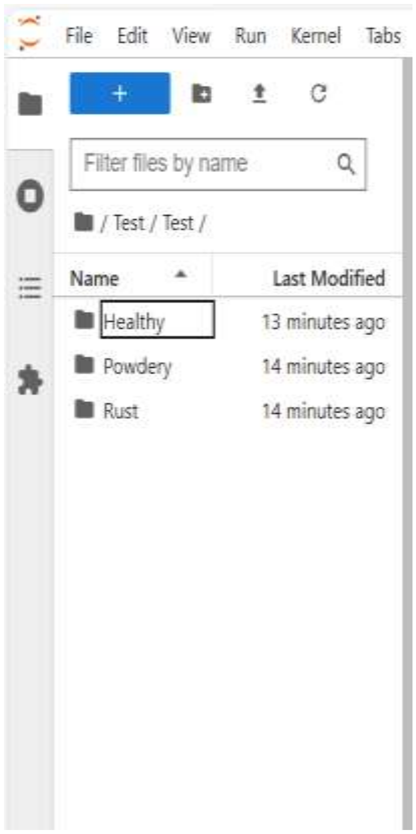


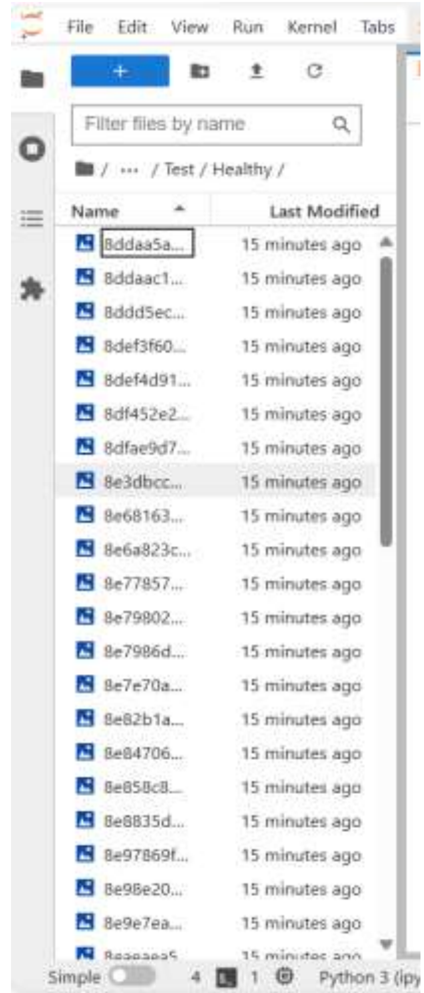
pip install kaggle
Collecting kaggle
Successfully built kaggle
Installing collected packages: text-unidecode, python-slugify, kaggle
Note: you may need to restart the kernel to use updated packages.
from kaggle.api.kaggle_api_extended import KaggleApi

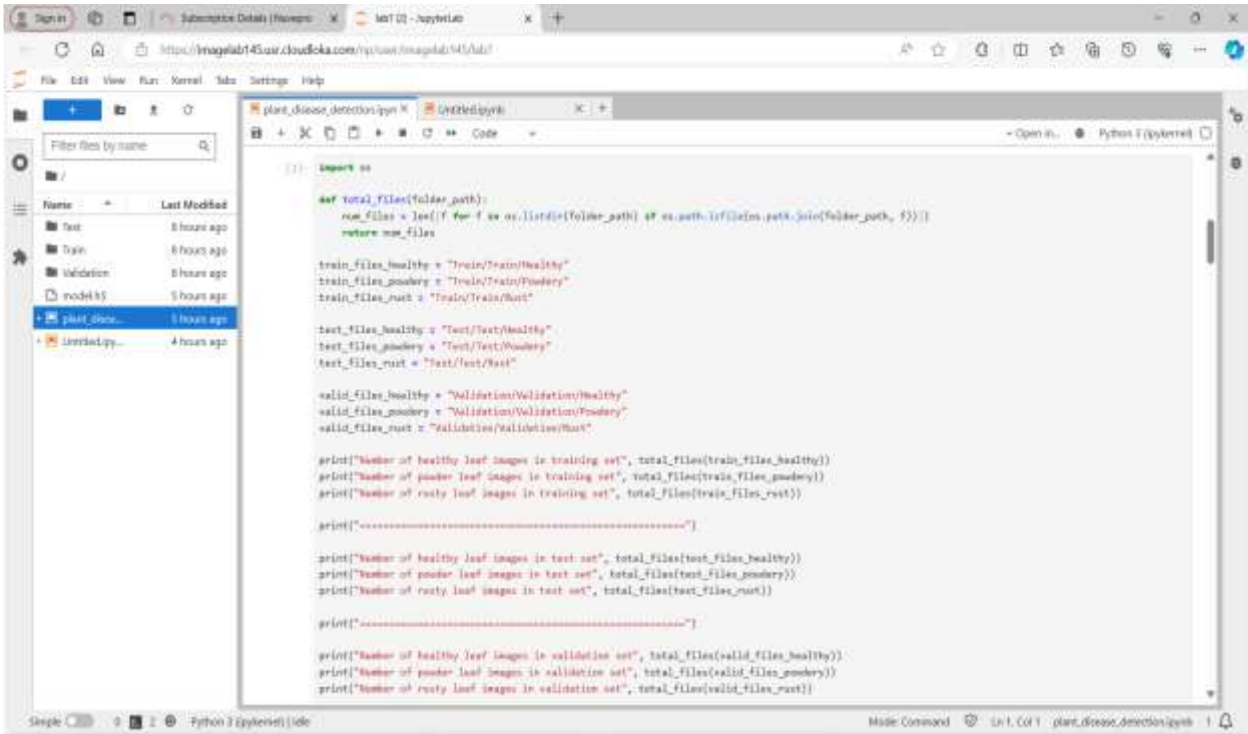


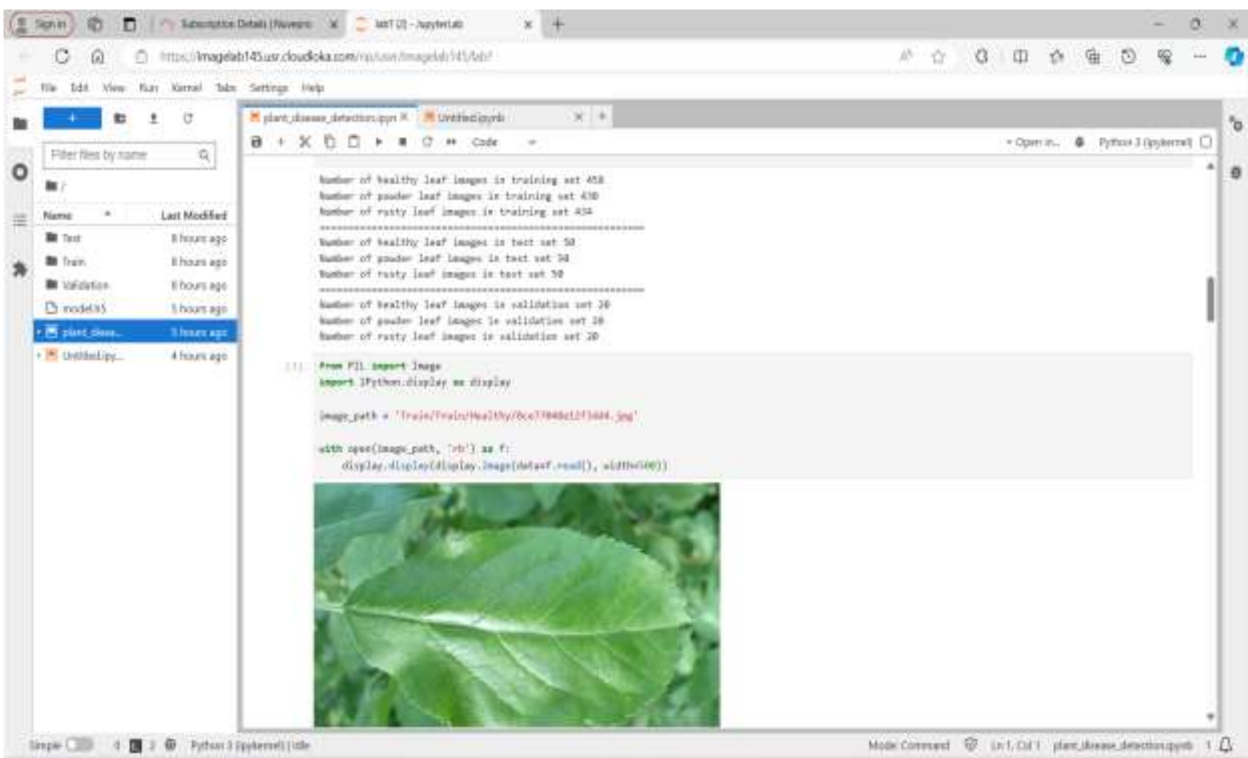
Successfully built kaggle
Installing collected packages: text-unidecode, python-slugify, kaggle
Note: you may need to restart the kernel to use updated packages.
from kaggle.api.kaggle_api_extended import KaggleApi
# Instantiate the Kaggle API
api = KaggleApi()
# Download the dataset

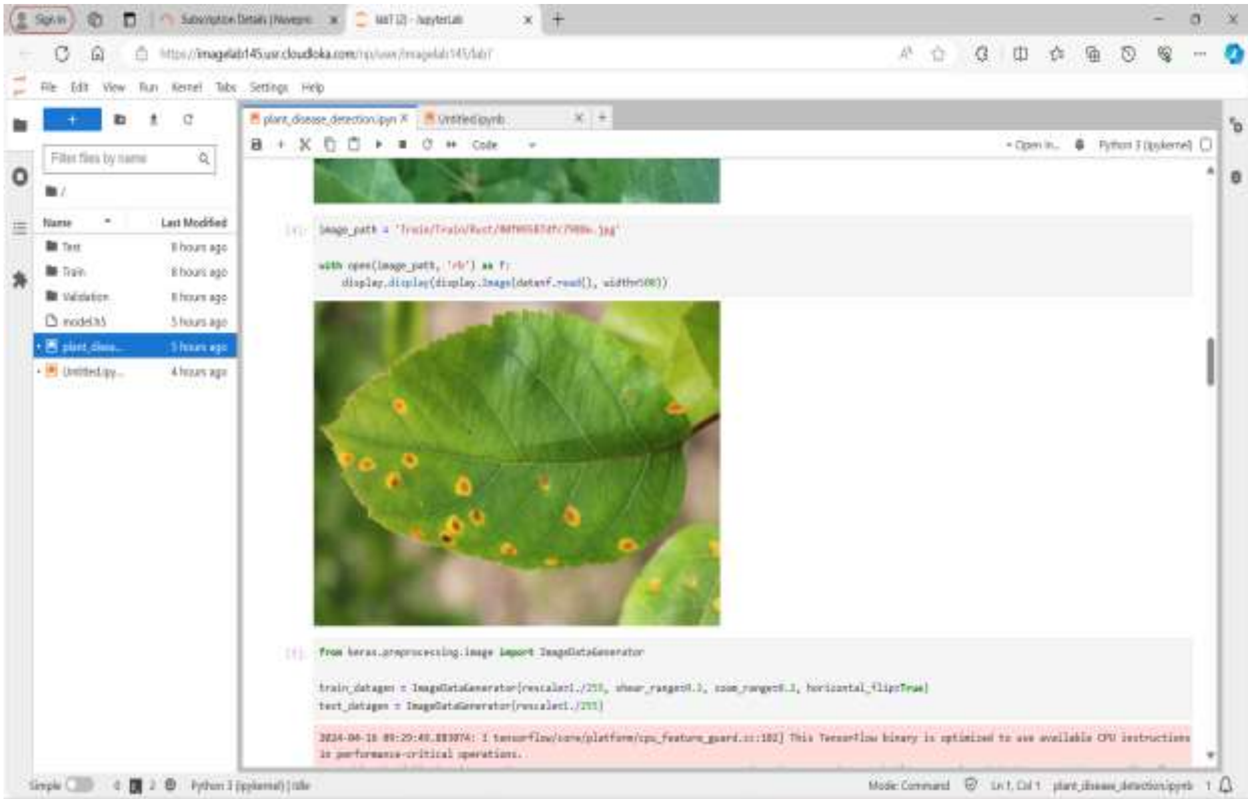

File Edit View Run Kernel Tabs
+
Filter files by name
/
Name
Last Modified
Test
14 minutes ago
Train
14 minutes ago
Validation
13 minutes ago
Untitled.ipy...
1 minute ago

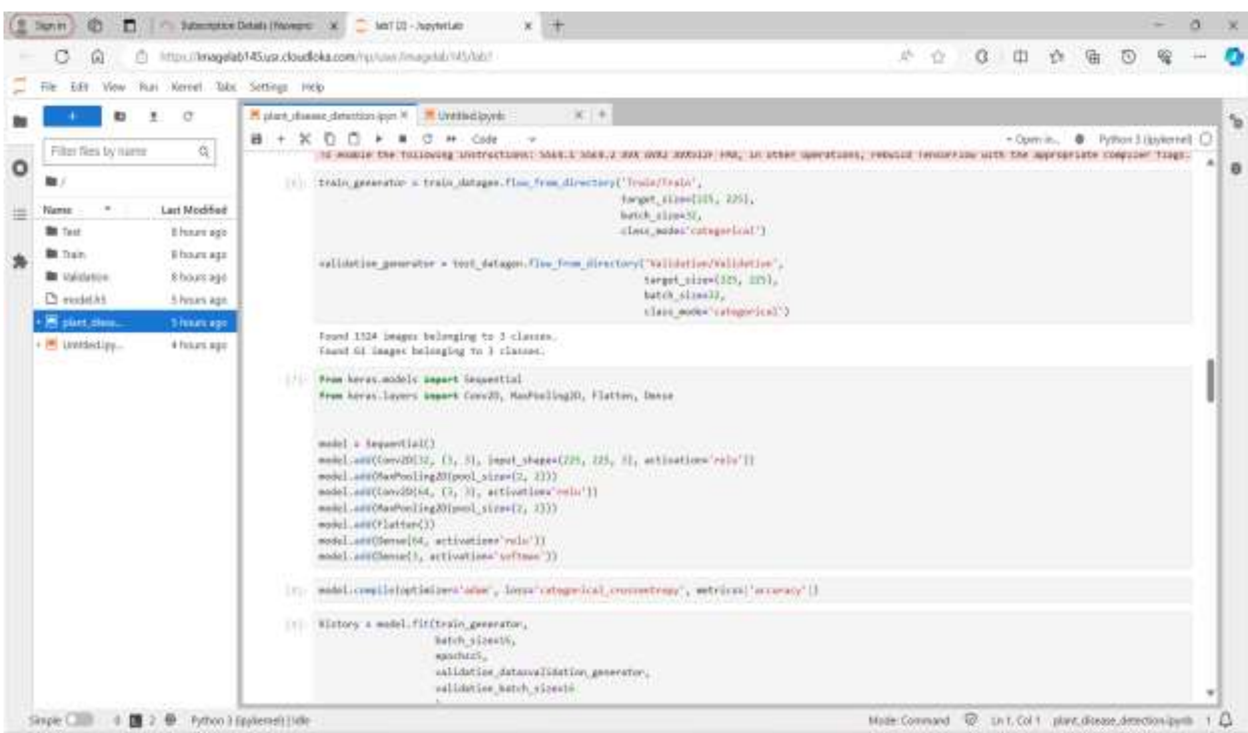

File Edit View Run Kernel Tabs
+
Filter files by name
/ Test / Test /
Name
Last Modified
Healthy
13 minutes ago
Powdery
14 minutes ago
Rust
14 minutes ago

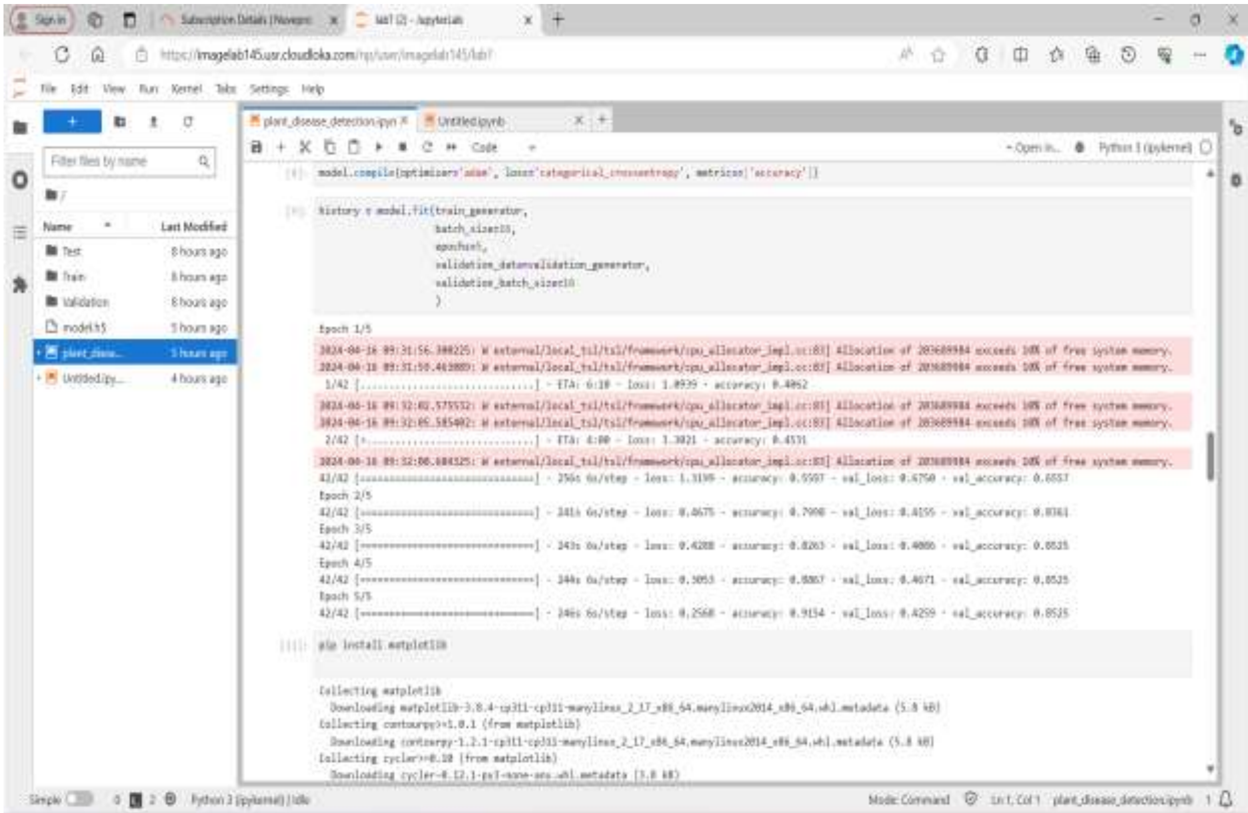

File Edit View Run Kernel Tabs
+
Filter files by name
/ ··· / Test / Healthy /
Name
Last Modified
8ddaa5a...
15 minutes ago
8ddaac1...
15 minutes ago
8ddd5ec...
15 minutes ago
8def3f60...
15 minutes ago
8def4d91...
15 minutes ago
8df452e2...
15 minutes ago
8dfae9d7...
15 minutes ago
8e3dbcc...
15 minutes ago
8e68163...
15 minutes ago
8e6a823c...
15 minutes ago
8e77857...
15 minutes ago
8e79802...
15 minutes ago
8e7986d...
15 minutes ago
8e7e70a...
15 minutes ago
8e82b1a...
15 minutes ago
8e84706...
15 minutes ago
8e858c8...
15 minutes ago
8e8835d...
15 minutes ago
8e97869f...
15 minutes ago
8e98e20...
15 minutes ago
8e9e7ea...
15 minutes ago
Simple
4
1
Python 3 (ipy

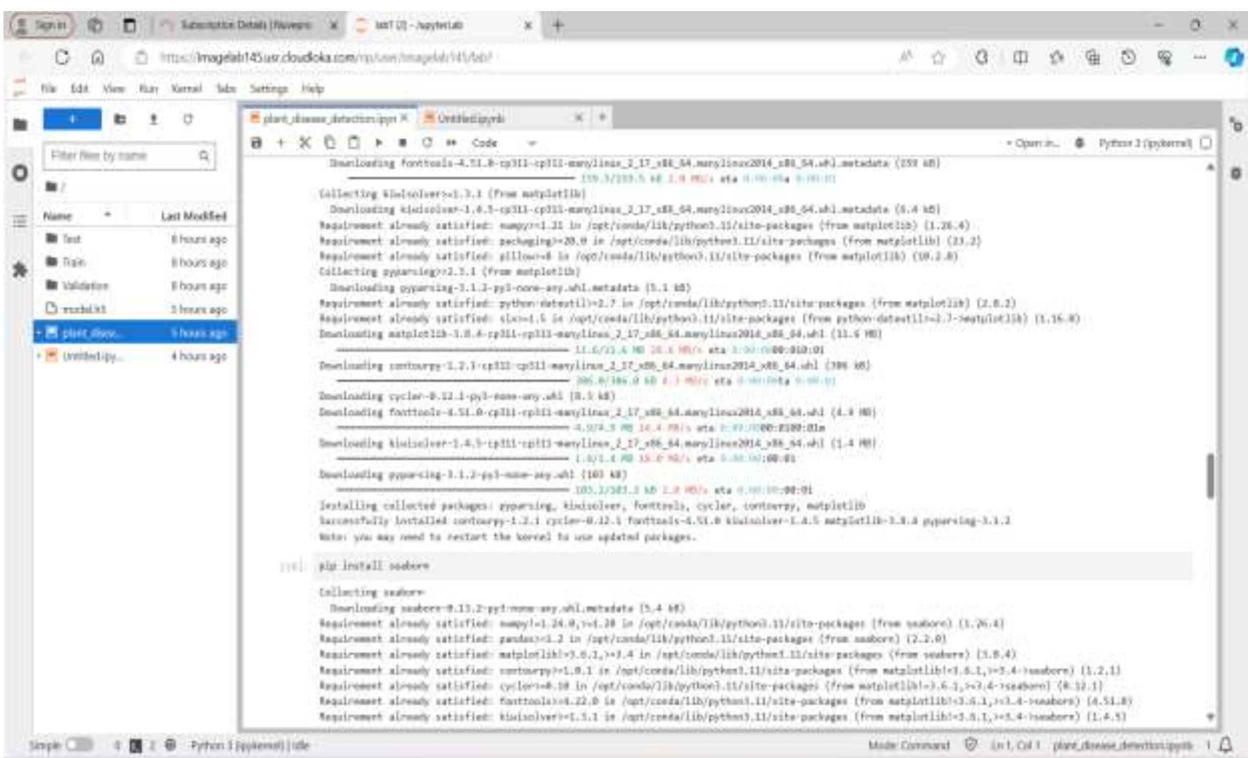

plant_disease_detection.ipynb
Untitled.ipynb
Code
Filter files by name
Name
Last Modified
Test
Train
Validation
model.h5
import os
def total_files(folder_path):
num_files = len([f for f in os.listdir(folder_path) if os.path.isfile(os.path.join(folder_path, f))])
return num_files
train_files_healthy = "Train/Train/Healthy"
train_files_powdery = "Train/Train/Powdery"
train_files_rust = "Train/Train/Rust"
test_files_healthy = "Test/Test/Healthy"
test_files_powdery = "Test/Test/Powdery"
test_files_rust = "Test/Test/Rust"
valid_files_healthy = "Validation/Validation/Healthy"
valid_files_powdery = "Validation/Validation/Powdery"
valid_files_rust = "Validation/Validation/Rust"
print("Number of healthy leaf images in training set", total_files(train_files_healthy))
print("Number of powder leaf images in training set", total_files(train_files_powdery))
print("Number of rusty leaf images in training set", total_files(train_files_rust))
print("Number of healthy leaf images in test set", total_files(test_files_healthy))
print("Number of powder leaf images in test set", total_files(test_files_powdery))
print("Number of rusty leaf images in test set", total_files(test_files_rust))
print("Number of healthy leaf images in validation set", total_files(valid_files_healthy))
print("Number of powder leaf images in validation set", total_files(valid_files_powdery))
print("Number of rusty leaf images in validation set", total_files(valid_files_rust))
Mode: Command
Python 3 (ipykernel) | Idle

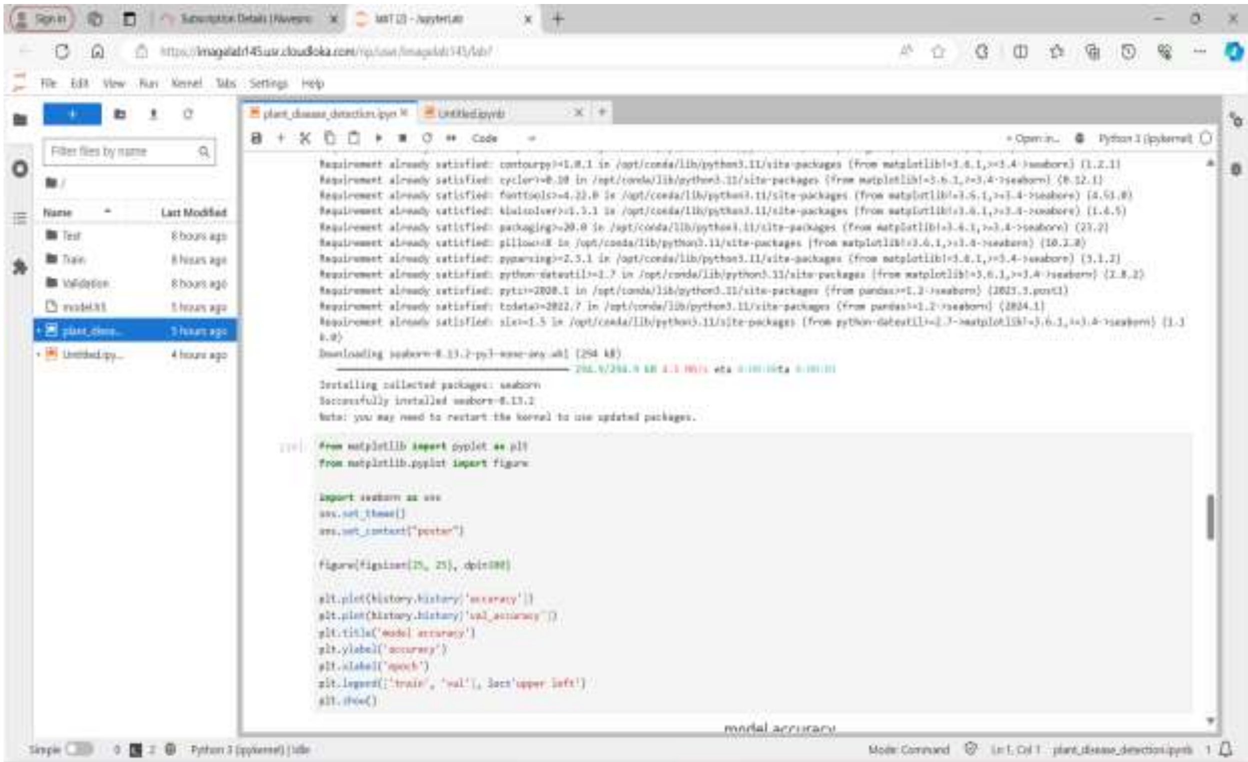

plant_disease_detection.ipynb
Untitled.ipynb
Number of healthy leaf images in training set 458
Number of powder leaf images in training set 430
Number of rusty leaf images in training set 434
Number of healthy leaf images in test set 50
Number of powder leaf images in test set 50
Number of rusty leaf images in test set 50
Number of healthy leaf images in validation set 20
Number of powder leaf images in validation set 20
Number of rusty leaf images in validation set 20
from PIL import Image
import IPython.display as display
with open(image_path, 'rb') as f:
Mode: Command
Python 3 (ipykernel) | Idle

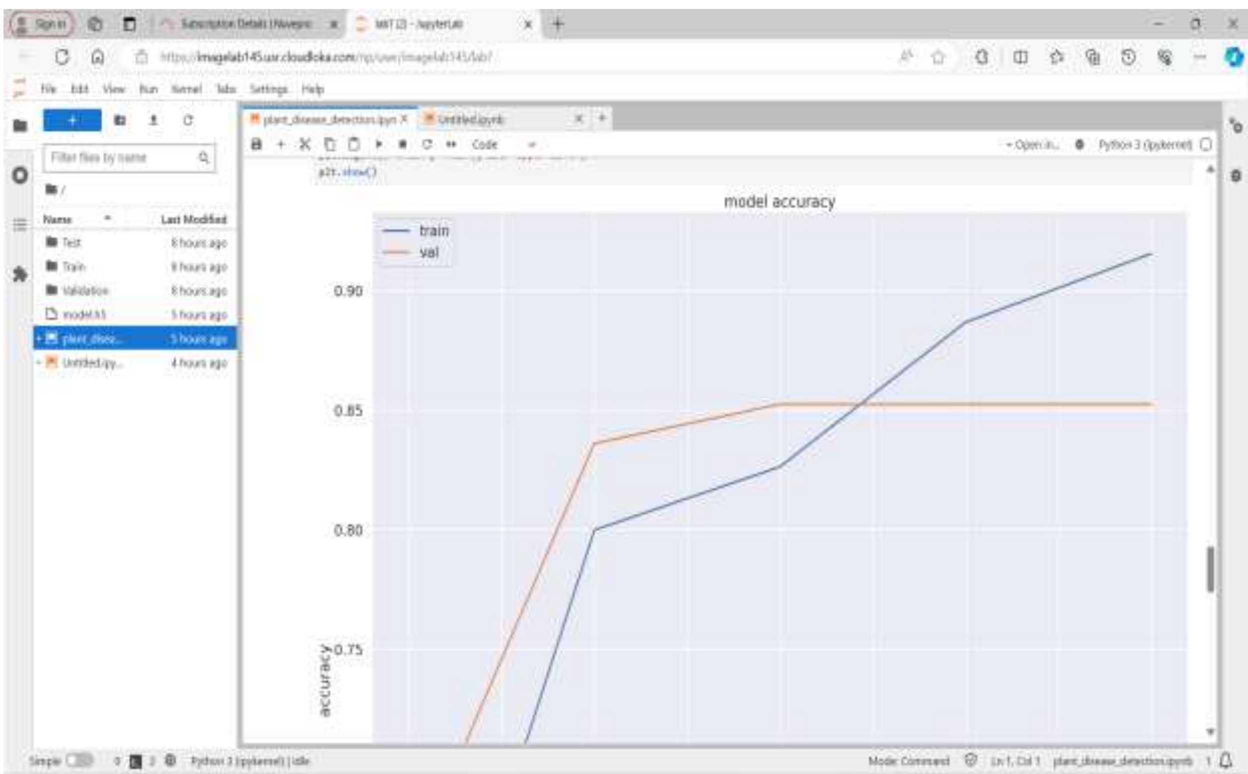

plant_disease_detection.ipynb
Untitled.ipynb
Code
Python 3 (ipykernel)
Name
Last Modified
Test
Train
Validation
model.h5
image_path = 'Train/Train/Rust/...jpg'
with open(image_path, 'rb') as f:
display.display(display.Image(data=f.read(), width=500))
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
Mode: Command
Ln 1, Col 1
plant_disease_detection.ipynb

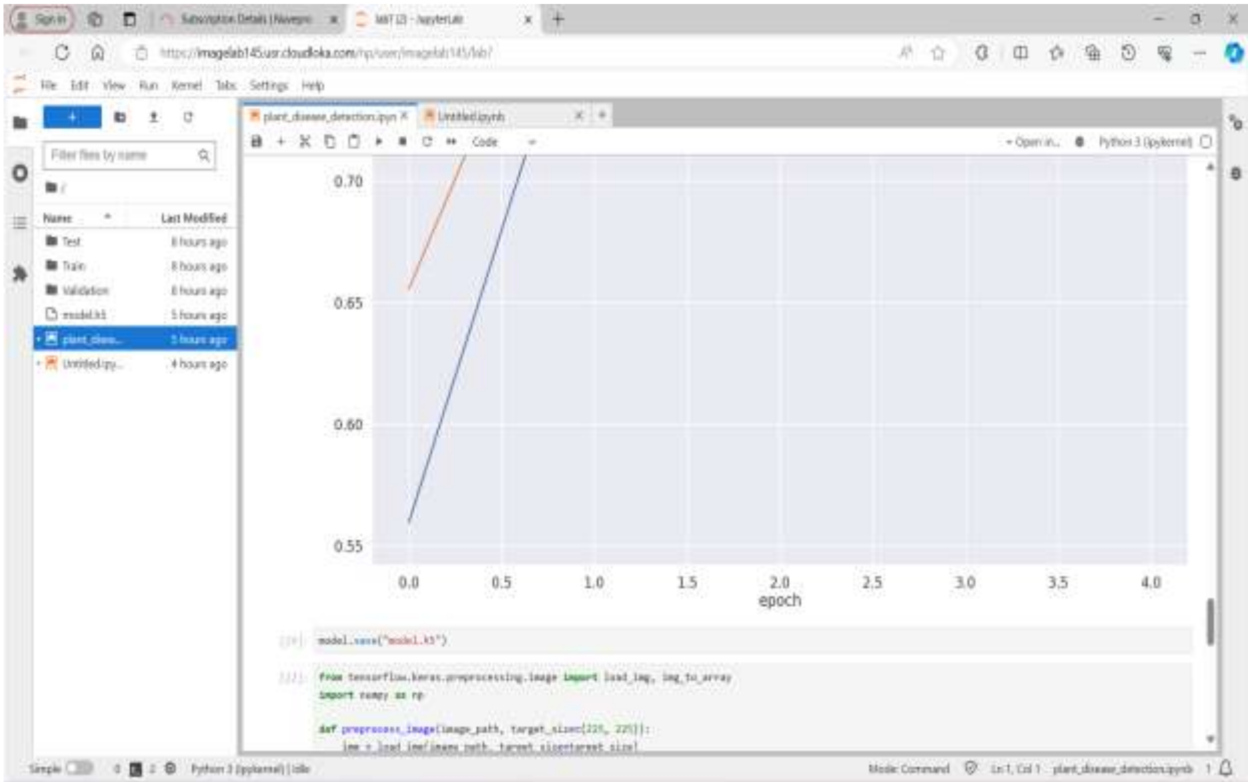

plant_disease_detection.ipynb
Untitled.ipynb
Code
Python 3 (ipykernel)
train_generator = train_datagen.flow_from_directory('Train/Train',
target_size=(225, 225),
batch_size=32,
class_mode='categorical')
validation_generator = test_datagen.flow_from_directory('Validation/Validation',
target_size=(225, 225),
batch_size=32,
class_mode='categorical')
Found 1324 images belonging to 3 classes.
Found 61 images belonging to 3 classes.
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(225, 225, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_generator,
batch_size=16,
epochs=5,
validation_data=validation_generator,
validation_batch_size=16
Mode: Command
Ln 1, Col 1
plant_disease_detection.ipynb

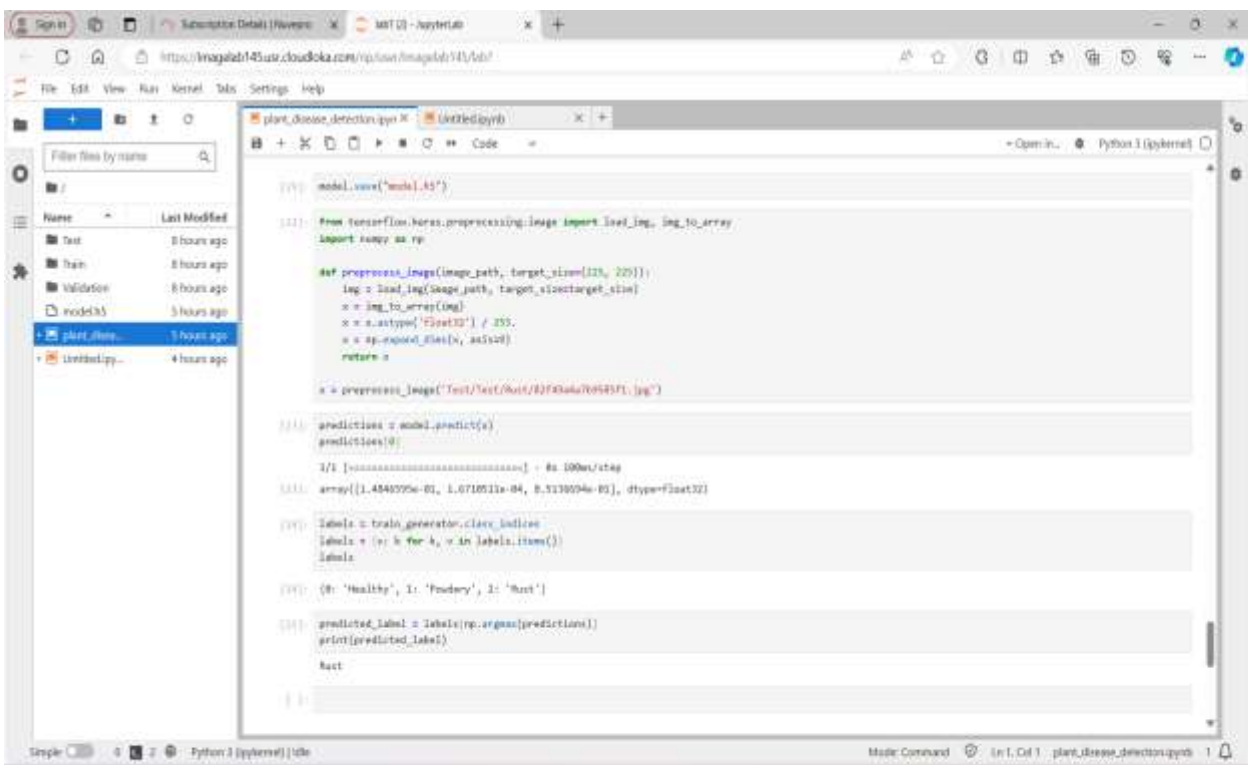

plant_disease_detection.ipynb
Untitled.ipynb
history = model.fit(train_generator,
Epoch 1/5
Epoch 2/5
Epoch 3/5
Epoch 4/5
Epoch 5/5
pip install matplotlib
Collecting matplotlib
Mode: Command


plant_disease_detection.ipynb
Untitled.ipynb
pip install seaborn
Collecting seaborn
Mode: Command


plant_disease_detection.ipynb
Untitled.ipynb
Installing collected packages: seaborn
Note: you may need to restart the kernel to use updated packages.
from matplotlib import pyplot as plt
from matplotlib.pyplot import figure
import seaborn as sns
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()


plant_disease_detection.ipynb
Untitled.ipynb
model accuracy
train
val
0.90
0.85
0.80
0.75
accuracy


plant_disease_detection.ipynb
Untitled.ipynb
Test
Train
Validation
model.h5
0.70
0.65
0.60
0.55
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
epoch


plant_disease_detection.ipynb
Untitled.ipynb
Test
Train
Validation
model.h5
predictions = model.predict(x)
{0: 'Healthy', 1: 'Powdery', 2: 'Rust'}
print(predicted_label)
Rust

CODE EXPLANATION:

1. Importing Libraries:

- os: Provides functions for interacting with the operating system, used for file and directory operations.
- PIL.Image: Used for image manipulation.
- IPython.display: Allows displaying images within the Jupyter Notebook.
- keras.preprocessing.image: Provides utilities for image data preprocessing.
- keras.models, keras.layers: Used to define and build the neural network model.
- matplotlib.pyplot, seaborn: Libraries for data visualization.
- numpy: Library for numerical operations.

2. Defining Functions:

- total_files: Counts the total number of files (images) in a given folder path.

3. Defining File Paths:

- Paths to the training, testing, and validation folders for each category of leaf images.

4. Printing Number of Files:

- Prints the number of files (images) in each category for training, testing, and validation sets.

5. Displaying Sample Images:

- Displays sample images from the training set using IPython.display.

6. Image Data Generators:

- ImageDataGenerator is configured for data augmentation and normalization for training and testing images.

7. Creating Data Generators:

- flow_from_directory method creates directory iterators for both training and validation data, generating batches of augmented images.

8. Defining the Convolutional Neural Network (CNN) Model:

- A sequential model is created with convolutional layers, max pooling layers, flattening layer, and dense layers with ReLU activation functions.
- Output layer has softmax activation for multiclass classification.

9. Compiling the Model:

- Configures the model for training with optimizer, loss function, and metrics.

10. Training the Model:

- fit method trains the model using data generated by the data generators.
- Training history is stored in history object.

11. Visualizing Training History:

- Plots model accuracy over epochs for both training and validation sets.

12. Saving the Model:

- The trained model is saved to a file named "model.h5".

13. Image Preprocessing Function:

- preprocess_image function loads and preprocesses an image for model prediction.

14. Making Predictions:

- A sample image is preprocessed and fed to the trained model to make predictions.
- Predicted label is obtained by mapping the index with the class labels.

15. Printing Predicted Label:

- Prints the predicted label for the sample image.

EXPLANATION OF PROJECT:

1. Data Collection:

- The project involves collecting a dataset of images of plant leaves, each labeled with the type of disease present (e.g., healthy, powdery mildew, rust).

2. Data Preprocessing:

- The collected dataset is likely to undergo preprocessing steps such as resizing images to a uniform size, splitting them into training, testing, and validation sets, and possibly augmenting the data to increase its diversity and robustness.

3. Model Architecture:

- A Convolutional Neural Network (CNN) architecture is chosen for this image classification task. CNNs are widely used for image classification tasks due to their ability to automatically learn features from images.

4. Model Training:

- The model is trained using the training dataset. During training, the model learns to classify images into different disease categories by adjusting its parameters based on the provided training data.

5. Model Evaluation:

- The trained model's performance is evaluated using the validation dataset. This step helps to assess how well the model generalizes to unseen data and whether it is overfitting or underfitting.

6. Visualizing Training History:

- The training history, including metrics such as accuracy and loss, is visualized using plots. This allows monitoring the model's performance over epochs and helps in identifying any issues such as overfitting or training convergence.

7. Model Deployment:

- Once the model achieves satisfactory performance on the validation set, it can be deployed for inference. This involves making predictions on new, unseen images to classify them into the appropriate disease categories.

8. Application:

- The trained model can be used in various applications, such as agricultural systems for automated disease detection in plants. It can help farmers and researchers identify diseased plants early, enabling timely intervention to prevent crop losses.